TUGAS AKHIR

PERPUSTAKAAN
I T S

Tgl. Terima	28-7-200
Terima Dari	H
No. Agenda Prp.	21.1373

RSE
629.89
Pra
p-1
1999

Disusun Oleh :

HERRYANTO PRASETYO

NRP : 2294100008

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1999

MILIK PERPUSTAK
ITS

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui,
Dosen Pembimbing

Ir. SOETIKNO
NIP. 130 445 231

S U R A B A Y A
Agustus, 1999

ABSTRAK

program Xilinx Foundation Series. Metode yang digunakan adalah Metod *Mixed-Design*, menggabungkan ABEL-HDL dan Skematik dalam desain hirarkis.

Model HDL Kontroler FPGA-Fuzzy pada FPGA XC4005XL yan terealisasi mempunyai kemampuan setara dengan IC NLX 221, misalnya, 4 bua *Input* dan *Output* digital 8 bit, *range* nilai *fuzzy* antar 0 sampai 63 (resolusi 6 bit) 6 tipe *Membership Function* dengan *floating membership function* dan *floatin action value*, dengan kemampuan penyimpanan *rule* internal sampai 63 variab *fuzzy*, dan 40 *rule* dalam 8 blok *output*.

Untuk kontroler FPGA-*Fuzzy* ini dibuat suatu program pendukung, FID (*Fuzzy Integrated Development Environment*). FIDE dibuat dengan menggunaka Microsoft Visual C++ 6.0 untuk beroperasi di Windows 95/98. Program i bermanfaat untuk menghasilkan *file-file* .mem yang digunakan untuk mengisi *rul* kontroler sebelum desain diimplementasikan ke FPGA dan *file* .hex untu pengisian *rule* dari RAM eksternal.

MILIK PERPUSTAKAAN ITS

KATA PENGANTAR

judul :

Pemanfaatan IC FPGA Xilinx sebagai Kontroler Berbasis Logika Fuzzy yang Dapat Dioptimasikan pada Suatu Aplikasi Tertentu.

Tugas akhir ini merupakan salah satu syarat dalam menyelesaika pendidikan di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institu Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dibuat berdasarkan teori-teori yang diperoleh selama mas perkuliahan dan berbagai buku literatur penunjangnya, bimbingan dari Dose Pembimbing serta informasi-informasi yang diperoleh melalui Internet.

Untuk itu Penulis ingin mengucapkan terima kasih sebesar-besarny kepada :

- Bapak Ir. Soetikno, selaku Dosen Pembimbing dan Koordinator Bidang Stud Elektronika Jurusan Teknik Elektro ITS, yang telah memberi ide untu pembuatan Tugas Akhir ini serta berbagai informasi berharga da bimbingannya.
- Bapak Ir. Harris Pirngadi, MT, selaku Dosen Wali yang telah mendukun penulis untuk menyelesaikan Tugas Akhir ini.
- Bapak Ir. Teguh Yuwono, selaku Ketua Jurusan Teknik Elektro ITS yan telah memberikan persetujuan atas Tugas Akhir ini.

- Teman-teman, Mahendro, Susanto, Andy, Eddy, yang selalu menyemangat hingga selesainya Tugas Akhir ini.

Penulis berharap kiranya Tugas Akhir ini dapat bermanfaat bagi para Pembaca, khususnya bagi teman-teman di Bidang Studi Elektronika Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya, terutama untuk pengembangan penggunaan IC FPGA Xilinx dan model HDL untuk Kontroler Fuzzy.

Surabaya, Agustus 1999

Penulis

DAFTAR IS

KATA PENGANTAR i

DAFTAR ISI v

DAFTAR GAMBAR x

DAFTAR TABEL x

BAB 1 PENDAHULUAN .....

1.1. Latar Belakang .....

1.2. Permasalahan .....

1.3. Tujuan .....

1.4. Pembatasan Masalah .....

1.5. Metodologi .....

1.6. Sistematika .....

1.7. Relevansi .....

BAB 2 FIELD PROGRAMMABLE GATE ARRAY (FPGA) 7

2.1. IC FPGA Xilinx XC4005XL 7

2.1.1. Deskripsi Umum 7

2.1.2. Penjelasan Modul-Modul yang Digunakan .....

2.1.2.1. Global Set-Reset .....

2.1.2.6. Global Nets dan Global Buffers ....
2.2. Xilinx Foundation Series ....
2.2.1. Deskripsi Umum ....
2.2.2. Alur Desain dengan Xilinx Foundation Series ....
2.2.3. Metode Mixed-Design ....
2.3. XS-40 Board dan XStend ....
2.3.1. Deskripsi Umum ....
2.3.2. Petunjuk Pengoperasian XS-Board – XStend ....
BAB 3 KONTROLER FPGA-FUZZY ....
3.1. Pendahuluan ....
3.1.1. Spesifikasi Kontroler FF ....
3.1.2. Deskripsi ....
3.1.3. Daftar Pin ....
3.1.4. Deskripsi Pin ....
3.1.4.1. Pin-Pin Input ....
3.1.4.2. Pin-Pin Output ....
3.2. Cara Kerja Kontroler FF ....
3.2.1. I/O & Fix Parameter Storage ....

3.3. Arsitektur Kontroler 4

3.3.1. Rangkaian Clock 4

3.3.2. Rangakaian 7-Segment 4

3.3.3. I/O & Fix Parameter Storage 4

3.3.4. Program Storage 4

3.3.5. Kontroler Logika 4

3.3.5.1. Initialization Stage 4

3.3.5.2. Programming Stage 4

3.3.5.3. Fetching Code Stage 4

3.3.5.4. Term Evaluation Stage 4

3.3.5.5. EOR Stage 5

3.3.5.6. EORRSO Stage 5

3.3.6. Fuzzifier 5

3.3.7. Rule Evaluator 5

3.3.7.1. Minimum Calculator 5

3.3.7.2. Maximum Calculator 5

3.3.8. Defuzzifier 5

4.2.1. Pemasukan Parameter dan Rule 5
4.2.2. Pembuatan File-File .mem dan .hex 6
4.2.3. Simulasi Kontroler 6
4.3. Penjelasan Program 6
4.3.1. Dokumen Utama (FIDE) 6
4.3.2. Dokumen File 7
4.3.3. View Utama (FIDE) 7
4.3.4. View Term 7
4.3.5. View Simulator 7
4.3.6. View File 7
4.3.7. Fungsi Konversi 7
4.3.8. Fungsi Simulasi 7

BAB 5 PENGUJIAN KONTROLER FF DAN PROGRAM FIDE 7
5.1. Pendahuluan 7
5.2. Pengujian Kontroler FPGA-Fuzzy dan Program FIDE 8
5.2.1. Pengujian Program FIDE 8
5.2.2. Pengujian Kontroler FPGA-Fuzzy 8

DAFTAR PUSTAKA .....

LAMPIRAN

LAMPIRAN A : DATASHEET XC4005XL (23)

LAMPIRAN B : MODEL HUBUNGAN XS-40 & XSTEND BOARD (2)

LAMPIRAN C : DESAIN KONTROLER FPGA-FUZZY (19)

C.1. Fuzzy.ucf (2)

C.2. Top-Level Schematic : FUZZY (3)

C.3. Macro-Level Schematic : F_IOREG

C.4. Macro-Level Schematic : F_PGMSTO

C.5. Macro-Level Schematic : F_CNTRLR

C.6. Macro-Level Schematic : F_FUZZFR

C.7. Macro-Level Schematic : F_EVLTR

C.8. Macro-Level Schematic : F_DFZZFR

C.9. Macro-Level Schematic : F_ASUB8

C.10. Macro-Level Schematic : F_ALFCUT

C.11. Macro-Level ABEL-HDL : f fsmcon.abl (3)

C.12. Macro-Level ABEL-HDL : F_AlfDcd.abl

MILIK PERPUSTAKA
ITS

D.3. FIDEParDlg.h (4)

D.4. FIDEParDlg.cpp (18)

D.5. FIDETrmV.h (2)

D.6. FIDETrmV.cpp (6)

D.7. FIDESIM.h (3)

D.8. FIDESIM.cpp (9)

D.9. FIDEDoc.h (2)

D.10. FIDEDoc.cpp (12)

D.11. FIDEView.h (2)

D.12. FIDEView.cpp (8)

D.13. FIDEFile.h (2)

D.14. FIDEFile.cpp (6)

DAFTAR GAMBAR

(Tanpa Fungsi RAM & Carry) 9

Gambar 2.3. Arsitektur FPGA Secara Umum 9

Gambar 2.4. Cara Instansiasi Komponen GSR pada Skematik 1

Gambar 2.5. Model Sederhana *Dual-Port RAM* XC4005XL 1

Gambar 2.6. *Fast Carry Logic* XC4005XL 1

Gambar 2.7. Diagram Blok IOB XC4005XL yang Disederhanakan 1

Gambar 2.8. Simbol Osilator untuk XC4005XL 1

Gambar 2.9. Alur Desain Menggunakan **Xilinx Foundation Series** 1

Gambar 2.10. Blok Diagram Hubungan FPGA dan Komponen-Komponen pada **XS-40 Board** 2

Gambar 2.11. Diagram Tata Letak **XS-40 Board** ver. 1.2 2

Gambar 2.12. Diagram Tata Letak **XStend Board** 2

Gambar 3.1. Diagram Blok **Kontroler FPGA-Fuzzy** yang Disederhanakan .. 3

Gambar 3.2. Diagram Blok **Fuzzy Processor** 3

Gambar 3.3. Peta Memori **I/O & Fix Parameter Storage** 3

Gambar 3.4. Peta Memori **Program Storage** 3

Gambar 3.5. Format Kode untuk **TE** 3

Gambar 3.6. Format Kode untuk **EOR** 3

Gambar 3.7. Format Kode untuk **EORRSO** 3

Gambar 3.12. Diagram Blok Modul **Defuzzifier** 4

Gambar 4.1. Tampilan Awal **FIDE** 5

Gambar 4.2. Dialog **Input, Output** dan **Term** 6

Gambar 4.3. Dialog **Rule** 6

Gambar 4.4. Window **Term Viewer** 6

Gambar 4.5 Window **Simulator** 6

Gambar 5.1. *Input, Output* dan *Term-Term* yang digunakan untuk pengujian . 7

Gambar 5.2. *Rule* yang digunakan untuk pengujian 8

DAFTAR TABEL

BAB 1

PENDAHULUAN

Logika *fuzzy* sangat bermanfaat untuk mengatasi masalah-masalah yang non linier dan dinamis, dan dengan adanya teknologi kontroler *fuzzy*, dimungkinkan penerapan logika *fuzzy* dalam berbagai bidang secara relatif lebih mudah dan cepat. Tetapi, karena kontroler *fuzzy* ini bersifat umum, sering terdapat kekurangan atau sifat-sifatnya yang tidak sesuai dengan kebutuhan.

Teknologi HDL yang kini sedang berkembang memungkinkan dibuatnya model dari suatu IC. Model ini kemudian dapat di-*customize* sesuai kebutuhan suatu aplikasi tertentu atau digunakan sebagai *macro*, *core* desain dan digunakan sebagai bagian dari desain yang lebih besar. Dengan adanya teknologi ini dimungkinkan pengembangan kontroler *Fuzzy* yang dapat dioptimasikan pada suatu aplikasi tertentu.

Untuk memfasilitasi pengujian dan pengembangan model HDL tersebut digunakan IC FPLD (*Field Programmable Logic Device*) yang merupakan superset dari PLA, PAL dan PLD dengan kemampuan *reprogrammable* sangat tinggi. Salah satunya, adalah IC FPGA yang dibuat XILINX yang merupakan *leading-vendor* dalam pasar FPGA dunia saat ini.

1.2. Permasalahan

Terdapat dua masalah dalam Tugas Akhir ini, yaitu masalah *hardware* dan *software* yang saling berkaitan untuk menghasilkan suatu fungsi kontroler *fuzzy* yang utuh.

fuzzy dalam format yang dimengerti Kontroler dan untuk membuat fungsi simulasi yang semirip mungkin dengan operasi Kontroler sebenarnya.

1.3. Tujuan

Tujuan Tugas Akhir ini adalah untuk membuat model HDL dari Kontroler Fuzzy, mengimplementasikan dan mengujinya dengan menggunakan IC FPGA Xilinx XC4005XL dengan bantuan **XS-40 Board** dan **XStend** dan membuat program pendukung Kontroler tersebut sehingga diperoleh suatu Kontroler Fuzzy yang **bisa digunakan**..

Kontroler Fuzzy yang **bisa digunakan** tersebut bukan merupakan tujuan akhir, tetapi baru merupakan awal pengembangan kontroler Fuzzy berbasis teknologi HDL. Dengan adanya model dasar ini dimungkinkan pengembangan lebih lanjut untuk menciptakan suatu kontroler yang spesifik untuk suatu aplikasi tertentu, sesuai judul Tugas Akhir ini.

1.4. Pembatasan Masalah

Terdapat beberapa pembatasan dalam Tugas Akhir ini :

- Kontroler FPGA-Fuzzy dibuat berdasarkan model IC NLX-221,
- Program FIDE dibuat menggunakan Microsoft Visual C++ 6.0 dengan memanfaatkan segala fasilitas *visual*-nya dan dukungan teknologi *ActiveX*,
- Tugas Akhir ini ditujukan untuk membuat model HDL Kontroler Fuzzy dan program pendukungnya sehingga dapat diperoleh suatu Kontroler Fuzzy yang **bisa digunakan** tanpa membuat suatu aplikasi untuk Kontroler Fuzzy tersebut,
- Untuk keperluan demo akan digunakan suatu contoh aplikasi kontroler *fuzzy* umum dengan mensimulasikan *input* dengan *DIP-switch* dan menampilkan *output* ke *7-Segment*.

1.5. Metodologi

Untuk mencapai tujuan yang telah disebutkan di atas, dilakukan langkah-langkah sebagai berikut :

Langkah pertama adalah mempelajari cara kerja dan struktur internal suatu Kontroler Fuzzy. Untuk ini dipelajari dua model, NLX 230 dan NLX 221 di mana akhirnya diputuskan untuk memilih NLX-221 karena arsitekturnya yang lebih maju dan cocok untuk FPGA yang akan digunakan.

kepentingan Tugas Akhir ini. Langkah pertama dan kedua berjalan secara iteratif, dilakukan berulang-ulang dengan saling mempertimbangkan satu sama lain untuk memperoleh gambaran struktur internal Kontroler FPGA-Fuzzy (FF) yang akan dibuat.

Langkah ketiga adalah pembuatan desain model HDL dari Kontroler FPGA-Fuzzy. Hal ini dilakukan dengan program **Xilinx Foundation Series** menggunakan metode *mixed-design*.

Langkah keempat adalah pembuatan bagian pertama program FIDE, yaitu merealisasikan fungsi pembuatan *file-file rule* yang dibutuhkan untuk pengujian Kontroler FF.

Langkah kelima adalah pengujian desain Kontroler FF dengan menggunakan *file-file rule* yang dihasilkan program FIDE. Pengujian dilakukan baik secara *software* dengan menggunakan fungsi simulator **Xilinx Foundation Series** sampai secara *hardware* dengan bantuan **XS-40 Board** dan **XStend**.

Langkah keenam adalah penyempurnaan program FIDE dengan menambahkan fungsi Simulator sesuai cara kerja internal Kontroler FF.

Langkah ketujuh adalah pengujian desain secara keseluruhan, *hardware* maupun *software* dengan menggunakan suatu contoh aplikasi *fuzzy* umum. Pengujian secara *hardware* dilakukan dengan bantuan **XS-Board** dan **XStend**.

disatukan menjadi Laporan Tugas Akhir ini.

1.6. Sistematika

Sistematika pembahasan dalam penyusunan buku Tugas Akhir ini adalah sebagai berikut :

- BAB 1 merupakan pendahuluan yang menjelaskan latar belakang, permasalahan, tujuan, pembatasan masalah, metodologi, sistematika dan relevansi dari Tugas Akhir ini.
- BAB 2 memberikan dasar teori secara singkat tentang *Field Programmable Gate Array* (FPGA) : deskripsi dan struktur internal FPGA, paket program yang digunakan untuk mendesain dengan FPGA dan *prototyping-board* yang digunakan untuk demo.
- BAB 3 menjelaskan tentang Kontroler FPGA-Fuzzy yang telah berhasil direalisasikan, mulai dari deskripsi secara umum, cara kerja sampai struktur internalnya.
- BAB 4 berisi penjelasan program *Fuzzy Integrated Development Environment* (FIDE) yang dibuat dengan Microsoft Visual C++ 6.0 sebagai bagian dari Tugas Akhir ini untuk mendukung pengoperasian Kontroler FPGA-Fuzzy.

- BAB 6 adalah penutup Tugas Akhir ini yang berisi Kesimpulan dan Saran.

1.7. Relevansi

Tugas Akhir ini meletakkan dasar pengembangan model HDL untuk kontroler *fuzzy* yang kemudian dapat digunakan dan disempurnakan serta dioptimasikan untuk suatu aplikasi tertentu.

Tugas Akhir ini juga merupakan dasar bagi pengembangan penggunaan IC FPGA Xilinx di Bidang Studi Elektronika Jurusan Teknik Elektro ITS.

BAB 2

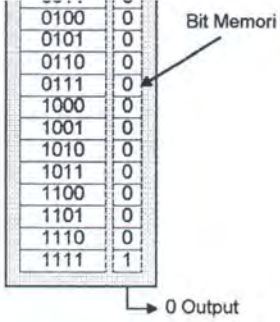
FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Untuk realisasi *hardware* **Kontroler Fuzzy-FPGA** ini digunakan IC FPGA XC4005XL buatan Xilinx. IC ini beroperasi dengan *power-supply* 3.3 V, sedang *input/output*-nya dapat dikonfigurasi sebagai TTL atau CMOS.

2.1.1. Deskripsi Umum

IC FPGA adalah sejenis IC FPLD, *Field Programmable Logic Device*, suatu IC yang dapat diprogram untuk menentukan fungsinya secara *hardware*, jadi bukan seperti keluarga kontroler dan prosesor yang diprogram secara *software*, sedangkan fungsi *hardware*-nya sudah terdefinisi sebelumnya.

FPGA buatan Xilinx terdiri dari blok-blok pembangun dasar yang berupa *lookup table* atau LUT. Perhatikan Gambar 2.1 yang menggambarkan LUT dengan 4 *input* dan memori kecil berukuran 16x1 bit. Input yang diberikan pada LUT akan menjadi alamat suatu bit memori tertentu sehingga LUT akan mengeluarkan nilai tersebut sebagai nilai *output*-nya. Hal ini memungkinkan membangun rangkaian logika apapun, asalkan mempunyai jumlah *input* 4 dan 1 *output* dalam LUT tersebut. Sebagai contoh, suatu fungsi AND-4-*input* dibuat

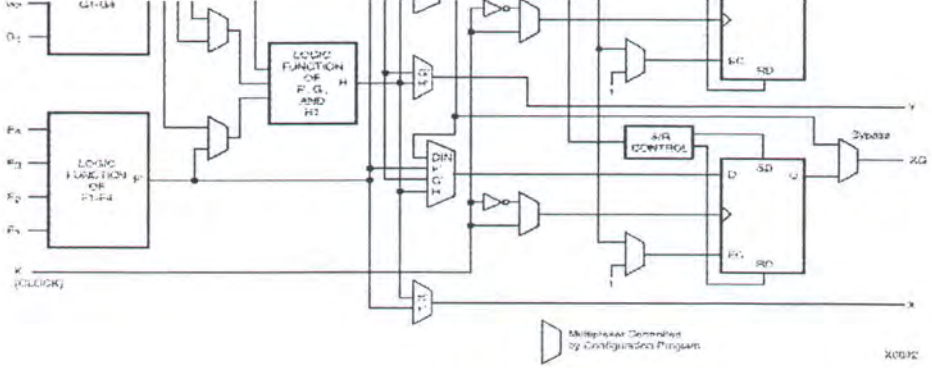


Gambar 2. 1.[1]

LUT 4 Input yang Diimplementasikan Sebagai AND-4-Input

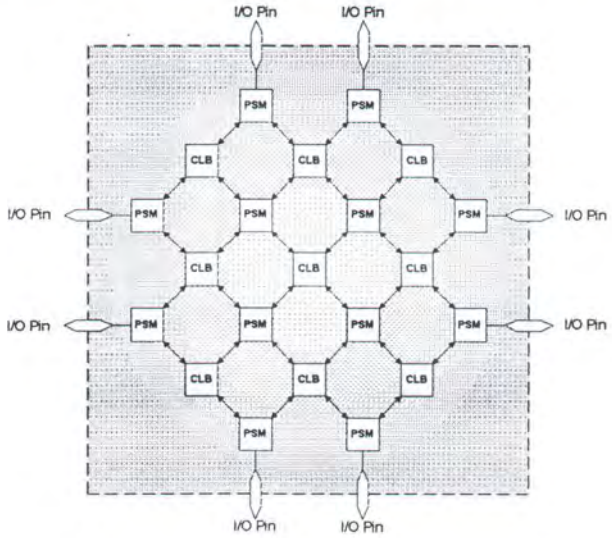
Pada IC XC4005XL, 3 buah LUT dikombinasikan dengan 2 *flip-flop* dan beberapa rangkaian kendali untuk membentuk suatu blok rangkaian logika yang dapat dikonfigurasi, *Configurable Logic Block* (CLB) seperti pada Gambar 2.2. Lalu CLB-CLB diatur dalam suatu *array* dengan matrik saklar yang dapat diprogram, *Programmable Switch Matrices* (PSM) di antaranya (Gambar 2.3.). PSM ini digunakan untuk menghubungkan *output* dari satu CLB ke *input* CLB yang lain. Pin-pin I/O FPGA dapat dihubungkan ke PSM dan CLB atau matrik hubungannya sendiri yang juga kompleks. Kebanyakan FPGA mempunyai lebih banyak CLB daripada pin I/O sehingga CLB tidak dapat mempunyai hubungan

[1] [10] **The Practical Xilinx Designer Lab Book**, p. 30.



Gambar 2. 2.[2]

Blok Diagram CLB XC4005XL yang Disederhanakan (Tanpa Fungsi RAM & Carry)



Gambar 2. 3.[3]

Arsitektur FPGA Secara Umum

---

[2] [3] **The Programmable Logic Data Book 1998**, p. 4-10.
[3] [10] **The Practical Xilinx Designer Lab Book**, p. 31.

storage element, yang mencatat apakah saklar yang dikendalikannya terbuka atau tertutup. Perubahan nilai pada elemen ini mengubah keadaan saklar dan mengubah pula fungsi FPGA. XC4005 menggunakan *Static Random Access Memory* (SRAM) sebagai elemen penyimpan sehingga bila aliran *power* terhenti, FPGA harus diprogram ulang.

Pemrograman FPGA dilakukan dengan menggunakan *file* .BIT yang merupakan hasil implementasi desain yang dibuat. Proses pembuatan desain sampai implementasi dan pembuatan *file* .BIT dilakukan dengan paket program **Xilinx Foundation Series** yang akan dijelaskan pada bagian **2.2**.

2.1.2. Penjelasan Modul-Modul yang Digunakan

Arsitektur FPGA Xilinx XC4000XL sangat teratur dan mirip *array* gerbang, tetapi bukan hanya kumpulan gerbang sederhana. Banyak terdapat sumber daya dan struktur tertentu yang dapat dimanfaatkan untuk menghasilkan desain yang efisien baik densitas maupun kecepatannya.

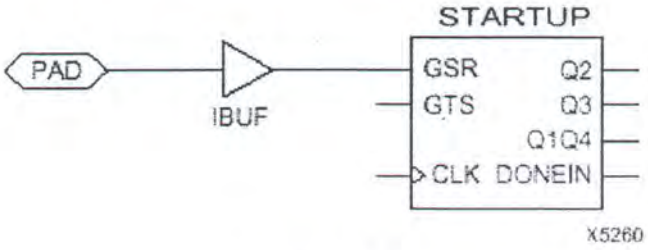
Dalam makalah ini akan dibahas beberapa sumber daya dan struktur yang akan digunakan dalam desain, yaitu :

- *Global Set-Reset,*

MILIK PERPUSTAKAAN
ITS

2.1.2.1. Global Set-Reset

Terdapat suatu jalur khusus untuk *Global Set-Reset* (GSR) untuk *set* atau *clear* setiap elemen penyimpan saat *power-up*, konfigurasi ulang atau ketika jalur *Reset* khusus diaktifkan. Jalur GSR ini menggunakan jaringan distribusi khusus yang tidak mengganggu jalur-jalur penghubung lain.



Gambar 2. 4.[4]

Cara Instansiasi Komponen GSR pada Skematik

Setiap *flip-flop* dapat dikonfigurasi untuk *set* atau *reset* dengan GSR, hal ini menyebabkan elemen penyimpan FPGA dapat diinisialisasi saat *power-up*. GSR juga dapat dimanfaatkan untuk membuat jalur *Reset* khusus pada desain. Hal ini dilakukan dengan menginstansiasi simbol **STARTUP** pada skematik atau kode

[4] [3] **The Programmable Logic Data Book 1998**, p.4-11.

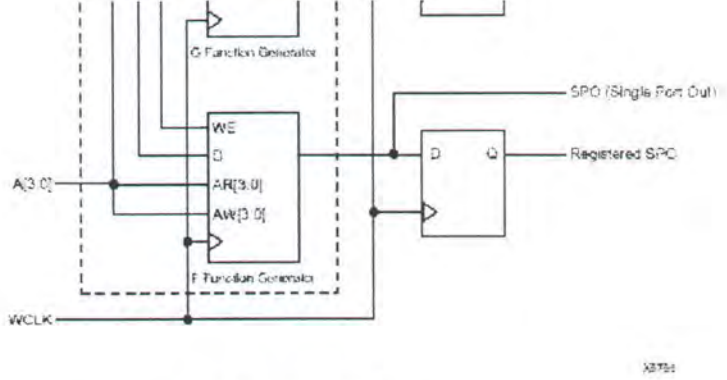
Unit dasar yang berupa LUT membuka suatu kemungkinan untuk menggunakan LUT tersebut sebagai RAM. Pada XC4005XL terdapat beberapa jenis RAM :

- *Single-Port Level-Sensitive RAM,*
- *Single-Port Edge-Triggered RAM,*
- *Dual-Port Edge-Triggered RAM.*

Level-Sensitive RAM berfungsi seperti *latch*, RAM dapat ditulisi secara asinkron, bergantung level sinyal pada pin **WE**-nya, sedang *Edge-Triggered RAM* berfungsi seperti *D-flip-flop* ditulisi pada transisi *clock*.

Perlu diperhatikan tentang *Dual-Port* RAM, komponen ini memungkinkan untuk melakukan operasi baca dan tulis secara bersamaan pada RAM, bahkan pada isi alamat yang sama. Hal ini dimungkinkan dengan cara menghubungkan secara paralel dua RAM dengan isi yang sama. Pada komponen ini, operasi penulisan akan dilakukan secara paralel terhadap kedua RAM, sedangkan operasi pembacaan dapat dilakukan pada lokasi yang berbeda pada masing-masing RAM. Dengan kata lain, terdapat satu *data / address bus* masukan (*write*) dan dua *data/address bus* keluaran (*read*). **A[3:0]** adalah *address bus* yang berhubungan dengan **D[7:0]** dan **SPO[7:0]** (alamat untuk menulis dan membaca) sedang

MILIK PERPUST
ITS



Gambar 2. 5.[5]

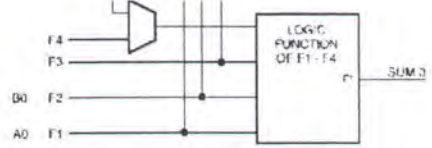
Model Sederhana *Dual-Port RAM* XC4005XL

2.1.2.3. Fast Carry Logic

Selain untuk mengimplementasi- kan logika dan penyimpanan data, CLB juga berisi *hardware* khusus yang digunakan untuk mempercepat penghasilan *carry* untuk *adder* dan *counter*. Juga terdapat jalur khusus untuk menghubungkan *hardware* khusus ini satu sama lain untuk menghasilkan *adder* dan *counter* multi-bit. Pemanfaatan *hardware* dan jalur khusus ini bermanfaat untuk menghasilkan *adder* dan *counter* yang cepat dan efisien.

---

[5] Ibid, p.4-15.



Gambar 2. 6.[6]

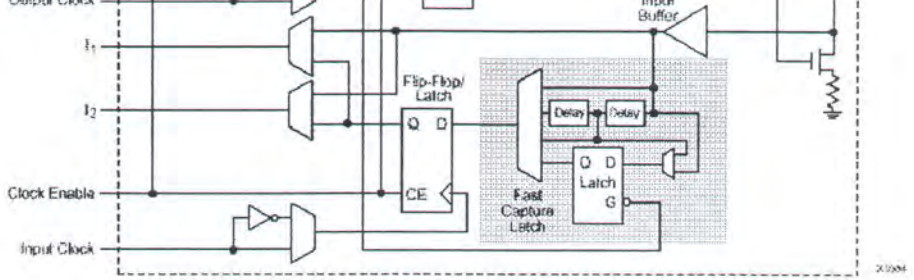
Fast Carry Logic XC4005XL

2.1.2.4. IOB

Untuk menghubungkan rangkaian logika internal FPGA dengan pin eksternal digunakan *Input/Output Blocks* (IOBs). Setiap IOB mengendalikan satu pin dan dapat dikonfigurasikan sebagai *input*, *output* atau sinyal *bidirectional*.

Tampak pada diagram, bahwa dalam IOB terdapat beberapa sumber daya yang digunakan untuk menghasilkan tipe *interface* yang diinginkan dengan dunia luar :

- *Register input* dapat dikonfigurasi sebagai *flip-flop* atau *latch*, dengan pilihan *inverted* atau *non-inverted clock*,
- *Input* dapat dikonfigurasi sebagai TTL atau CMOS,
- Data *input* dapat diberi *delay* untuk meningkatkan waktu *setup* untuk *flip-flop input* sehingga tidak dibutuhkan positif *hold-time* untuk jalur *clock*,



Gambar 2. 7.[7]

Diagram Blok IOB XC4005XL yang Disederhanakan

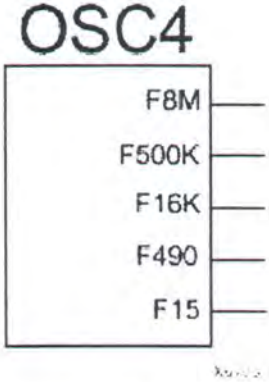
- Terdapat *latch* tambahan dengan sumber *clock* dari *output clock*, hal ini memungkinkan memberikan *clock* yang berbeda pada kedua unit penyimpan *input* pada IOB tersebut sehingga unit penyimpan pertama dapat digunakan sebagai *fast-capture latch*, penerima data sementara secara cepat untuk kemudian disinkronisasikan dengan *clock* internal pada *register* kedua,
- *Output* dapat di-*invert* secara internal dalam IOB, langsung ke pin, atau disinkronisasi dengan *flip-flop* terlebih dahulu,
- *Output* dapat dikonfigurasikan sebagai *three-state* atau *bidirectional*, dapat dihubungkan ke Vcc dengan *pull-up* atau ke Gnd dengan *pull-down*,

---

[6] [7] **XAPP 043.00 Improving XC4000 Design Performance**, p. 8-23.
[7] [3] **The Programmable Logic Data Book 1998**, p.4-22.

2.1.2.5. On-Chip Oscillator

XC4005XL mempunyai osilator internal. Osilator ini digunakan untuk kepentingan pemrograman FPGA dan inisialisasi awal setelah *power-up*. Untuk ini osilator berjalan pada frekuensi nominal 8 MHz yang bervariasi antara 4 dan 10 MHz sesuai proses, Vcc dan suhu.



Gambar 2. 8.[8]

Simbol Osilator untuk XC4005XL

Setelah konfigurasi, osilator dapat digunakan untuk sumber *clock* pada desain. Untuk kepentingan ini, selain frekuensi nominal 8 MHz disediakan pula frekuensi 500kHz, 16kHz, 490Hz dan 15Hz yang dapat bervariasi antara -50%

[8] Ibid, p.4-28.

- *Global Low-Skew Buffers (BUFGLS),*
- *Global Early Buffers (BUFGE).*

Global Low-Skew Buffers adalah *buffer clock* standar. *Buffer* ini harus digunakan untuk setiap jalur *clock* internal atau jalur yang men-*drive* sebagian besar *device*. *Global Early Buffers* didesain untuk menghasilkan akses *clock* yang lebih cepat, tetapi akses CLB dibatasi untuk seperempat *device*. Kedua jenis *buffer* tersebut dikendalikan dari pin yang sama. Karenanya sebuah pin dapat men-*drive* satu *buffer* dari tiap tipe secara bersamaan.

2.2. Xilinx Foundation Series

Untuk membuat desain *hardware* **Kontroler FPGA-Fuzzy** digunakan paket program **Xilinx Foundation Series** buatan **Xilinx, Inc.**

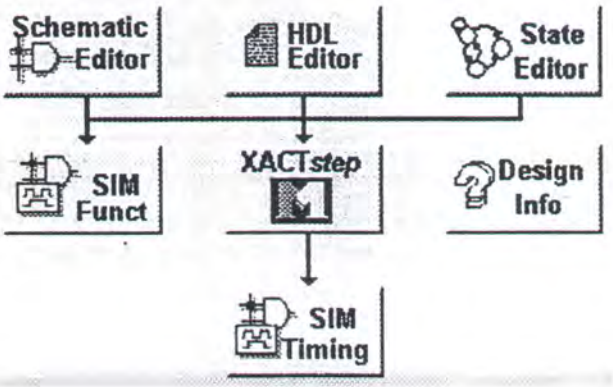
2.2.1. Deskripsi Umum

Xilinx Foundation Series merupakan suatu paket program buatan **Xilinx, Inc.** yang digunakan sebagai lingkungan pengembangan terpadu, *Integrated*

- Melakukan simulasi fungsional berdasarkan *netlist* tersebut,
- Mengimplementasikan desain untuk menghasilkan *file* .BIT yang dapat digunakan untuk mengkonfigurasi FPGA, dan
- Melakukan simulasi waktu berdasarkan hasil implementasi desain tersebut.

2.2.2. Alur Desain dengan Xilinx Foundation Series

Pertama-tama, desain dimasukkan, baik berbentuk skematik, kode HDL, *state-machine* atau gabungan ketiganya (*mixed-design*). Penjelasan mengenai metode *mixed-design* dapat diperoleh pada bagian berikutnya. Setelah itu, desain disintesis menjadi *netlist*.



Gambar 2. 9.

Alur Desain Menggunakan **Xilinx Foundation Series**

implementasi, M1 dan siap diimplementasikan ke FPGA. Proses implementasi dibuat sangat mudah dengan *flow* yang memudahkan pengontrolan proses. Bila akan dilakukan simulasi waktu, *Timing Simulation*, perlu dibuat data waktu pada waktu implementasi ini.

Timing Simulation merupakan suatu program yang memungkinkan simulasi operasi FPGA yang cukup mendekati operasi nyatanya di *hardware*. Data waktu yang dibuat pada proses implementasi digunakan dalam simulasi sehingga diperoleh *delay* eksekusi dan proses-proses yang hampir sama dengan keadaan operasi sebenarnya. Simulasi waktu ini merupakan tahap pengujian desain secara *software* yang terakhir dan paling tepat. Proses perbaikan dan simulasi waktu dapat dilakukan berkali-kali sampai desain dapat disimulasikan dan beroperasi sesuai keinginan. Berdasarkan pengalaman Penyusun, karena kualitas IC FPGA yang cukup bagus, bila desain telah beroperasi dengan benar pada simulasi waktu, operasi nyatanya di *hardware* pun akan berjalan dengan lancar.

grafis dengan deskripsi HDL-nya, karena desain yang dimasukkan dalam bentuk *state* akan diubah dulu dan disintesis dalam bentuk HDL-nya). Juga didukung cara *mixed-design*, pencampuran berbagai macam tipe desain dalam satu desain. Untuk metode *mixed-design* ini, *top-level* desainnya harus berupa skematik, sedang desain-desain dengan tipe yang lain disintesis menjadi *macro* untuk kemudian dimasukkan dalam desain berupa komponen.

Dalam Tugas Akhir ini digunakan metode terakhir, yaitu metode *mixed-design*. Pertama dibuat skematik untuk modul-modul yang digunakan, seperti **I/O & Fix Value Registers**, **Program Storage**, **Fuzzifier**, **Rule Evaluator** dan **Defuzzifier** yang masing-masing kemudian akan disintesis menjadi *macro* dan akan disatukan dengan bagian-bagian lainnya dalam skematik *top-level* yang akan disintesis ke IC.

Pemilihan metode ini karena adanya kebebasan yang besar dalam memasukkan desain, misalnya untuk fungsi-fungsi aritmetika umum, dapat digunakan komponen-komponen aritmetika yang sudah terdapat **Xilinx Unified Library**, pustaka umum Xilinx, dapat pula digunakan *template* komponen, **LOGIBLOX** untuk menghasilkan komponen-komponen yang dapat

buffer khusus, rangkaian *startup* dan sebagainya yang yang sulit diakses pada tingkat HDL (terutama untuk ABEL-HDL).

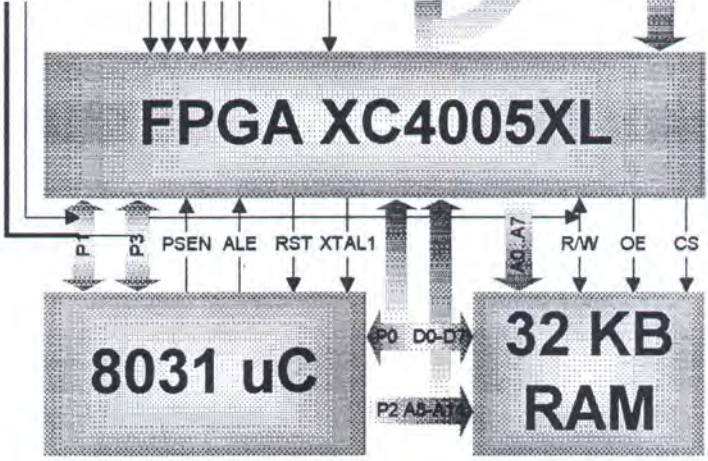
Keuntungan lainnya, adalah lebih transparannya realisasi desain pada IC. Untuk pemasukan desain pada tingkat HDL, realisasi sebenarnya pada IC hampir sepenuhnya bergantung pada *compiler*. Sulit dibayangkan hasil akhir suatu desain. Perubahan sedikit pada desain akan membawa perubahan yang sangat besar pada jumlah CLB yang dibutuhkan. Pada tingkat skematik, realisasi desain lebih transparan, walaupun ada penyederhanaan dan perubahan yang dilakukan *compiler*, tetapi sejak awal sudah dapat diperkirakan mengenai jumlah CLB yang dibutuhkan sehingga dapat dibuat desain dengan lebih efisien

2.3. XS-40 Board dan Xstend

Untuk keperluan demo digunakan **XS-40 Board** ver. 1.2 dan **Xstend** ver. 1.3 buatan **X Engineering Software Systems Corp. (XESS)**.

2.3.1. Deskripsi Umum

XS-40 Board adalah suatu *prototype-board* untuk IC FPGA Xilinx XC4005XL. **XS-Board** disertai dengan paket program, **XS Tools** yang terdiri dari **XSLoad** untuk *download bitstream* untuk konfigurasi FPGA atau *file* .hex Intel



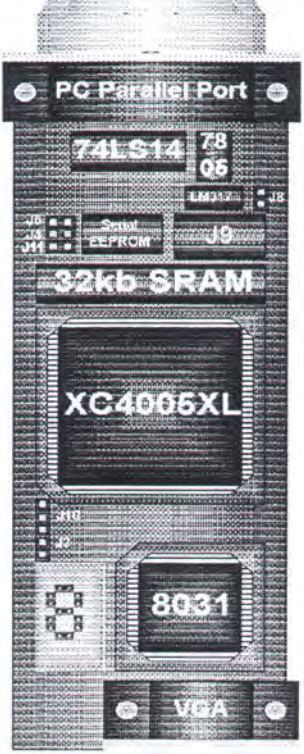
Gambar 2. 10.[9]

Blok Diagram Hubungan FPGA dan Komponen-Komponen pada **XS-40 Board**

Sebagai *prototype-board,* terdapat beberapa komponen pendukung untuk keperluan eksperimen :

- Mikro kontroler 8031 yang dapat digunakan untuk membantu fungsi yang dijalankan oleh FPGA,
- Osilator 12 MHz untuk sumber *clock* FPGA dan Mikro Kontroler,
- 32k x 8 SRAM yang digunakan untuk menyimpan program untuk mikro kontroler atau untuk menyediakan data bagi FPGA,

[9] [5] **XS40, XSP, and XS95 Board Manual**, p.13.



XS40 Board

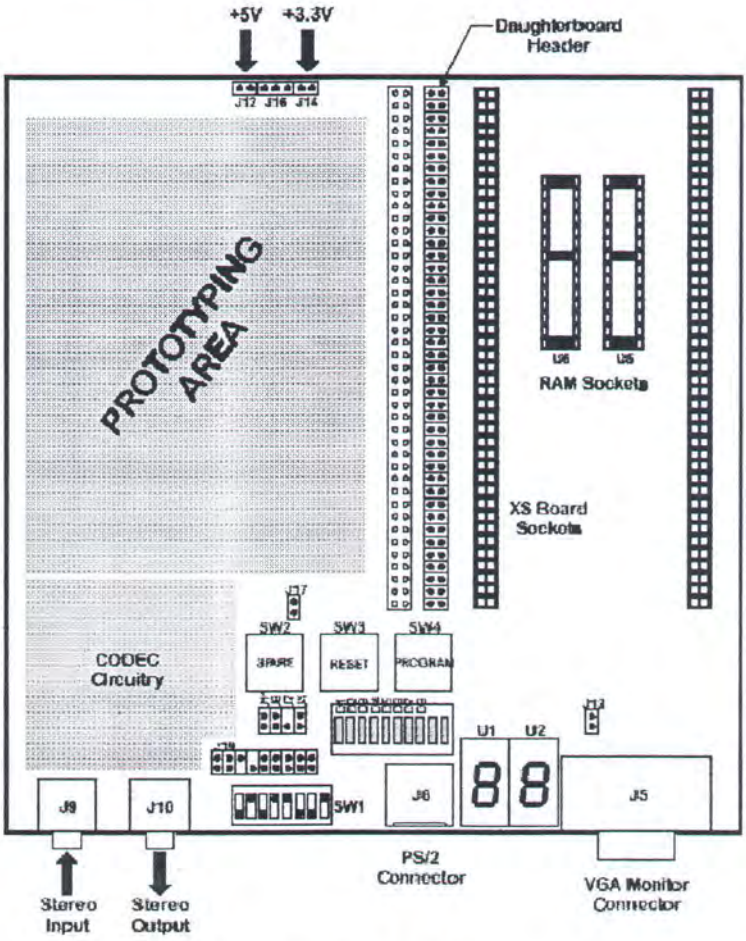
Gambar 2. 11.[10]

Diagram Tata Letak **XS-40 Board** ver 1.2

- *Power-regulator* untuk mengubah tegangan sekitar 8 – 10 V menjadi tegangan 3.3 V dan 5V yang diperlukan untuk operasi komponen-komponen di XS-Board;

[10] [7] Ibid, p.5.

kemungkinan eksperiman desain FPGA :



Gambar 2. 12.[11]

Diagram Tata Letak **XStend Board**

[11] [6] **XStend Board V1.3 Manual**, p.5.

untuk memproses sinyal audio dalam kombinasi dengan FPGA yang diimplementasikan sebagai DSP.

2.3.2. Petunjuk Pengoperasian XS-Board - XStend

1. Tempatkan **XS-40 Board** pada bagian kanan atas **XStend**, pada bagian paling kanan dari dua kolom soket 84 pin, dan pastikan semua pin pada pinggir **XS-Board** masuk ke soket.
2. Setting *jumper* sebagai berikut :
   XS-40 Board : J4, J8, J11 : On; J6, J10 : Off; J7 : tidak digunakan
   XStend Board : J4, J7, J11 : On; J8, J13, J17 : Off; J16 : tidak digunakan
3. Implementasikan desain, buat file bitstream untuk FPGA.
4. Masuk ke sesi **DOS prompt.**
5. Pindah ke *subdirectory* tempat *file bitstream* atau **.SVF** berada
6. Pastikan *power supply* terhubung pada **XS Board**. 7-segment akan menyala '8'. *Power-Supply* yang diberikan harus bernilai sekitar 8V – 10V, dan tidak diperlukan regulasi yang terlalu baik, karena **XS-40 Board** mempunyai rangkaian regulasi internal untuk memberikan tegangan 3.3V dan 5V yang diperlukan **XS-40 Board** dan **XStend Board**.
7. Pastikan kabel paralel terhubung antara PC dan **XS Board**. Gunakan LPT1 dengan mode SPP.
8. Ketik (misalnya desain disimpan dengan nama **"circuit"**) :
   `C:\> XSLOAD CIRCUIT.BIT`

`C:\> XSPORT 10101` bit-bit MSB bila tidak ditulis, dianggap '0'

11. Pengujian dapat dilakukan berkali-kali dengan mengulang langkah 7 dan mengamati respons 7-segment.
12. Setelah selesai, ketik "`exit`" <**enter**> untuk kembali ke **Window**

BAB 3

KONTROLER FPGA-FUZZY

Hardware yang dibuat adalah **Kontroler FPGA-Fuzzy (FF)** yang diimplementasikan pada IC FPGA XC4005XL dari Xilinx. Untuk keperluan demo, IC ini dioperasikan di suatu *prototype-board*, XS-40 Board (XS-Board) dan *expansion-board* Xstend.

3.1.1. Spesifikasi Kontroler FF :

- 8 bit - 4 *Input* dan 4 *Output* paralel
- *Range* nilai *fuzzy* antara 0 sampai 63 (resolusi 6 bit)
- 6 tipe *Membership Function*
- *Floating Membership Function* dan *Floating Action Value*
- 56 x 8 bit konstanta yang dapat digunakan sebagai nilai *center, width* atau *action*
- 256 x 6 bit memori untuk menyimpan *rule*
- Kemampuan penyimpanan internal sampai 63 variabel *fuzzy* dan 40 *rule fuzzy* dalam 8 blok *output*
- Parameter dan *rule fuzzy* dapat dimasukkan sebelum atau setelah sintesis desain

Instruksi Kontroler FF yang berupa parameter dari *rule fuzzy* dapat dimasukkan ke dalam kontroler sebelum atau sesudah desain disintesis dan di-*download*-kan ke FPGA. Pemasukan instruksi sebelum sintesis berupa *file* .mem yang dimasukkan ke komponen-komponen RAM dalam desain, sedang pemasukan instruksi sesudah desain berupa *file* .hex yang di-*download* ke RAM eksternal dan kemudian disalin ke RAM internal kontroler FF. *File-file* ini dihasilkan oleh program **FIDE** yang merupakan bagian dari Tugas Akhir ini. Penjelasan lebih lanjut tentang **FIDE** dapat diperoleh di **BAB 4 FIDE**.

Kontroler FF dapat menggunakan 6 tipe *Membership Function* yang mempunyai *slope* konstan dan hanya memerlukan nilai *input, center* dan *width*. Untuk *input* dapat pula digunakan *loopback* dari *output*, sehingga terdapat 8 sumber nilai *input*. Untuk *center* dan *width*, selain menggunakan nilai konstan dapat pula menggunakan nilai *input* atau *output* (*floating membership function*).

Metode Defuzzifikasi dapat berupa *Immediate*, nilai *output* sama dengan nilai *action value* atau *Accumulate*, nilai *output* yang baru sama dengan nilai *output* yang lama ditambah / dikurangi nilai *action value*. Action value dapat berupa konstanta atau nilai *input* dan *output* (*floating action value*).

5	OSU_A2	S2 R7-S XS-T / A2 RAM XS-B	47	IPD<3>	D3 Parallel Port
6			48	IPD<4>	D4 Parallel Port
7	EC	DIP-Sw 1 XS-T	49	IPD<5>	D5 Parallel Port
8	CS	DIP-Sw 2 XS-T	50	OSL4_A12	S4 L7-S XS-T / A12 RAM
9	CCS	DIP-Sw 3 XS-T	51	OSL2_A10	S2 L7-S XS-T / A10 RAM
10	OP_RD<7>	LED 8 XS-T / D7 RAM XS-B	52	GND	GND
11	VCC	VCC	53		
12	GND	GND	54	VCC	VCC
13	CLK	Oscillator 12 MHz di XS-B	55		
14			56	OSL3_A11	S3 L7-S XS-T / A11 RAM
15			57	OSL1_A9	S1 L7-S XS-T / A9 RAM
16			58	OSL5_A13	S5 L7-S XS-T / A13 RAM
17			59	OSL0_A8	S0 L7-S XS-T / A8 RAM
18	CAS5	S5 7-S XS-B	60	OSL6_A14	S6 L7-S XS-T / A14 RAM
19	CAS6	S6 7-S XS-B	61	OE_RAM	OE RAM XS-B
20	CAS3	S3 7-S XS-B	62	WE_RAM	WE RAM XS-B
21	GND	GND	63	VCC	VCC
22	VCC	VCC	64	GND	GND
23	CAS4	S4 7-S XS-B	65	CE_RAM	CE RAM XS-B
24	CAS2	S2 7-S XS-B	66	PGM	DIP-Sw 7 XS-T / PC_
25	CAS0	S0 7-S XS-B	67		
26	CAS1	S1 7-S XS-B	68		
27			69	UPDT_IN	DIP-Sw 8 XS-T / PC_
28			70		
29			71		
30			72		
31	GND	GND	73		
32	IPD<6>	D6 Parallel Port	74	VCC	VCC
33	VCC	VCC	75		
34	IPD<7>	D7 Parallel Port	76	GND	GND
35	OP_RD<4>	LED 5 XS-T / D4 RAM XS-B	77		
36	RST_UC	RST Micro Controller 8051	78	OSU_A3	S3 R7-S XS-T / A3 RAM
37	RST	Push Button RESET XS-T	79	OSU_A4	S4 R7-S XS-T / A4 RAM
38	OP_RD<3>	LED 4 XS-T / D3 RAM XS-B	80	OP_RD<6>	LED 7 XS-T / D6 RAM X
39	OP_RD<2>	LED 3 XS-T / D2 RAM XS-B	81	OP_RD<5>	LED 6 XS-T / D5 RAM X
40	OP_RD<1>	LED 2 XS-T / D1 RAM XS-B	82	OSU_A5	S5 R7-S XS-T / A5 RAM
41	OP_RD<0>	LED 1 XS-T / D0 RAM XS-B	83	OSU_A6	S6 R7-S XS-T / A6 RAM
42	VCC	VCC	84	A7	A7 RAM XS-B

hubungan antara FPGA dengan komponen komponen lain pada **XS-Board** dan **XStend**.

3.1.4.1. Pin-Pin Input

CCS *flag Counter Clock Select* dihubungkan dengan DIP-Sw 3 pada **XStend** :

- 0 (ON) : *clk counter* = 1/8 **CLK**,
- 1 (OFF) : *clk counter* = 1/16 **CLK**.

CLK *Clock*, jalur masuk sumber *clock* eksternal, dihubungkan dengan osilator 12 MHz pada **XS-Board**.

CS *flag (Master) Clock Select* dihubungkan dengan DIP-Sw 2 pada **XStend** :

- 0 (ON) : *clk* utama = 1/2 **CLK**,
- 1 (OFF) : *clk* utama = 1/4 **CLK**.

EC *flag External Clock.* Dihubungkan dengan DIP-Sw 1 pada **XStend**. Digunakan untuk memilih *clock* eksternal (OFF) / internal (ON). Untuk demo, *clock* eksternal dihubungkan dengan osilator 12 MHz pada **XS-Board**. Clock internal dihubungkan dengan *clock* 15 Hz yang berada dalam IC FPGA.

IPD [7:0] *Input Pad*, merupakan jalur masuknya data *input* baru pada fase operasi kontroler. Dihubungkan dengan D[7:0] PC Parallel Port.

UPDT_IN *Update Input*. Dihubungkan dengan DIP-Sw 8 pada **XStend.** Bila pin ini diset HI (OFF), maka nilai pada **IPD** akan digunakan untuk meng-*update* nilai *input* pada *register* internal kontroler. Pin ini juga diperuntukkan hanya untuk kepentingan demo untuk memungkinkan pengecekan kebenaran hasil proses kontroler terhadap suatu nilai *input* tertentu.

3.1.4.2. Pin-Pin Output

A7 RAM Address A7, merupakan alamat RAM di **XS-Board**.

CAS [6:0] Counter Address 7 Segment, menampilkan *lower word* alamat *I/O Counter* pada 7-Segment di **XS-Board**. Hanya digunakan untuk kepentingan demo. Pada aplikasi sebenarnya, yang dikeluarkan adalah *address counter* itu sendiri yang digunakan untuk memilih *input* atau *output* berdasarkan slot waktunya.

CE_RAM *Chip-Enable RAM*. Pada fase program, **CE_RAM** diset LO secara internal untuk memungkinkan pembacaan RAM oleh kontroler.

OE_RAM *Output-Enable RAM*. Pada fase program, **OE_RAM** diset LO secara internal untuk memungkinkan pembacaan RAM oleh kontroler.

operasi, dimultipleks dengan alamat RAM di **XS-Board** yang bermanfaat pada fase program.

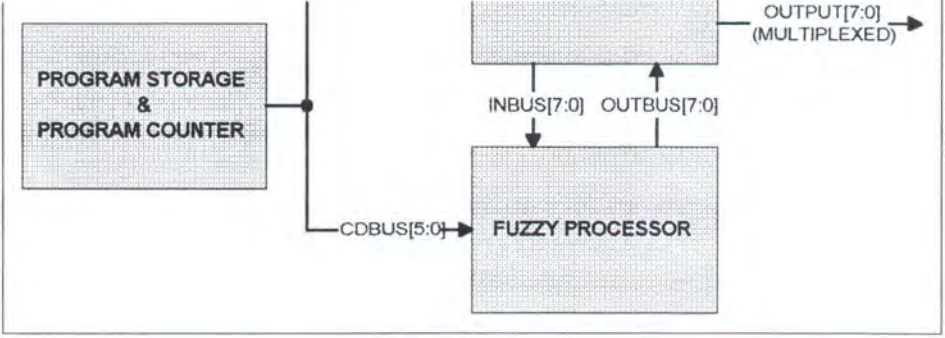
OSU_A [6:0] Upper Word Output 7 Segment / RAM Address [A6..A0], menampilkan *upper word* nilai *output* pada 7-Segment di **XStend** pada fase operasi, dimultipleks dengan alamat RAM di **XS-Board** yang bermanfaat pada fase program.

RST_UC *Reset Micro Controller*. Digunakan hanya untuk keperluan demo dengan **XS-Board**. **RST_UC** selalu diset HI secara internal untuk mempertahankan *micro controller* pada kondisi *reset*-nya supaya tidak mengganggu jalannya komponen yang lain.

WE_RAM *Write-Enable RAM*. **WE_RAM** selalu diset HI secara internal untuk mencegah perubahan isi RAM pada **XS-Board**.

3.2. Cara Kerja Kontroler FF

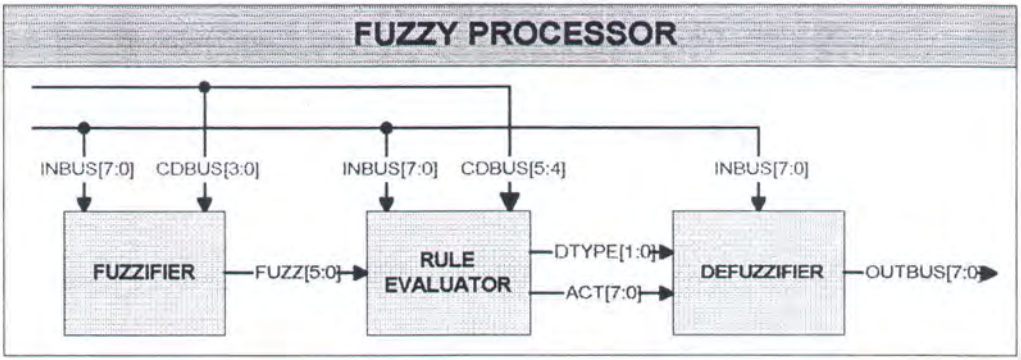
Pemrosesan data terdiri dari beberapa tahap. Pertama data di-*sampling* dan disimpan dalam *input register* pada bagian **I/O & Fix Parameter Storage** melalui *bus* **INPUT[7:0]** yang merupakan jalur termultipleks dalam waktu, sesuai



Gambar 3. 1.

Diagram Blok **Kontroler FPGA-Fuzzy** yang Disederhanakan

Perhitungan kemudian dilakukan pada bagian **Fuzzy Processor** yang terdiri dari **Fuzzifier, Rule Evaluator** dan **Defuzzifier**.



Gambar 3. 2.

Diagram Blok **Fuzzy Processor**

Fuzzifier membandingkan nilai *input* tersebut dengan variabel-variabel *fuzzy* sesuai *membership function*–nya untuk menentukan nilai *fuzzy* suatu *term* **FUZZ[5:0]**. Nilai-nilai *fuzzy* tersebut lalu diproses lebih lanjut oleh bagian **Rule**

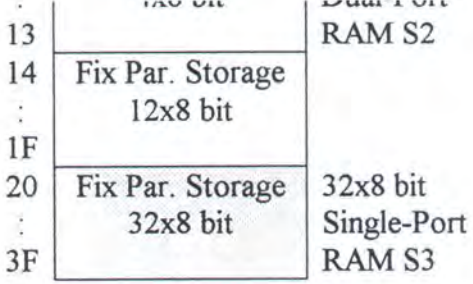
perhitungan nilai *output*.

Nilai *output* ini kemudian disimpan di ***I/O & Fix Parameter Storage*** melalui **OUTBUS[7:0]** untuk kemudian dikeluarkan secara berkala ke pin *output* atau digunakan untuk *loopback* internal atau sebagai variabel *fuzzy*. Proses pengeluaran nilai *output* melalui **OUTPUT[7:0]** sebagaimana proses pemasukan nilai *input* juga berlangsung terpisah dari proses internal kontroler *fuzzy* dan termultipleks dalam empat slot waktu. Slot waktu yang digunakan sama dengan nilai *input*, misalnya, proses *sampling input* pertama adalah bersamaan dengan proses pengeluaran nilai *output* pertama.

3.2.1. I/O & Fix Parameter Storage

Bagian ini merupakan tempat penyimpanan konstanta yang digunakan untuk *term* dan *rule* dan sekaligus sebagai *I/O Port*, untuk penyimpanan nilai *input* dan *output*.

Nilai *input* dan *output* disimpan dalam RAM dan diperlakukan seperti isi *memory* biasa dengan perkecualian adanya *update* otomatis untuk *input* dan pengeluaran nilai *output* ke pin-pin secara periodik. Proses *update input* dan pengeluaran nilai *output* ini dilakukan secara paralel, terpisah dari proses perhitungan nilai *fuzzy* pada **Fuzzy Processor**.



Gambar 3. 3.

Peta Memori **I/O & Fix Parameter Storage**

Secara keseluruhan disediakan RAM 64 x 8 bit, masing-masing 4 *byte* untuk menyimpan sementara nilai *input* dan *output*, dan 56 *byte* untuk menyimpan konstanta-konstanta yang digunakan sebagai parameter. Ke-64 *byte* nilai tersebut dapat digunakan sebagai nilai *center, width* dan *action value*, sedang *input* dapat diambil dari 8 *byte* nilai *input / output* (*loopback*). Penggunaan penyimpan yang digabung menjadi satu ini memungkinkan adanya fasilitas *floating* pada nilai *center, width* dan *action value*.

3.2.2. Program Storage

Program Storage terdiri dari RAM 256 x 6 bit yang digunakan untuk menyimpan *rule-rule fuzzy*.

- **TE, Term Evaluation**, dengan kode 2 – 7 yang menentukan tipe *membership function* yang digunakan untuk mengevaluasi nilai *input* :
  - 2 : Symmetrical Inclusive M F,
  - 3 : Symmetrical Exclusive M F,
  - 4 : Left Inclusive M F,
  - 5 : Left Exclusive M F,
  - 6 : Right Inclusive M F,
  - 7 : Right Exclusive M F.

  Kode ini diikuti 3 *word* data alamat nilai *input*, *center* dan *width* pada **I/O & Fix Parameter Storage** yang digunakan untuk mengevaluasi *term* tersebut.

5	4	3	2	1	0
Not Used		MF Type			
Input Value Address					
Center Value Address					
Width Value Address					

Gambar 3. 5.

Format Kode untuk **TE**

- **EOR, End Of Rule**, kode akhir suatu *rule*. Bit 3:0 bernilai 1, sedangkan bit 5:4 menentukan tipe defuzzifikasi *rule* tersebut :
  - 0 : IMM, Immediate, *output = action value*,

MILIK PERPUSTAKAAN ITS

Type					
Action Value Address					

Gambar 3. 6.

Format Kode untuk **EOR**

- **EORRSO, End Of Rule(s) Reffering Specific Output**, kode akhir kumpulan *rule-rule* yang mengacu pada *output* yang sama, dengan kode 0. Kode ini diikuti 1 *word* data alamat nilai *output* yang akan di-*update*.

5	4	3	2	1	0
Not Used		0	0	0	0
Output Value Address					

Gambar 3. 7.

Format Kode untuk **EORRSO**

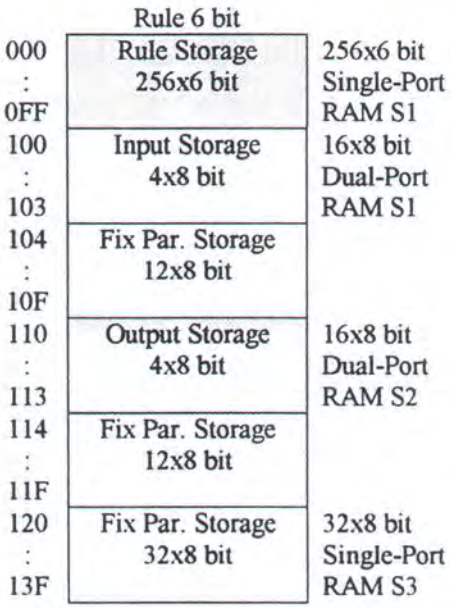
- **EORs, End Of Rules**, kode akhir *rule* dengan nilai 8. Kode ini digunakan untuk menandai akhir dari *rule* supaya pembacaan *rule* dimulai dari awal kembali. Kode ini hanya digunakan bila *rule storage* tidak terisi penuh. Bila terisi penuh, secara otomatis pembacaan *rule* akan dimulai dari awal kembali.

5	4	3	2	1	0
Not Used		1	0	0	0

Gambar 3. 8.

Format Kode untuk **EORs**

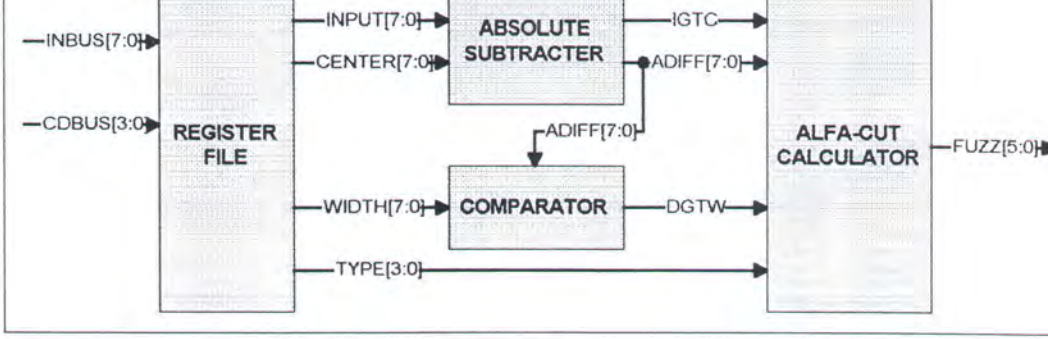
power-on. Pengisian dilakukan dengan menentukan isi suatu komponen RAM pada desain, lalu mensintesis desain tersebut. Isi RAM tersebut akan menjadi bagian dari *file* konfigurasi yang kemudian di-*download* ke FPGA.



Gambar 3. 9.

Peta Memori pada File .hex yang Di-*download*-kan pada RAM Eksternal

Selain itu juga dimungkinkan dengan men-*download* nilai dari RAM eksternal. Hal ini dilakukan dengan memberikan nilai HI pada pin **PGM**, lalu memberikan pulsa pada pin **RST** yang akan membawa kontroler ke fase penulisan RAM internalnya. Para fase ini, pertama akan ditulisi RAM pada **Program Storage** kemudian pada **I/O & Fix Parameter Storage**.



Gambar 3. 10.

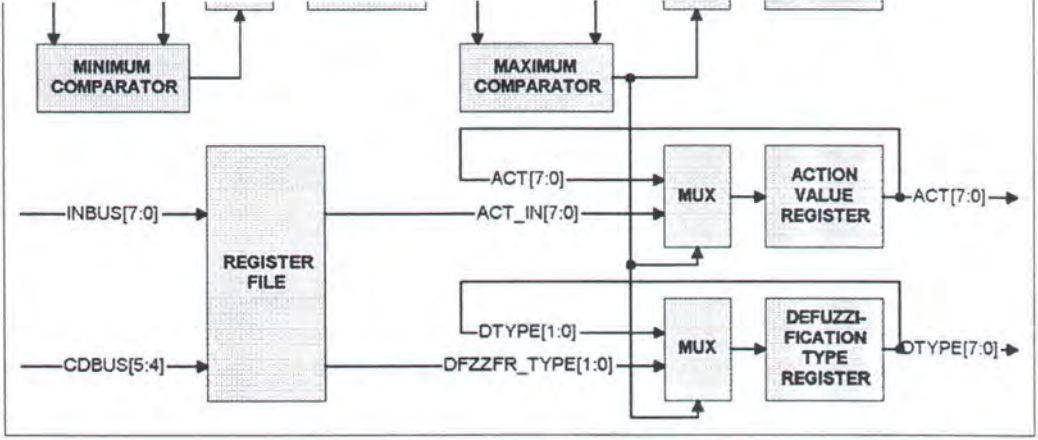
Diagram Blok Modul **Fuzzifier**

Pertama-tama, nilai *input*, **INPUT[7:0]** dibandingkan dengan nilai *center*, **CENTER[7:0]** lalu selisihnya, **ADIFF[7:0]** dibandingkan dengan *width*, **WIDTH[7:0]**. Hasil-hasil perhitungan tersebut akan diproses oleh **Alfa-Cut Calculator** berdasarkan tipe *membership function term*, **TYPE[3:0]**, *flag* **IGTC** (*Input Greater Than Center*) dan *flag* **DGTW** (*Difference Greater Than Width*) untuk menentukan nilai *fuzzy*-nya, **FUZZ[5:0]**.

3.2.5. Rule Evaluator

Rule Evaluator berfungsi untuk menentukan nilai *action value* dan tipe operasi defuzzifikasi yang akan digunakan.

Secara umum, terdapat dua proses pada bagian ini. Pada bagian pertama, nilai *fuzzy* dari *term-term* suatu *rule* yang dihasilkan dari **Fuzzifier** dibandingkan.



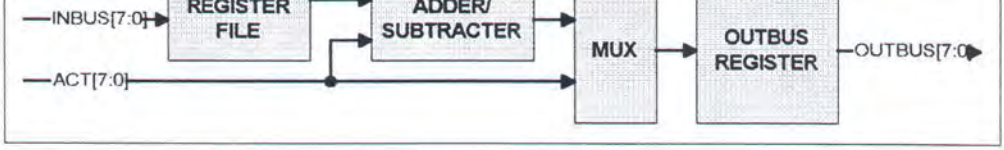
Gambar 3. 11.
Diagram Blok Modul **Rule Evaluator**

Kemudian, nilai *fuzzy rule-rule* yang mengacu pada suatu *output* tertentu akan dibandingkan. Rule dengan nilai *fuzzy* terbesar akan menjadi *rule* pemenang, **MAX_FUZZ[5:0]**. Nilai *action value* dan tipe defuzzifikasinya akan digunakan untuk meng-*update output* yang diacunya, **ACT[7:0]** dan **DTYPE[1:0]**.

3.2.6. Defuzzifier

Defuzzifier akan meng-*update* nilai suatu *output* berdasarkan *action value* dan tipe operasi defuzzifikasi dari *rule* pemenang, **ACT[7:0]** dan **DTYPE[1:0]**.

Terdapat dua macam tipe operasi defuzzifikasi, *accumulate* dan *immediate*. Pada tipe operasi *accumulate*, *action value* akan ditambahkan atau



Gambar 3. 12.

Diagram Blok Modul **Defuzzifier**

Nilai *output* yang baru pada **OUTBUS[7:0]** dituliskan ke **I/O & Fix Parameter Storage** untuk kemudian dikeluarkan secara periodik atau digunakan sebagai variabel *fuzzy* (*floating membership-function & floating action-value*).

3.3. Arsitektur Kontroler FF

Terdapat dua proses yang berjalan secara terpisah pada kontroler, proses *sampling input* dan pengeluaran nilai *output* dan proses operasi internal kontroler. Pemisahan kedua proses ini dimaksudkan supaya kedua proses tersebut dapat menggunakan *clock* yang berbeda. Proses *I/O* menggunakan *counter clock* **BCNT_CLK** sedang proses internal menggunakan *clock* utama **BCLK**. Pemisahan ini memungkinkan **Fuzzy Processor** beroperasi pada frekuensi tinggi meskipun proses *I/O*-nya lambat.

3.3.1. Rangkaian Clock

Terdapat 2 sumber *clock* yang dapat digunakan :

Penjelasan yang lebih lengkap dapat diperoleh pada *sheet* **2** dari *sheet* utama.

3.3.2. Rangkaian Seven-Segment

Untuk keperluan demo digunakan 3 buah *7-Segment* :

- 1 buah 7-Segment *common cathode (active high)* pada XS-Board untuk menampilkan *lower word* dari *I/O counter address*,
- 2 buah 7-Segment *common anode (active low)* di XStend untuk menampilkan nilai *output* sesuai nomor slot waktu pada *I/O counter address*.

Dekoder *7-Segment* diwujudkan dengan menggunakan ABEL HDL, metode *truth table* yang dapat dilihat pada *listing module* **F_7SDCD.ABL** untuk dekoder *active high* dan *listing module* **F_N7SDCD.ABL** untuk dekoder *active low*.

3.3.3. I/O & Fix Parameter Storage

I/O & Fix Parameter Storage terdiri dari dua buah *dual-port, edge-triggered* 16 x 8 bit RAM, **S1** dan **S2**, satu *single-port, edge-triggered* 32 x 8 bit RAM **S3** serta *counter* 6 bit, F_CNT6 **C1** yang digunakan untuk mengalamati **S1**, **S2** dan **S3**. Realisasi rangkaian dapat dilihat pada *sheet* **F_IOREG** pada

sebagai *fix parameter register* (lihat Gambar 3.3. Peta Memori **I/O & Fix Parameter Storage**). Karena direncanakan bahwa *sampling input* dan pengeluaran nilai *output* berjalan secara paralel dengan operasi internal kontroler, maka untuk **S1** dan **S2** digunakan RAM tipe *dual-port* yang memungkinkan penulisan dan pembacaan isi RAM secara bersamaan.

RAM **S1** ditulisi pada proses program dan pemasukan nilai *input* dan diperlukan untuk dibaca pada proses pembacaan parameter. Karenanya :

- **A[3:0]** dihubungkan dengan **CNT_ADDR[5:0]** yang menyediakan alamat penulisan RAM pada proses program dan *input/output*,
- **DI[7:0]** dihubungkan dengan **INPAD[7:0]** saat fase operasi dan dengan **RAMDATA[7:0]** saat fase program melalui multiplekser **M3**,
- **SPO[7:0]** tidak dihubungkan karena tidak diperlukan pembacaan pada proses program dan *input/output*,
- **WE** harus bernilai HI saat pemrograman 16 *byte* terbawah dari **I/O & Fix Parameter Storage** dan saat operasi bila **UPDT_IN** diset HI, karenanya digunakan sistem dekoder seperti pada *sheet*,
- **DPRA[3:0]** dihubungkan dengan **D_ADDR[3:0]** yang akan dihasilkan bagian **Fuzzifier** untuk menyediakan alamat RAM yang akan digunakan sebagai parameter dalam proses *fuzzification* dan *defuzzification*,

RAM **S2** ditulisi pada proses program dan proses penulisan nilai *output* dan dibaca untuk pengeluaran nilai *output*. Karenanya :

- **A[3:0]** dihubungkan dengan **D_ADDR[3:0]** saat penulisan parameter dan dengan **CNT_ADDR[5:0]** saat pemrograman. Hal ini direalisasikan dengan menggunakan multiplekser **M5**,
- demikian pula **D[7:0]** dihubungkan dengan **OUTBUS[7:0]** saat penulisan parameter dan dengan **RAMDATA[7:0]** saat pemrograman dengan menggunakan multiplekser **M4**,
- **SPO[7:0]** dihubungkan dengan **INBUS[7:0]** melalui **M1** dan **M2**,
- **WE** harus *high* saat penulisan parameter dan pemrograman 16 *byte* kedua **I/O & Fix Parameter Storage** atau saat penulisan nilai *output* yang baru, karenanya digunakan sistem dekoder seperti pada *sheet*,
- **DPRA[3:0]** dihubungkan dengan **CNT_ADDR[3:0]** untuk menghasilkan alamat bagi pengeluaran nilai *output*.
- **DPO[7:0]** dihubungkan dengan **OUTPAD[7:0]** untuk mengeluarkan nilai *output*.

RAM **S3** hanya ditulisi saat pemrograman awal dan selanjutnya hanya dapat dibaca untuk pembacaan nilai parameter. Karenanya :

F_CN16 **C1** bermanfaat untuk mengalamati **S1**, **S2** dan **S3** saat fase program dan mengalamati *input* dan *output register* saat fase operasi untuk melakukan *update* nilai *input* dan pengeluaran nilai *output*. Sinyal **TERM_CNT** di-AND-kan dengan **PGM** untuk menandai akhir proses pengisian **I/O & Fix Parameter Storage**. Pada fase operasi, **C1** dioperasikan sebagai *counter* 2 bit, sesuai jumlah *input* dan *output* yang masing-masing sebanyak 4 buah karenanya digunakan rangkaian *reset counter* seperti pada *sheet*.

Rangkaian multiplekser **M1** dan **M2** akan memilih jalur yang akan dihubungkan ke **INBUS[7:0]** berdasarkan urutan **S1** menempati alamat 16 *byte* terbawah dari **I/O & Fix Parameter Storage**, kemudian di atasnya, **S2**, dan terakhir **S3** sesuai dengan Peta Memori pada Gambar 3.3. Komponen **FMAP** yang ada pada *sheet* digunakan untuk memetakan fungsi yang terdiri dari banyak *gate* ke satu *function-generator* pada CLB FPGA.

3.3.4. Program Storage

Realisasi rangkaian secara detil dapat dilihat pada *sheet* **F_PGMSTO** pada **LAMPIRAN**.

Bagian ini terdiri dari komponen **S1**, RAM 6x256 yang digunakan untuk menyimpan kode-kode program dan **C1**, *counter* 8 bit yang digunakan sebagai

dihubungkan dengan **PGMT** yang dihasilkan oleh kontrol logika. Data yang akan dituliskan berasal dari RAM eksternal, sehingga jalur datanya dihubungkan dengan pin **OP_RD[5:0]**, sedangkan *output*-nya dihubungkan dengan jalur *program bus* **CODE[5:0]**.

C1 berupa *counter* 8 bit yang dioptimasikan menggunakan rangkaian *carry logic* XC4000XL. Pin **CE** (*chip enable*) dan **R** (*reset*) diatur kontroler logika. Untuk **PC** ini digunakan transisi positif *clock* supaya sinkron dengan kontroler logika dan terdapat cukup waktu bagi alamat yang baru untuk stabil sebelum data dibaca dari **S1**. Keluaran **C1** selain digunakan untuk **S1** juga dikeluarkan untuk kemudian dimultipleks melalui multiplekser **M4** (pada *sheet* utama) dengan jalur keluaran *counter* pada **I/O & Fix Parameter Storage** dan digunakan untuk mengalamati RAM eksternal pada fase program. Pulsa yang dihasilkan **TERM_CNT** yang menandai tercapainya limit perhitungan *counter* 8 bit (255) di-AND dengan **PGM** menghasilkan sinyal **ENDPGM** yang menandakan selesainya penulisan seluruh isi **S1**.

3.3.5. Kontroler Logika

Skema Kontroler Logika dapat dilihat pada *sheet* **F_CNTRLR** pada **LAMPIRAN**. Bagian utama Kontroler Logika ini adalah komponen

F_FSMCON.ABL pada LAMPIRAN.

Untuk mengatur jalannya **Kontroler FF**, digunakan suatu *finite state machine* dengan 14 *state*. Secara umum, terdapat 6 macam proses yang terjadi pada **Kontroler FF** ini :

- Tahap Inisialisasi, ***Initialization Stage***,
- Tahap Pemrograman, ***Programming Stage***,
- Tahap pembacaan kode operasi, ***Fetching Code Stage***,
- Tahap evaluasi *term*, **TE *stage***,
- Tahap akhir dari *rule*, **EOR *stage***, dan
- Tahap akhir dari *rule-rule* yang mereferensikan suatu *output* tertentu, **EORRSO *stage***.

3.3.5.1. Initialization Stage

Tahap ini dimasuki bila sinyal **RST** diaktifkan. Pada tahap ini diaktifkan sinyal **Hold_PC** untuk menon-aktifkan **PC** supaya pembacaan kode pada tahap berikutnya dimulai pada alamat 0. Juga ditentukan tahap berikutnya, apakah memasuki fase pemrograman atau fase operasi. Bila **PGM** diaktifkan sebelum **RST** diaktifkan, **Kontroler FF** akan memasuki fase pemrograman, bila **PGM**

Tahap pemrograman ini terdiri dari dua bagian, pemrograman **Program Storage** dan pemrograman **I/O & Fix Parameter Storage**.

- Pemrograman **Program Storage**

  Pada tahap ini terjadi pengisian **Program Storage** dengan mengaktifkan sinyal **PGM1** yang memungkinkan penulisan pada **S1 F_PGMSTO**. Tahap ini diakhiri dengan aktifnya sinyal **EndPgm1** dan dilanjutkan dengan pengisian **I/O & Fix Parameter Storage**.

- Pemrograman **I/O & Fix Parameter Storage**

  Pada tahap ini terjadi inisialisasi *input* dan *output register* dan pengisian *fix value registers*. Seperti tahap sebelumnya, pengisian **I/O & Fix Parameter Storage** dilaksanakan dengan aktivasi sinyal **PGM2** yang merupakan sinyal *write-enable registers* dan diakhiri dengan sinyal **EndPgm2**. Tahap ini dilanjutkan dengan tahap pembacaan kode operasi. Sinyal **PGM2** juga menyebabkan di-*reset*-nya **PC** untuk menjamin pembacaan kode operasi dimulai dari alamat 0.

3.3.5.3. Fetching Code Stage

Tahap ini merupakan tahap penentuan arah operasi. Untuk tipe operasi **TE**, dilanjutkan ke **TE Stage**, **EOR** ke **EOR Stage**, dan **EORRSO** ke **EORRSO**

Tahap ini terdiri dari 3 bagian :

- **St_Ld_In**, pada tahap ini, kode pada **Program Storage** berupa alamat *input* yang akan digunakan dan jalur data, **INBUS[7:0]** akan berisi nilai *input* yang dialamati. Dengan aktivasi **Ld_In**, maka nilai tersebut akan dimasukkan pada **DFF Input** pada **Fuzzifier**.
- **St_Ld_Cnt**, pada tahap ini, kode pada **Program Storage** berupa alamat nilai *center* yang akan digunakan, baik yang *fix* maupun *floating* (dari nilai *input / output*). Seperti sebelumnya, alamat ini akan menyebabkan jalur data berisi nilai *center* yang diinginkan dan aktivasi **Ld_Cnt** akan menyebabkan tersimpannya nilai tersebut pada **DFF Center** pada **Fuzzifier**. Masuknya nilai *center* ini menyebabkan **F_ASUB8 A1** menghasilkan selisih absolut antara *center* dan *input*, **ADIFF[7:0]**.
- **St_Ld_Wdt**, pada tahap ini kode pada **Program Storage** berupa alamat nilai *width* yang digunakan yang kemudian diproses seperti nilai-nilai sebelumnya. Tetapi, di sini, sinyal **Ld_Wdt** juga sekaligus dihubungkan dengan sinyal **En_Asub8** pada **Fuzzifier** yang menyebabkan 'keluar'-nya hasil **F_ASUB8 A1**, sehingga tersedia dua operan untuk **F_CMPGT8 CM1**. Kemudian hasil-hasil **CM1** bersama-sama dengan hasil **A1** akan digunakan oleh **F_ALFCUT DCD1** untuk menghitung nilai *fuzzy term* tersebut, **FUZZ[5:0]**.

pertama, maka nilai ini akan langsung diteruskan ke **F_MINDFF D1**.

- **St_En_Min**, pada tahap ini kode dari **Program Storage** juga tidak digunakan. **F_MINDFF D1** akan memberikan nilainya, **MIN_FUZZ[5:0]** ke **F_CMPGT6 CMP2** untuk di-*max-compare*-kan dengan nilai sebelumnya. Bila nilai ini merupakan nilai *fuzzy* pertama dari *rule* pertama, maka nilai ini akan langsung diteruskan ke **F_MAXDFF D4**. Tahap ini merupakan bagian terakhir tahap evaluasi *term* dan dilanjutkan dengan kembali pada tahap pembacaan kode.

3.3.5.5. EOR Stage

Tahap ini terdiri dari 2 bagian :

- **St_Ld_Act**, kode **Program Storage** berupa alamat *action value* yang akan digunakan untuk *rule* yang bersangkutan. Aktivasi **Ld_Act** menyebabkan masuknya nilai tersebut ke **D2**. Sebelumnya, sinyal **Ld_Type** juga menyebabkan masuknya tipe defuzzifikasi untuk *rule* yang sedang diproses ke **D3**. Nilai *action* akan diteruskan ke **D5** dan tipe defuzzifikasi akan diteruskan ke **D6** bila nilai *fuzzy rule* yang diproses ini lebih besar dari nilai *fuzzy rule* sebelumnya, atau bila *rule* ini merupakan *rule* pertama, nilai *action* dan tipe defuzzifikasi akan langsung diteruskan.

F_MINDFF D1. Tahap ini merupakan bagian akhir tahap akhir *rule* sehingga dilanjutkan kembali dengan tahap pembacaan kode.

3.3.5.6. EORRSO Stage

Tahap ini terdiri dari 3 bagian :

- **St_Ld_Out**, kode **Program Storage** berupa alamat nilai *output* yang akan dioperasikan. Aktivasi sinyal **Ld_Out** akan menyebabkan masuknya nilai *output* dari **INBUS[7:0]** ke **D1** yang akan menyediakan operan kedua bagi **F_ADSU8 M1** (operan pertama adalah **ACT[7:0]**). Operasi **M1**, penjumlahan atau pengurangan ditentukan oleh sinyal **ADD_SUB** yang berasal dari MSB kode operasi.
- **St_En_Dfzzfr**, kode **Program Storage** tidak digunakan. Nilai *output* yang baru dimasukkan ke **D2** dengan aktivasi sinyal **En_Dfzzfr**. Nilai *output* yang baru ini dapat berupa hasil operasi **M1** atau nilai *action value rule* pemenang, tergantung sinyal **ACC_IMM** yang berasal dari MSB-1 kode operasi.
- **St_Wr_Out**, *updating* nilai *output* pada *register* penyimpan nilai *output*. Terjadi *resetting* **F_MAXDFF R3** dan dilanjutkan dengan kembali ke tahap pembacaan kode operasi.

center, *width* dan tipe operasi, **D1**, **D2**, **D3** dan **D4**.

- Absolute Subtracter 8 bit ter-*register*, **F_ASUB8 A1** yang berfungsi untuk menghitung harga mutlak selisih nilai *input* dan *center* dan menentukan apakah *input* lebih besar dari *center*. Penjelasan lebih rinci dari bagian ini dapat dilihat pada *sheet* **F_ASUB8** dalam **LAMPIRAN**.

  Pertama-tama ditentukan apakan nilai *input* lebih besar dari *center* atau sebaliknya di **F_CMPGT8 CM1** dan menghasilkan *flag* **IGTC**. Flag ini menentukan apakah nilai *center* dikurangkan dari nilai *input* atau sebaliknya. Pemilihan operan ini terjadi di multiplekser **M1** dan **M2** dan operasi pengurangan dilakukan di *registered-subtracter* 8 bit **A1**. Pemilihan operan ini diperlukan untuk menghasilkan selisih absolut antara *input* dan *center*.
- Comparator 8 bit, **F_CMPGT8 CM1** yang berfungsi untuk menentukan apakah selisih *input* dan *center* lebih besar dari *width*, *flag* **DGTW**.
- Alfa Cut Calculator, **F_ALFCUT DCD1** yang berfungsi untuk menentukan nilai *fuzzy* / bobot suatu *term*, **FUZZ[5:0]** berdasarkan parameter-parameter :
  - apakah *input* lebih besar dari *center*, **IGTC**,
  - apakah selisih lebih besar dari *width*, **DGTW**,
  - tipe membership function, **TYPE[3:0]**,

berdasarkan parameter-parameter di atas, nilai *fuzzy* suatu *term*, apakah minimum (0), selisih, komplemen dari selisih atau maksimum (63) pada **F_MUX4x6 M1**. Pada bagian akhir ditambahkan **F_DFF6 D1** untuk sinkronisasi sinyal.

3.3.7. Rule Evaluator

Skema **Rule Evaluator** dapat dilihat pada *sheet* **F_EVLTR** pada **LAMPIRAN**. Modul ini terdiri dari dua bagian, yaitu :

- minimum calculator yang berfungsi untuk menentukan nilai *fuzzy* suatu *rule* yang berasal dari nilai *fuzzy* terkecil dari nilai *fuzzy term-term* penyusun *rule* tersebut, **MIN_FUZZ[5:0]** dan
- maximum calculator yang menentukan *rule* pemenang, yaitu *rule* dengan nilai *fuzzy* terbesar dan sekaligus menyediakan nilai *action* dan tipe *defuzzifikasi* untuk *defuzzifier*, **MAX_FUZZ[5:0]**, **ACT[7:0]**, dan **DTYPE[1:0]**.

3.3.7.1. Minimum Calculator

Bagian ini diimplementasikan dengan :

menjadi nilai *fuzzy rule* yang sedang diproses, **MIN_FUZZ[5:0]**. Register ini dapat di-*set* untuk menyimpan nilai maksimum (63) saat terjadi pergantian *rule* yang dievaluasi untuk menjamin masuknya nilai *fuzzy term* pertama suatu *rule*.

3.3.7.2. Maximum Calculator

Bagian ini diimplementasikan dengan :

- Comparator 6 bit, **CMP2** untuk membandingkan nilai *fuzzy rule* yang baru dengan yang tersimpan dalam DFF maximum **D4**,
- Multiplekser 2x6 bit, **M2** untuk memilih nilai *fuzzy rule* yang baru,
- DFF **D4** untuk menyimpan nilai maksimum sementara yang kemudian akan menjadi nilai *fuzzy* pemenang **MAX_FUZZ[5:0]**, **D4** di-*reset* saat terjadi pergantian *output* yang diproses untuk menjamin masuknya nilai *fuzzy rule* pertama dari rangkaian *rule* yang mengacu pada suatu nilai *output*.
- DFF **D2** dan **D3** yang digunakan untuk menyimpan *action value* dan tipe operasi dari *rule* yang sedang diproses,
- Multiplekser **M3** dan **M4** yang digunakan untuk menentukan *action value* dan tipe defuzzifikasi yang akan diteruskan ke **D5** dan **D6** berdasarkan hasil

3.3.8. Defuzzifier

Defuzzifier terdiri dari dua buah DFF, **D1** dan **D2**, sebuah *adder / subtracter 8 bit*, **A1** dan sebuah multiplekser, **M1**. Penjelasan lebih lengkap dapat diperoleh pada *sheet* **F_DFZZFR** pada **LAMPIRAN**.

Mula-mula nilai *output* yang akan di-*update* disimpan di **D1**. Kemudian nilai ini dioperasikan dengan nilai *action* yang berasal dari bagian **F_EVLTR** di **A1**. Operasi di **A1** ditentukan oleh sinyal **ADD_SUB** yang berasal dari bit 6 dari kode operasi, HI untuk penjumlahan dan LO untuk pengurangan. Hasil operasi ini kemudian masuk ke **M1**. Nilai *output* yang baru ditentukan oleh sinyal ACC_IMM, apakah hasil operasi dari **A1** (*accumulate* : **ACC_IMM** = HI) atau **ACT[7:0]** (*immediate* : **ACC_IMM** = LO). Sistem pengkodean ini menjelaskan adanya kode 0, 1, 3 untuk tipe defuzzifikasi. Nilai *output* yang baru disimpan dulu di **D2** untuk sinkronisasi dan kemudian akan digunakan untuk *update* nilai *output* di *output storage*.

BAB 4

FUZZY INTEGRATED DEVELOPMENT ENVIRONMENT (FIDE)

4.1. Pendahuluan

FIDE (*Fuzzy Integrated Development Environment*) adalah program penunjang untuk **Kontroler FF** yang dibuat menggunakan bahasa pemrograman Visual C++ 6.0. Fungsi utama FIDE adalah untuk membuat *file-file* yang diperlukan untuk pemrograman *rule fuzzy* kontroler dan untuk mensimulasikan jalannya kontroler.

4.1.1. Spesifikasi FIDE :

- Berjalan pada sistem operasi Windows 95 / 98
- Menerima masukan parameter-parameter *fuzzy* dalam bentuk label, nilai dan kata-kata dan diubah sesuai format internal masing-masing menjadi 5 macam *file* yang dibutuhkan untuk pengoperasian **Kontroler FF** :
  - 4 buah *file* .mem, PAR1, PAR2, PAR2 dan RULE.mem dalam format *file* .mem LogiBLOX yang masing-masing digunakan untuk mengisi *input, output, fixed parameter* dan *rule storage* sebelum desain diimplementasikan, dan
  - *file* .hex, PARAM dalam format *file* hex Intel yang berisi seluruh data yang terdapat dalam keempat *file* .mem, yang disatukan menjadi satu *file*

4.1.2. Deskripsi

FIDE mempunyai dua fungsi utama :

- Penghasil file (*file generator)*, dan
- Simulator.

Kontroler FF dapat diprogram dengan dua cara :

- Mengisi **I/O & Fix Parameter Storage** serta **Program Storage** sebelum desain diimplementasikan, dengan menggunakan 4 buah *file* .mem, satu untuk masing-masing *storage*, atau
- Mengisi suatu unit *external memory* dengan *file* .hex yang berisi keempat parameter dan *rule* seperti di atas, lalu mengoperasikan kontroler untuk men-*download* isi *external memory* tersebut.

FIDE dapat menerima masukan parameter dan *rule fuzzy* dalam bentuk label, dan nilai dan kata-kata, mengubahnya menjadi kode-kode bahasa mesin yang dimengerti **Kontroler FF**, lalu memformatnya menjadi kelima *file* tersebut sesuai format masing-masing (penjelasan lebih lanjut tentang proses perubahan dapat diperoleh pada bagian **4.3**, **Penjelasan Program**).

mungkin dengan keadaan demo sebenarnya, *Input* dan konfigurasi diwujudkan dalam bentuk *DIP-Switch* dan *output* dalam bentuk tampilan *7-Segment*.

Untuk memperjelas cara kerja **Kontroler FF**, ditampilkan nilai *bus-bus* internal kontroler, seperti **CDBUS, INBUS, INPUT**, dan sebagainya (penjelasan tentang *bus-bus* internal **Kontroler FF** dapat diperoleh pada **Bab 3 FIELD PROGRAMMABLE GATE ARRAY**). Selain itu juga ditampilkan penjelasan proses yang terjadi pada kontroler pada setiap setengah periode *clock* utama.

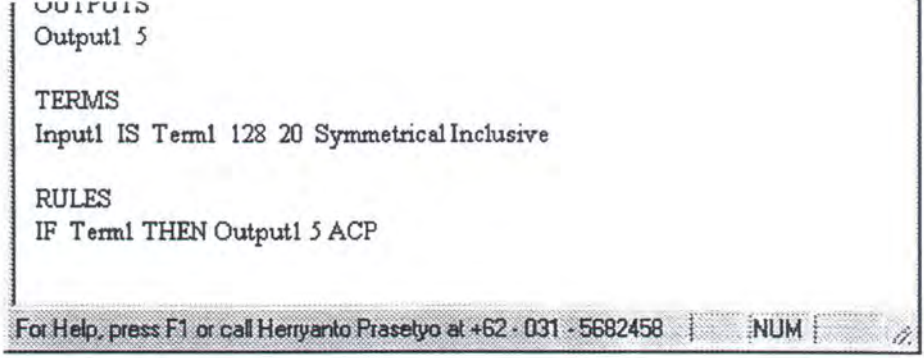
Simulator ini ditujukan untuk :

- menguji kebenaran proses perhitungan internal kontroler,
- mempelajari cara perhitungan internal kontroler, dan
- menguji suatu rule sebelum diimplementasikan pada FPGA.

4.2. Cara Pengoperasian FIDE

Proses pembuatan *rule* untuk **Kontroler FF** dengan menggunakan FIDE terdiri dari tiga bagian :

- pemasukan parameter dan rule,
- pembuatan *file-file* .mem dan .hex

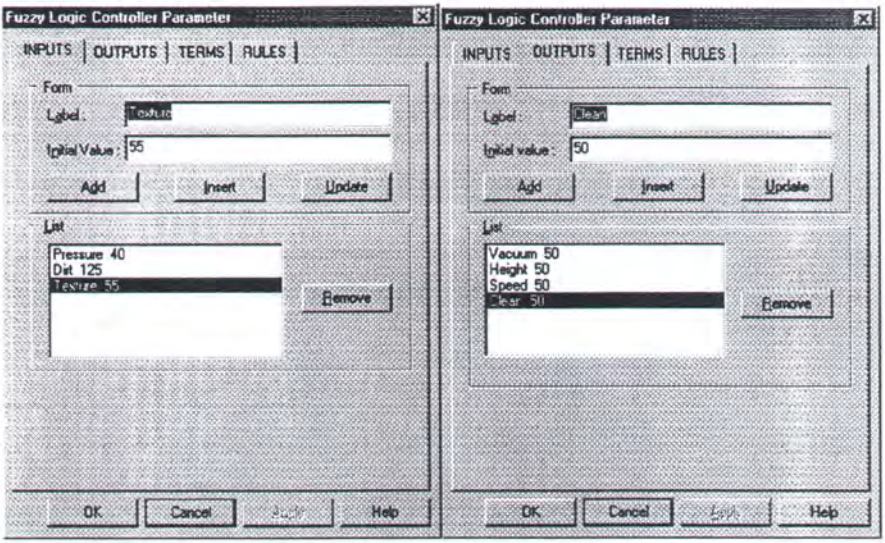


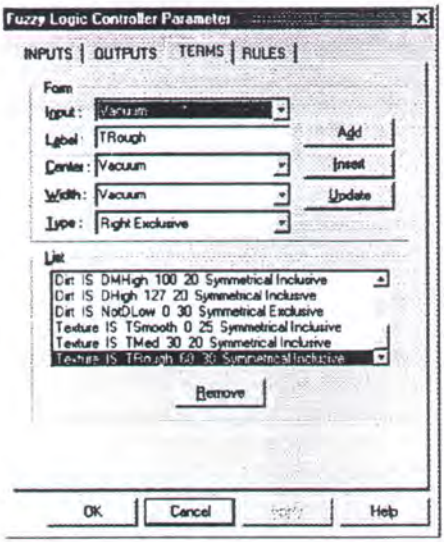
Gambar 4. 1.

Tampilan Awal **FIDE**

4.2.1. Pemasukan Parameter dan Rule

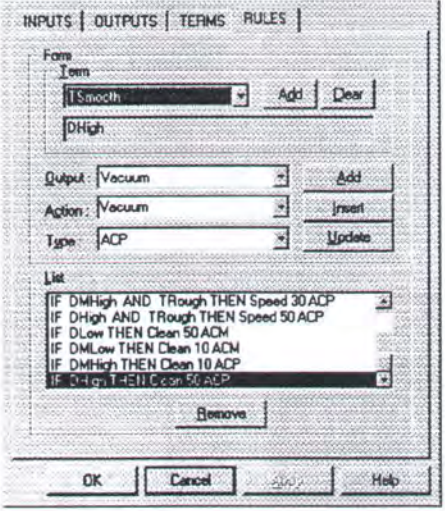
- Gunakan dokumen baru yang telah disediakan, ganti namanya bila perlu. Perhatikan adanya tampilan parameter dan *rule* awal dokumen baru tersebut. Untuk mengubah ukuran karakter pilih *menu* **View** -> **Text**, lalu pilih, **Smallest, Small, Medium, Large**, atau **Largest**.
- Tampilkan *dialog* "**Fuzzy Logic Controler Parameter**" dengan memilih *menu* **FLC -> Edit**, kombinasi tombol <**Ctrl+E**> atau *toolbar* .
- Isi / ubah parameter, mulai dari *Tab* "**INPUTS**", "**OUTPUTS**" dan "**TERMS**". Pada setiap akhir pengisian suatu *tab*, tekan tombol **Apply** supaya perubahan tersebut diikuti juga pada *tab-tab* yang lain. Label harus dimulai dengan huruf ('a'..'z', 'A'..'Z'). Untuk nilai-nilai *CENTER* dan *WIDTH* pada





Gambar 4. 2.

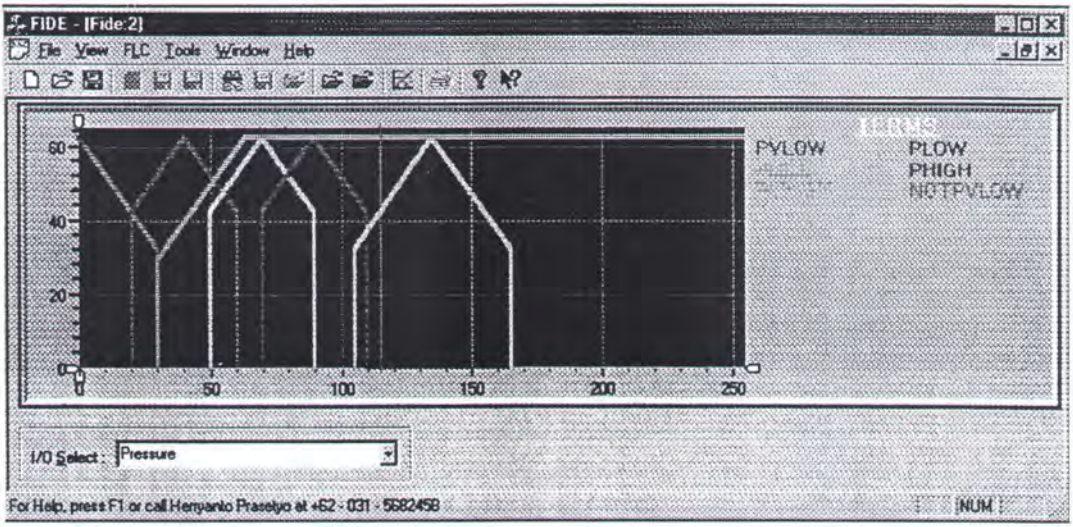
Dialog **Input**, **Output** dan **Term**



Gambar 4. 3.

Dialog **Rule**

- Setelah selesai, tekan tombol **OK** untuk meng-*update* isi variabel penyimpan *rule* dan parameter di *memory* dan melakukan konversi internal untuk persiapan pembuatan *file-file* .mem dan .hex atau tombol **Cancel** untuk membatalkan perubahan.
- Term-term yang telah dimasukkan dapat dilihat bentuk *membership-function*-nya dengan menggunakan **Term Viewer** dengan memilih *menu* **Tools** -> **Term Viewer**, kombinasi tombol <**Ctrl+T**>, atau *toolbar* .
- Dalam *window* **Term Viewer** ini dapat dipilih *term-term* dari *Input* atau *Output* mana yang akan ditampilkan, serta dapat dilihat nilai *fuzzy term-term*



Gambar 4. 4

Window **Term Viewer**

- Konfigurasi parameter dan *rule* dalam format internal FIDE tersebut dapat disimpan untuk pengubahan lebih lanjut dengan memilih *menu* **File -> Save As** untuk mengubah nama *file* atau **File -> Save**, kombinasi tombol <**Ctrl+S**>, atau *toolbar* untuk menyimpan langsung.
- File tersebut dapat dibuka dengan memilih *menu* **File -> Open**, kombinasi tombol <**Ctrl+O**>, atau *toolbar* . yang menyebabkan FIDE membuka *window* baru berisi dokumen tersebut.

buah *file* .mem.

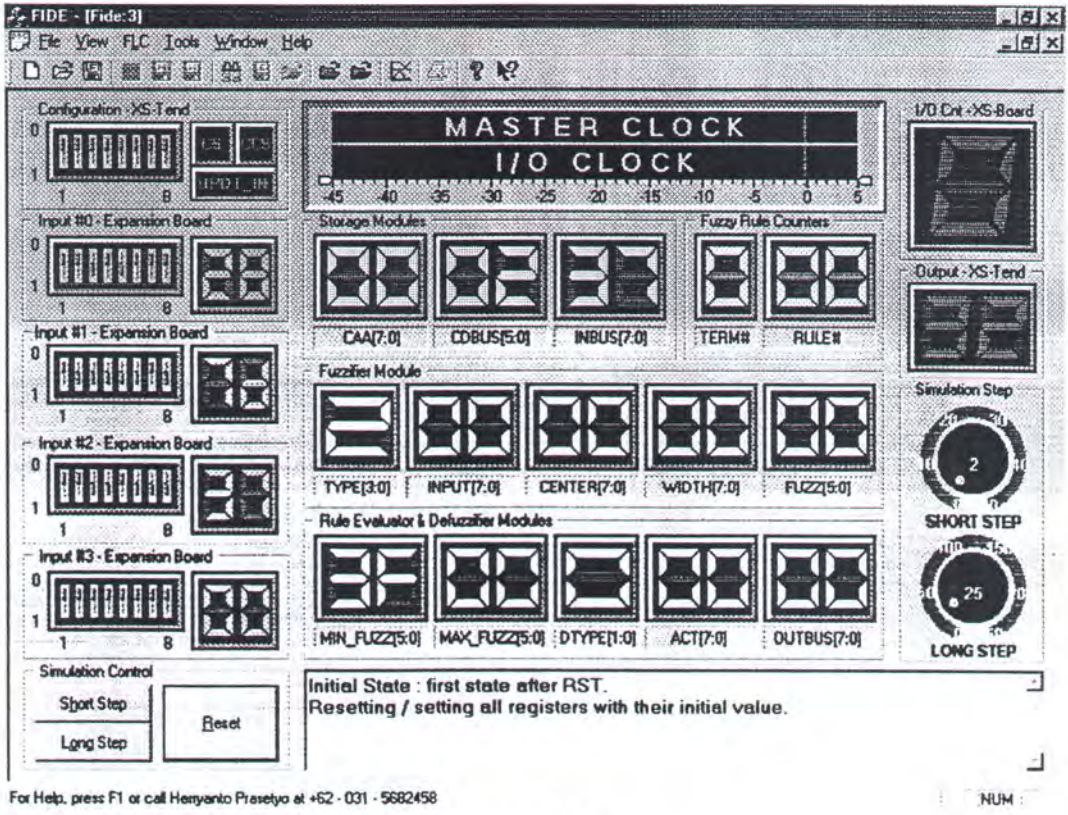
- Untuk membuat *file* .hex, pilih *menu* **FLC -> Generate -> .hex Files**, kombinasi tombol <**Ctrl+H**> atau *toolbar* . Lalu tentukan nama dan lokasi penyimpanan *file* tersebut.
- File-file yang telah dibuat dapat dilihat dengan memilih *menu* **Tools -> File Viewer -> .mem File**, kombinasi tombol <**Ctrl+1**> atau *toolbar* untuk *file* .mem dan *menu* **Tools -> File Viewer -> .hex File**, kombinasi tombol <**Ctrl+2**> atau *toolbar* untuk *file* .hex. File Viewer ini hanya dapat digunakan untuk melihat (*read-only*) isi *file-file* tersebut. Untuk mengubah isi *file* dapat digunakan editor teks seperti Notepad.

4.2.3. Simulasi Kontroler

- Untuk mengaktifkan simulator, pilih *menu* **Tools -> Simulator**, kombinasi tombol <**Ctrl+U**> atau *toolbar* . Perlu diperhatikan bahwa simulator hanya dapat diaktifkan bila *window* yang aktif adalah *window* utama, yang menampilkan parameter *fuzzy*, bukan *window* penampil isi *file*.

  Simulator terdiri dari beberapa bagian :

tidak mempunyai fungsi yang relevan dengan program simulator ini. Pemasukan nilai dapat dilakukan dengan *click mouse* pada tombol-tombol *DIP-Switch* atau penekanan tombol yang bersesuaian yang terdapat di sebelah *DIP-Switch* tersebut.



Gambar 4. 5.

Window **Simulator**

- **Input** mewakili 4 *DIP-Switch* pada yang digunakan untuk memasukkan nilai *input* yang baru pada **Kontroler FF**. Pemasukan nilai dilakukan

- **Long Step** untuk menjalankan simulasi dengan *step* panjang,
- **Reset** untuk mengulangi simulasi dari awal, mulai data ke-0. Tombol *reset* ini diimplementasikan secara *hardware* dalam bentuk *push-button* pada Xstend.

- **Clock Chart** digunakan untuk menampilkan bentuk gelombang kedua macam *clock* yang digunakan dalam **Kontroler FF**, *clock* utama (*master clock*) dan *clock I/O.* Perlu diperhatikan adanya garis merah pada diagram. Garis itu dapat digeser pada suatu nilai tertentu. Pergeseran ini akan menampilkan data simulasi pada urutan nilai tersebut. Juga perhatikan adanya penunjuk warna putih pada tepi bawah diagram. Penunjuk ini dapat digeser ke kanan atau ke kiri untuk memperbesar atau memperkecil skala – x dari diagram.
- **13 7-Segment** berwarna biru yang digunakan untuk menampilkan nilai *bus-bus* internal **Kontroler FF**. Penjelasan lebih lanjut tentang nilai-nilai ini dapat dilihat pada bagian *hardware*.
- **2 7-Segment** berwarna kuning yang menunjukkan urutan *term* dan *rule* yang sedang diproses.

- **Simulation Step** digunakan untuk menentukan lamanya *step* simulasi, untuk *short step* dan *long step*.
- **Comment** menampilkan komentar tentang proses yang sedang terjadi pada simulasi.

- Pengoperasian simulator dilakukan dengan :
  - Atur **Simulation Step**.
  - Tekan tombol **Reset**.
  - Tentukan **konfigurasi** kontroler dan nilai tiap **Input**
  - Jalankan simulator dengan menekan tombol **Short Step** atau **Long Step**.
  - Lakukan perubahan **konfigurasi** atau nilai **input** bila perlu.
  - Pindahkan garis merah pada diagram *clock* untuk melihat hasil simulasi terdahulu bila perlu.
  - Diagram dapat di-*zoom* dengan menarik *handle* putih di tepi bawah dengan *mouse*.
  - Untuk mengatur besarnya karakter pada bagian komentar dapat dilakukan dengan memilih *menu* **View -> Text**, lalu dapat dipilih antara **Smallest**, **Small**, **Medium**, **Large**, atau **Largest**.

dalam aplikasi, sedang View adalah kelas yang menampilkan data-data tersebut pada *window* dan berfungsi sebagai *user-interface* untuk memfasililtasi perubahan pada *document*. Pada FIDE terdapat 2 *document* dan 3 *view* :

- Dokumen Utama (FIDE),
- Dokumen File,
- View Utama (FIDE),
- View Term,
- View Simulator, dan
- View File.

Hanya akan dijelaskan bagian-bagian utama program, sedang penjelasan lebih lengkap dapat diperoleh pada *source-code* program. File deskripsi masing-masing kelas akan disebutkan untuk memudahkan pencarian.

4.3.1. Dokumen Utama (FIDE)

Dokumen utama ini merupakan inti dari program. Deskripsi kelas ini terdapat pada *file* **FIDEDoc.h** dan **FIDEDoc.cpp**. Dokumen ini berisi :

- **Data Term** yang terdiri dari **Label**, *string* (nilai) **Input**, **Center**, dan **Width** serta **tipe Membership Function** *term* tersebut (6 macam, sesuai pada **Kontroler FF**). Deskripsi kelas ini terdapat pada *file* **FIDEData.h** dan **FIDEData.cpp**,
- **Data Rule** yang terdiri dari **term-term** yang menyusunnya, **output** yang diacu, *string* (nilai) **Action** serta **tipe defuzzifikasinya** (3 macam, sesuai **Kontroler FF**). Deskripsi kelas ini terdapat pada *file* **FIDEData.h** dan **FIDEData.cpp**,
- **Data Terkonversi** yang merupakan hasil konversi keempat data tersebut di atas. Kelas ini berisi dari :
  - **array nilai**, 64 bilangan yang berisi **nilai awal Input** dan **Output** serta **konstanta-konstanta** yang digunakan untuk **term** dan **rule**. Dari *array* ini, 16 bilangan pertama akan diisikan pada PAR1.mem, 16 bilangan berikutnya pada PAR2.mem, dan 32 bilangan terakhir pada PAR3.mem. Array ini juga akan ditempatkan pada bagian akhir dari PARAM.hex,
  - **array rule**, 256 bilangan yang berisi **kode bahasa mesin** dari **rule** yang kemudian akan diisikan pada RULE.mem dan bagian awal dari

data di atas.

Deskripsi kelas ini terdapat pada *file* **FIDEData.h** dan **FIDEData.cpp**,

- **Data Simulasi** yang digunakan untuk mengisi obyek-obyek *Graphical User Interface (GUI)* pada *window* Simulator. Deskripsi kelas ini terdapat pada *file* **FIDEData.h** dan **FIDEData.cpp**. Kelas ini terdiri dari :
  - **data *state*** simulasi, terdiri dari **Present** dan **Previous State** yang digunakan untuk menentukan operasi kontroler pada suatu urutan data simulasi tertentu,
  - **data *clock***, terdiri dari **Master** dan **I/O Clock** yang merupakan nilai *clock* tersebut pada urutan data simulasi tertentu,
  - **data *I/O***, menyimpan data-data yang ditampilkan pada *hardware*, terdiri dari *flag-flag* **konfigurasi** (CS, CCS dan UPDT_IN), keempat **input** dan keempat **output** pada suatu urutan data simulasi tertentu,
  - **data buses**, menyimpan nilai **bus-bus internal Kontroler FF** yang tidak ditampilkan pada *hardware* (CAA, CAB, CDBUS, INBUS, TYPE, INPUT, CENTER, WIDTH, FUZZ, MIN_FUZZ, MAX_FUZZ, ACT, DTYPE dan OUTBUS) pada suatu urutan data simulasi tertentu,

secara dinamik saat eksekusi File Viewer dan segera dimusnahkan saat *window view*-nya ditutup. Deskripsi kelas ini terdapat pada *file* **FIDEFile.h** dan **FIDEFile.cpp**.

4.3.3. View Utama (FIDE)

View Utama ini merupakan pasangan dari Dokumen Utama dan mempunyai fungsi sebagai berikut :

- **menampilkan** data-data **parameter fuzzy** dalam bentuk kalimat di *window*, dengan fasilitas *scrolling, print* dan pengubahan ukuran karakter,
- **memfasilitasi pengubahan data parameter *fuzzy*** dengan menyalin data dari dokumen utama ke variabel penyimpan pada kelas *dialog* **Fuzzy Logic Controller Parameter**, menyalin isi variabel yang telah diubah oleh *user* ke dokumen utama bila *dialog* ditutup dengan menekan tombol OK, atau menghapus variabel tersebut bila *dialog* ditutup dengan menekan tombol Cancel,
- **memfasilitasi operasi pembuatan *file-file*** .mem dan .hex,
- **memfasilitasi** pengaktifan *window* **Simulator**,

4.3.4. View Term

View Term merupakan kelas *view* pembantu untuk dokumen utama yang berfungsi untuk menampilkan representasi grafik *term-term* dari suatu nilai *input* atau *output*. Hal ini dilakukan dengan menggunakan komponen GUI *stripchart* yang merupakan komponen *ActiveX*. Deskripsi kelas ini terdapat pada *file* **FIDETrmV.h** dan **FIDETrmV.cpp**.

4.3.5. View Simulator

View Simulator merupakan kelas *view* pembantu untuk dokumen utama yang mempunyai fungsi sebagai berikut :

- **menampilkan** komponen-komponen **GUI** dari Simulator,
- **memfasilitasi perhitungan simulasi** yang dilakukan oleh dokumen utama dengan mengirimkan data masukan dari komponen-komponen GUI Simulator pada dokumen utama, memanggil fungsi perhitungan dari dokumen utama, dan mengirimkan hasil perhitungan tersebut kembali ke komponen-komponen GUI Simulator tersebut.

Deskripsi kelas ini terdapat pada *file* **FIDESIM.h** dan **FIDESIM.cpp**.

komponen Diagram (*Strip Chart*) untuk tampilan *multi block* dan *I/O block*,

- komponen *Toggle* untuk konfigurasi,
- komponen *Knob* untuk pengaturan *simulation step*.

Penggunaan komponen ActiveX menyebabkan diciptakannya kelas-kelas **wrapper** untuk mengakses *property*, *method* dan *event* dari komponen-komponen tersebut. Deskripsi kelas-kelas ini terdapat pada *file-file* **font.h**, **font.cpp**, **knob.h**, **knob.cpp**, **led.h**, **led.cpp**, **numled.h**, **numled.cpp**, **picture.h**, **picture.cpp**, **strip.h**, **strip.cpp**, **toggle.h** dan **toggle.cpp**.

4.3.6. View File

View ini merupakan pasangan Dokumen File dan digunakan untuk **File Viewer**. View ini hanya berfungsi untuk menampilkan data yang telah tersimpan dalam Dokumen File ke *window* dan menyediakan fasilitas *scrolling* serta pengubahan ukuran karakter. Deskripsi kelas ini terdapat pada *file* **FIDEFile.h** dan **FIDEFile.cpp**.

4.3.7. Fungsi Konversi

Fungsi Konversi terdiri dari tiga bagian :

nilai nol dan tidak akan digunakan untuk penyimpanan konstanta **term**.

- Pemasukan data-data **term**, untuk memasukkan nilai-nilai konstanta yang digunakan untuk nilai *center* dan *width* ke **array nilai** pada **Data Terkonversi**, lalu menyimpan label, tipe, *input*, *center*, dan *width term* tersebut ke **array term** pada **Data Terkonversi**. Untuk *input, center* dan *width*, yang disimpan adalah urutan nilai tersebut pada **array nilai** yang pada implementasi *hardware* akan menjadi alamat nilai tersebut pada ***I/O & Fix Parameter Storage***. Hasil dari proses ini adalah suatu **tabel referensi term-term** yang digunakan yang terdiri dari label untuk fasilitas pencarian *term* pada pemasukan *rule* dan data 4 bilangan sesuai format kode internal **Kontroler FF**, yang terdiri dari tipe *membership function*, alamat *input*, alamat *center* dan alamat *width* dari *term* tersebut. Selain itu proses ini juga menyelesaikan pengisian **array nilai** yang memuat nilai awal *input* dan *output* serta konstanta-konstanta yang digunakan untuk parameter *term*. Array inilah yang kemudian akan diproses menjadi PAR1.mem, PAR2.mem, PAR3.mem serta bagian akhir dari PARAM.hex dan kemudian diisikan pada *I/O & Fix Parameter Storage* pada implementasi *hardware*-nya.

(**EOR**) beserta kode defuzzifikasi serta alamat nilai *action* pada **array rule**. Bila *rule* ini merupakan *rule* terakhir dari kumpulan *rule-rule* yang mengacu pada suatu *output*, maka juga diberikan kode akhir *rule-rule* (**EORRSO**) beserta alamat *output* yang diacu ke **array rule**. Akhirnya, jika *rule* ini merupakan *rule* terakhir untuk **Kontroler FF** ini, sedangkan **array rule** belum terisi penuh, diberikan kode akhir *rule* (**EORs**) yang menandai akhir *rule* dan tanda untuk kembali memproses *rule* pertama pada *hardware*.

4.3.8. Fungsi Simulasi

Fungsi Simulasi merupakan fungsi perhitungan nilai-nilai yang akan ditampilkan pada komponen-komponen GUI pada *window* Simulator, yang meliputi beberapa tahap :

- **Perhitungan clock**. **Master clock** (**MCLK**) dan **I/O clock** (**IOCLK**) dihitung berdasarkan nilai sebelumnya dan *flag clock* yang digunakan (**CS** dan **CCS**).
- **Perhitungan I/O**. Pada bagian ini dihitung nilai-nilai yang ditampilkan pada demo *hardware* :

MILIK PERPUSTAKAAN ITS

- **Update Output**, **Output** yang ditampilkan di GUI di-*update* dengan nilai **Output** pada urutan sesuai CAB.

- **Perhitungan bus-bus internal**. Pada bagian ini dihitung nilai *bus-bus* bagian dalam **Kontroler FF** yang tidak ditampilkan pada saat demo *hardware* :
  - **Perhitungan State**. Perhitungan **state** merupakan bagian yang paling penting dalam perhitungan *bus-bus* internal ini karena akan menentukan jenis perhitungan yang akan dilakukan. Perhitungan **state** ini dilakukan berdasarkan tipe operasi (**TYPE**) dan **state** sebelumnya dan dilakukan oleh **fungsi perhitungan state** pada kelas **Data Simulasi**.
  - **Perhitungan Storage Modules** terdiri dari :
    - **Perhitungan CAA**, *program-counter* yang dihitung oleh **fungsi perhitungan CAA** pada kelas **Data Simulasi**,
    - **Perhitungan CDBUS**, mengisi **CDBUS** dengan nilai dari **array rule** dari **Data Terkonversi** sesuai urutan yang ditunjukkan oleh **CAA**,
    - **Perhitungan INBUS**, mengisi **INBUS** dengan nilai dari **array nilai** dari **Data Terkonversi** sesuai urutan yang ditunjukkan oleh **CDBUS**.
  - **Perhitungan Fuzzifier Module** terdiri dari :

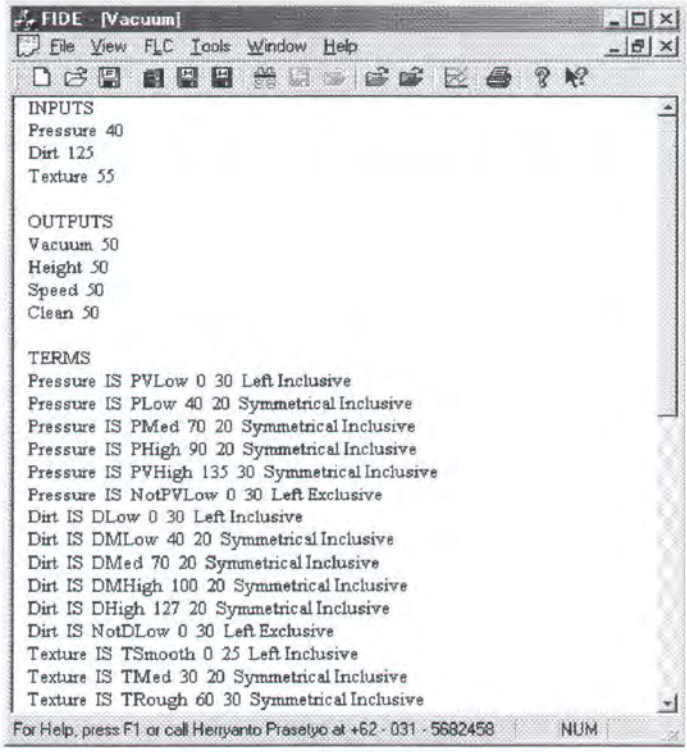
pada transisi negatif **MCLK** dan **state** St_LD_In,

- **Perhitungan CENTER**, mengisi **CENTER** dengan nilai pada **INBUS** pada transisi negatif **MCLK** dan **state** St_Ld_Cnt,
- **Perhitungan WIDTH**, mengisi **WIDTH** dengan nilai pada **INBUS** pada transisi negatif **MCLK** dan **state** St_Ld_Wdt,
- **Perhitungan FUZZ**, mengisi **FUZZ** dengan hasil **fungsi perhitungan FUZZ** pada kelas **Data Simulasi** pada transisi negatif **MCLK** dan **state** St_En_AlfCt.

- **Perhitungan Rule Evaluator & Defuzzifier Modules** terdiri dari :
  - **Perhitungan MIN_FUZZ**, mengisi **MIN_FUZZ** dengan hasil **fungsi perhitungan MIN_FUZZ** pada kelas **Data Simulasi** pada transisi negatif **MCLK** dan **state** St_En_Min,
  - **Perhitungan MAX_FUZZ**, mengisi **MAX_FUZZ** dengan hasil **fungsi perhitungan MAX_FUZZ** pada kelas **Data Simulasi** pada transisi negatif **MCLK** dan **state** St_En_Max,
  - **Perhitungan DTYPE**, mengisi **DTYPE** dengan 4 bit MSB **INBUS** pada transisi negatif **MCLK** dan **state** St_En_Max,

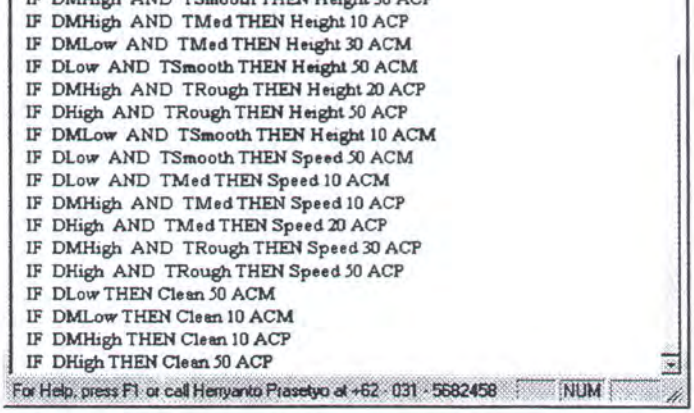
- **Pemberian komentar** tentang proses yang terjadi pada **state** tertentu.

Untuk pengujian **Kontroler FPGA-Fuzzy** dan program **FIDE** digunakan suatu model penerapan kontroler *fuzzy* sederhana, sistem pengaturan *vacuum cleaner*. Sistem ini menggunakan 3 *input* : *pressure*, tekanan pada *vacuum*, *dirt*, jumlah debu dan *texture*, tekstur lantai dan menghasilkan 4 *output* : *vacuum*, pengontrol *vacuum*, *height*, ketinggian sikat, *speed*, kecepatan putaran sikat dan *clean*, indikator kebersihan lantai.



Gambar 5. 1.

Input, Output dan *Term-Term* yang digunakan untuk pengujian



Gambar 5. 2.

Rule yang digunakan untuk pengujian

Pengujian tidak berorientasi pada keberhasilan pengaturan proses tetapi lebih berorientasi pada kebenaran perhitungan yang dilakukan **Kontroler FF**, dan berjalannya fungsi-fungsi pada program **FIDE**.

5.2. Pengujian Kontroler FPGA-Fuzzy dan Program FIDE

Pengujian dilakukan pertama dengan membuat *file* yang dibutuhkan untuk mengisi *rule* **Kontroler FF** dengan menggunakan **FIDE**, mensimulasikan dan memeriksa bentuk *term-term*-nya, lalu memprogram dan mengoperasikan **Kontroler FF** untuk mengamati hasil operasi *hardware*-nya. Hasil operasi **Kontroler FF** ini kemudian dibandingkan dengan hasil simulasi **FIDE**.

- Setelah dicapai hasil yang diinginkan, dibuat *file* .hex-nya, karena parameter *fuzzy* ini akan dimasukkan melalui RAM eksternal, supaya tidak perlu melalui proses implementasi ulang yang banyak memakan waktu.

5.2.2. Pengujian Kontroler FPGA-Fuzzy

- **XS-40 Board** digabungkan dengan **XStend**, lalu diberi *power-supply* (sekitar 8-10V) melalui **J9** pada **XS-40 Board**.
- Kemudian *port* DB25 pada **XS-40 Board** dihubungkan dengan **PC Parallel Port**.
- *File* bitstream (**fuzzy.bit**) dan *file* .hex hasil program **FIDE** di-*download*-kan ke FPGA dengan menggunakan program **XSLoad**.
- Berikan sinyal HI pada PGM dengan meng-OFF-kan *DIP Switch* 4 pada **XStend** supaya **Kontroler FF** dapat men-*download rule fuzzy* dari RAM eksternal.
- Nilai yang ditampilkan pada *7-Segment* dapat diamati dan dibandingkan dengan hasil simulasi menggunakan program **FIDE**.

Dengan terselesaikannya Tugas Akhir ini dapat ditarik beberapa kesimpulan :

- Pemanfaatan teknologi FPLD dan HDL memungkinkan dibuatnya IC sesuai kebutuhan dalam waktu yang relatif singkat dan mudah.
- Untuk memperoleh hasil desain yang efektif dan efisien diperlukan pemahaman struktur dan sumber daya IC FPLD yang digunakan.
- Metode *Mixed-Design* pada **Xilinx Foundation Series** memberikan akses penggunaan fasilitas khusus FPGA dan memungkinkan pemanfaatan CLB secara lebih efektif.
- Desain **Kontroler FPGA-Fuzzy** pada IC FPGA XC-4005XL terealisasi dengan menggunakan seluruh CLB (196) yang ada pada XC-4005XL, 49 pin dari 63 pin I/O yang disediakan dengan jumlah total gerbang logika ekuivalen dengan 11910.
- Penggunaan teknologi *ActiveX* yang ditawarkan program Microsoft Visual C++ sangat membantu pembuatan program **FIDE**, terutama untuk mengimplementasikan komponen-komponen *Graphical User Interface* (GUI) pada *Window* **Simulator**.

6.2. Harapan dan Saran

Kontroler FPGA-Fuzzy dan program **FIDE** merupakan suatu langkah awal dalam pengembangan kontroler *fuzzy* pada FPGA. Keduanya merupakan satu kesatuan kontroler *fuzzy* yang **bisa digunakan**. Karena Tugas Akhir ini hanya merupakan langkah awal, Penulis berharap ada teman-teman, khususnya dari Bidang Studi Elektronika Jurusan Teknik Elektro FTI-ITS yang berminat untuk melanjutkan Tugas Akhir ini dengan menggunakannya untuk dioptimasikan pada suatu aplikasi tertentu sehingga dapat benar-benar menunjukkan keunggulan kontroler *fuzzy* berbasis FPGA ini dibandingkan kontroler *fuzzy* konvensional.

Berikut Penulis akan memberikan beberapa saran untuk lebih meningkatkan kemampuan **Kontroler FPGA-Fuzzy** ini :

- Menggunakan IC FPGA dengan kapasitas lebih besar, misalnya XC4010XL atau XC4013XL.
- *Slope* pada *membership function* dengan mudah dapat dibuat variabel, yaitu dengan menyisipkan *shift-register* di antara **Subtracter** dan **Alfa-Cut Calculator** dan sedikit penyesuaian pada **Alfa-Cut Decoder**-nya pada modul

Counter sehingga memungkinkannya untuk menyimpan sementara nilai yang lama dan me-*load* suatu nilai tertentu dan kembali ke nilai yang lama bila perlu.

- Menambahkan fasilitas-fasilitas yang sesuai dengan aplikasi yang akan digunakan, misalnya penambahan rangkaian PWM, *filter* digital, dan sebagainya.
- Perlu pula mempelajari dan mengembangkan fasilitas JTAG yang terdapat pada IC FPGA yang memungkinkan pemrograman dan pengetesan IC hanya dengan menggunakan 4 pin.
- Juga perlu melaksanakan pembuatan *proto-typing* buatan sendiri sehingga tidak selalu bergantung pada **XS-Board** dan **XStend**. Hal ini juga akan lebih memberikan kebebasan desain karena tidak perlu menyesuaikan dengan tata letak komponen pada kedua **Board** tersebut.

[2] ____, **NLX221, NLX221P, NLX222, NLX222P STAND ALONE FUZZY LOGIC CONTROLLERS PRELIMINARY DATA**, Adaptive Logic, 1994.

[3] ____, **THE PROGRAMMABLE LOGIC DATA BOOK 1998**, Xilinx, 1998.

[4] ____, **Xilinx Foundation Series Online Help System**, Xilinx, 1998.

[5] ____, **XS40, XSP, and XS95 Board Manual**, XESS corp., 1998.

[6] ____, **XStend Board V1.3 Manual**, XESS corp., 1999.

[7] Camilleri, Nick and Chris Lockhard, **XAPP 043.000 Improving XC4000 Design Performance**, Xilinx.

[8] Gregory, Kate, **Special Edition Using Visual C++ 6**, QUE, 1998.

[9] Toth, Viktor, **Visual C++ 5 Unleashed**, Sams Publishing, 1997.

[10] Van den Bout, Dave, **The Practical Xilinx Designer Lab Book**, Prentice-Hall International, Inc, 1998.

LAMPIRAN

	2432	25,000	32,768	15,000 - 45,000	32 x 32	1,024	2,560	25
XL	2432	28,000	32,768	18,000 - 50,000	32 x 32	1,024	2,560	25
XL	3078	36,000	41,472	22,000 - 65,000	36 x 36	1,296	3,168	28
	3800	44,000	51,200	27,000 - 80,000	40 x 40	1,600	3,840	32
	4598	52,000	61,952	33,000 - 100,000	44 x 44	1,936	4,576	35
	5472	62,000	73,728	40,000 - 130,000	48 x 48	2,304	5,376	38
	7448	85,000	100,352	55,000 - 180,000	56 x 56	3,136	7,168	44

ax values of Typical Gate Range include 20-30% of CLBs used as RAM.

ctionality in low-voltage families is the same as sponding 5-Volt family, except where numerical are made to timing or power.

on

ries devices are implemented with a regular, grammable architecture of Configurable Logic Bs), interconnected by a powerful hierarchy of ting resources, and surrounded by a perimeter mable Input/Output Blocks (IOBs). They have outing resources to accommodate the most erconnect patterns.

s are customized by loading configuration data memory cells. The FPGA can either actively nfiguration data from an external serial or PROM (master modes), or the configuration written into the FPGA from an external device eripheral modes).

ries FPGAs are supported by powerful and d software, covering every aspect of design atic or behavioral entry, floorplanning, simula- atic block placement and routing of intercon- e creation, downloading, and readback of the n bit stream.

nx FPGAs can be reprogrammed an unlimited mes, they can be used in innovative designs where hardware is changed dynamically, or where ware must be adapted to different user applica FPGAs are ideal for shortening design and develop cycles, and also offer a cost-effective solution for pr tion rates well beyond 5,000 systems per month. For lo high-volume unit cost, a design can first be implement the XC4000E or XC4000X, then migrated to one of X compatible HardWire mask-programmed devices.

Taking Advantage of Reconfiguration

FPGA devices can be reconfigured to change logic fun while resident in the system. This capability gives the tem designer a new degree of freedom not available any other type of logic.

Hardware can be changed as easily as software. De updates or modifications are easy, and can be mad products already in the field. An FPGA can even be re figured dynamically to perform different functions at d ent times.

Reconfigurable logic can be used to implement sy self-diagnostics, create systems capable of being reco ured for different environments or operations, or imple multi-purpose hardware for a given application. A added benefit, using reconfigurable FPGA devices si fies hardware design and debugging and shortens pr time-to-market.

E device is pinout- and bitstream-compatible
rresponding XC4000 device. An existing
tream can be used to program an XC4000E
ever, since the XC4000E includes many new
XC4000E bitstream cannot be loaded into an
ce.

ries devices are not bitstream-compatible with
ray size devices in the XC4000 or XC4000E
wever, equivalent array size devices, such as
XC4025E, XC4028EX, and XC4028XL, are
atible.

nents in XC4000E and XC4000X

ystem Speed

d XC4000X devices can run at synchronous
k rates of up to 80 MHz, and internal perfor-
xceed 150 MHz. This increase in performance
ious families stems from improvements in both
essing and system architecture. XC4000
es use a sub-micron multi-layer metal process.
many architectural improvements have been
scribed below.

XL family is a high performance 3.3V family
35µ SRAM technology and supports system
MHz.

nce

ies -2 and faster speed grades are fully PCI
C4000E and XC4000X devices can be used to
one-chip PCI solution.

f the carry logic chain has increased dramati-
parameters, such as the delay on the carry
h a single CLB (TBYP), have improved by as

either level-sensitive (asynchronous) single-port R
edge-triggered (synchronous) single-port RAM, edge
gered (synchronous) dual-port RAM, or as combinat
logic.

Configurable RAM Content

The RAM content can now be loaded at configuration t
so that the RAM starts up with user-defined data.

H Function Generator

In current XC4000 Series devices, the H function gene
is more versatile than in the original XC4000. Its inputs
come not only from the F and G function generators
also from up to three of the four control input lines. Th
function generator can thus be totally or partially inde
dent of the other two function generators, increasing
maximum capacity of the device.

IOB Clock Enable

The two flip-flops in each IOB have a common clock en
input, which through configuration can be activated ind
ually for the input or output flip-flop or both. This c
enable operates exactly like the EC pin on the XC4
CLB. This new feature makes the IOBs more versatile,
avoids the need for clock gating.

Output Drivers

The output pull-up structure defaults to a TTL
totem-pole. This driver is an n-channel pull-up transi
pulling to a voltage one transistor threshold below Vcc,
like the XC4000 family outputs. Alternatively, XC4
Series devices can be globally configured with CMOS
puts, with p-channel pull-up transistors pulling to Vcc. A
the configurable pull-up resistor in the XC4000 Series
p-channel transistor that pulls to Vcc, whereas in the c
nal XC4000 family it is an n-channel transistor that pul
a voltage one transistor threshold below Vcc.

ll Pin Pull-Up Resistors

guration, the three mode pins, M0, M1, and ak pull-up resistors. For the most popular con- de, Slave Serial, the mode pins can thus be ted.

ode inputs can be individually configured with ak pull-up or pull-down resistors after configu-

AM input pin has a permanent weak pull-up.

3000A, XC4000 Series devices have "Soft hen the configuration process is finished and tarts up, the first activation of the outputs is slew-rate limited. This feature avoids poten- ounce when all outputs are turned on simulta- nediately after start-up, the slew rate of the tputs is, as in the XC4000 family, determined ual configuration option.

XC4000A Compatibility

000 bitstreams can be used to configure an vice. XC4000A bitstreams must be recompiled n the XC4000E due to improved routing though the devices are pin-for-pin compatible.

regular global clocks, a Fast Capture latch driven by early clock is available. The input data can be ini loaded into the Fast Capture latch with the early clock, transferred to the input flip-flop or latch with the low-s global clock. A programmable delay on the input cai used to avoid hold-time requirements. See "IOB Input nals" on page 23 for more information.

Latch Capability in CLBs

Storage elements in the XC4000X CLB can be config as either flip-flops or latches. This capability makes FPGA highly synthesis-compatible.

IOB Output MUX From Output Clock

A multiplexer in the IOB allows the output clock to se either the output data or the IOB clock enable as the ou to the pad. Thus, two different data signals can share a gle output pad, effectively doubling the number of de outputs without requiring a larger, more expensive p age. This multiplexer can also be configured as AND-gate to implement a very fast pin-to-pin path. "IOB Output Signals" on page 26 for more information.

Additional Address Bits

Larger devices require more bits of configuration data daisy chain of several large XC4000X devices may req a PROM that cannot be addressed by the eighteen addr bits supported in the XC4000E. The XC4000X Se therefore extends the addressing in Master Parallel con uration mode to 22 bits.

Specifications

in-to-pin input and output parameters, the a.c. parameter delay specifications included in this documer measuring internal test patterns. All specifications are representative of worst-case supply voltage and jur conditions. The parameters included are common to popular designs and typical applications. For d ns requiring more detailed timing information, see the appropriate family a.c. supplements available o NX at http://www.xilinx.com.

Maximum Ratings

	Description			Un
	Supply voltage relative to GND		-0.5 to 4.0	V
	Input voltage relative to GND (Note 1)		-0.5 to 5.5	V
	Voltage applied to 3-state output (Note 1)		-0.5 to 5.5	V
	Longest Supply Voltage Rise Time from 1 V to 3V		50	m
	Storage temperature (ambient)		-65 to +150	°C
	Maximum soldering temperature (10 s @ 1/16 in. = 1.5 mm)		+260	°C
	Junction temperature	Ceramic packages	+150	°C
		Plastic packages	+125	°C

mum DC overshoot or undershoot above V_{CC} or below GND must be limited to either 0.5 V or 10 mA, whichever is eas ve. During transitions, the device pins may undershoot to -2.0 V or overshoot to + 7.0 V, provided this over- or undershoot han 10 ns and with the forcing current being limited to 200 mA.

ses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stres only, and functional operation of the device at these or any other conditions beyond those listed under Operating Conditio nplied. Exposure to Absolute Maximum Ratings conditions for extended periods of time may affect device reliability.

ended Operating Conditions

Description		Min	Max	U
Supply voltage relative to GND, T_J = 0 °C to +85°C	Commercial	3.0	3.6	V
Supply voltage relative to GND, T_J = -40°C to +100°C	Industrial	3.0	3.6	V
High-level input voltage		50% of V_{CC}	5.5	V
Low-level input voltage		0	30% of V_{CC}	V
nput signal transition time			250	n

ction temperatures above those listed as Operating Conditions, all delay parameters increase by 0.35% per °C. and output measurement threshold is ~50% of V_{CC}.

Input capacitance (sample tested)	BGA, SBGA, PQ, HQ, MQ packages		10	pF
	PGA packages		16	pF
Pad pull-up (when selected) @ V_{in} = 0 V (sample tested)		0.02	0.25	mA
Pad pull-down (when selected) @ V_{in} = 3.6 V (sample tested)		0.02	0.15	mA
Horizontal Longline pull-up (when selected) @ logic Low		0.3	2.0	mA

th up to 64 pins simultaneously sinking 12 mA.
th no output current loads, no active input or Longline pull-up resistors, all I/O pins Tri-stated and floating.

L Global Buffer Switching Characteristic Guidelines

e switching parameters is modeled after testing methods specified by MIL-M-38510/605. All devices are 1
tested. Internal timing parameters are derived from measuring internal test patterns. Listed below
ve values where one global clock input drives one vertical clock line in each accessible column, and wher
OB and CLB flip-flops are clocked by the global clock net.

vertical clock lines are connected, the clock distribution is faster; when multiple clock lines per column are dr
ne global clock, the delay is longer. For more specific, more precise, and worst-case guaranteed data, reflec
uting structure, use the values provided by the static timing analyzer (TRCE in the Xilinx Development Syst
notated to the simulation netlist. These path delays, provided as a guideline, have been extracted from the s
zer report. All timing parameters assume worst-case operating conditions (supply voltage and junc
Values apply to all XC4000XL devices and are expressed in nanoseconds unless otherwise noted.

L Global Low Skew Buffer to Clock K

		Speed Grade	All	-3	-2	-1	-09	-08	Un
Description	**Symbol**	**Device**	**Min**	**Max**	**Max**	**Max**	**Max**	**Max**	
oad through GLS buffer to put, K	T_{GLS}	XC4002XL	0.3	2.1	1.8	1.6	1.5		n
		XC4005XL	0.4	2.7	2.3	2.0	1.9		n
		XC4010XL	0.5	3.2	2.8	2.4	2.3		n
		XC4013XL	0.6	3.6	3.1	2.7	2.6	2.3	n
		XC4020XL	0.7	4.0	3.5	3.0	2.9		n
		XC4028XL	0.9	4.4	3.8	3.3	3.2		n
		XC4036XL	1.1	4.8	4.2	3.6	3.5	3.1	n
		XC4044XL	1.2	5.3	4.6	4.0	3.9		n
		XC4052XL	1.3	5.7	5.0	4.5	4.4		n
		XC4062XL	1.4	6.3	5.4	4.7	4.6	4.0	n
		XC4085XL	1.6	7.2	6.2	5.7	5.5		n

		Speed Grade	All	-3	-2	-1	-09	-08	Un
Description	Symbol	Device	Min	Max	Max	Max	Max	Max	
oad through GE buffer to any put.	T_{GE}	XC4002XL	0.1	1.6	1.4	1.3	1.2		n
		XC4005XL	0.3	1.9	1.8	1.7	1.6		n
		XC4010XL	0.3	2.2	1.9	1.7	1.7		n
		XC4013XL	0.4	2.4	2.1	1.8	1.7	1.5	n
		XC4020XL	0.4	2.6	2.2	2.1	2.0		n
		XC4028XL	0.3	2.8	2.4	2.1	2.0		n
		XC4036XL	0.3	3.1	2.7	2.3	2.2	1.9	n
		XC4044XL	0.2	3.5	3.0	2.6	2.4		n
		XC4052XL	0.3	4.0	3.5	3.0	3.0		n
		XC4062XL	0.3	4.9	4.3	3.7	3.4	3.0	n
		XC4085XL	0.4	5.8	5.1	4.7	4.3		n

L Global Early BUFGE #s 3, 4, 7, and 8 to IOB Clock

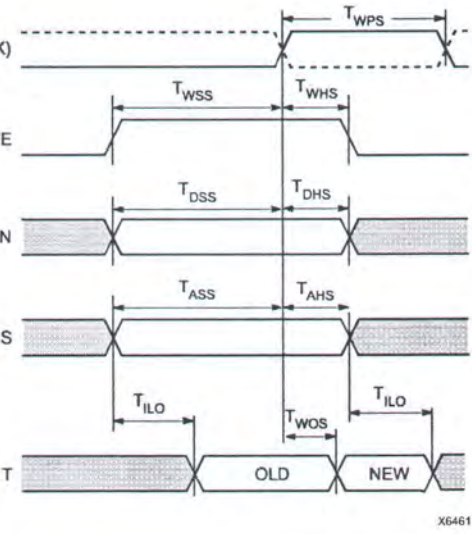
		Speed Grade	All	-3	-2	-1	-09	-08	Un
Description	Symbol	Device	Min	Max	Max	Max	Max	Max	
oad through GE buffer to any put.	T_{GE}	XC4002XL	0.5	2.8	2.5	2.1	1.7		n
		XC4005XL	0.7	3.1	2.8	2.7	2.5		n
		XC4010XL	0.7	3.5	3.1	2.8	2.7		n
		XC4013XL	0.7	3.8	3.3	2.9	2.8	2.4	n
		XC4020XL	0.8	4.1	3.6	3.4	3.2		n
		XC4028XL	0.9	4.4	3.9	3.4	3.3		n
		XC4036XL	0.9	4.7	4.2	3.7	3.6	3.1	n
		XC4044XL	1.0	5.1	4.5	4.0	3.7		n
		XC4052XL	1.1	5.5	4.8	4.3	4.3		n
		XC4062XL	1.2	5.9	5.2	4.8	4.5	4.0	n
		XC4085XL	1.3	6.8	6.0	5.5	5.2		n

SR/H0 via H to X/Y outputs	T_{HH0O}		2.5		2.2		2.0		1.8		
H1 via H to X/Y outputs	T_{HH1O}		2.4		2.1		1.9		1.6		
DIN/H2 via H to X/Y outputs	T_{HH2O}		2.5		2.2		2.0		1.8		
EC, DIN/H2 to YQ, XQ output (bypass)	T_{CBYP}		1.5		1.3		1.1		1.0		
rry Logic											
ts (F1, F2, G1, G4) to C_{OUT}	T_{OPCY}		2.7		2.3		2.0		1.6		
input (F3) to C_{OUT}	T_{ASCY}		3.3		2.9		2.5		1.8		
nputs (F1, F3) to C_{OUT}	T_{INCY}		2.0		1.8		1.5		1.0		
unction generators to X/Y outputs	T_{SUM}		2.8		2.6		2.4		1.7		
bypass function generators	T_{BYP}		0.26		0.23		0.20		0.14		
lay, C_{OUT} to C_{IN}	T_{NET}		0.32		0.28		0.25		0.24		
Delays											
p-Flop outputs Q	T_{CKO}		2.1		1.9		1.6		1.5		
tch outputs Q	T_{CKLO}		2.1		1.9		1.6		1.5		
before Clock K											
	T_{ICK}	1.1		1.0		0.9		0.8		0.8	
a H	T_{IHCK}	2.2		1.9		1.7		1.6		1.5	
H0 through H	T_{HH0CK}	2.0		1.7		1.6		1.4		1.4	
H1 through H	T_{HH1CK}	1.9		1.6		1.4		1.2		1.1	
H2 through H	T_{HH2CK}	2.0		1.7		1.6		1.4		1.4	
DIN	T_{DICK}	0.9		0.8		0.7		0.6		0.6	
EC	T_{ECCK}	1.0		0.9		0.8		0.7		0.7	
S/R, going Low (inactive)	T_{RCK}	0.6		0.5		0.5		0.4		0.4	
F/G	T_{CCK}	2.3		2.1		1.9		1.3		1.2	
F/G and H	T_{CHCK}	3.4		3.0		2.7		2.1		2.0	
fter Clock K											
	T_{CKI}	0		0		0		0		0	
a H	T_{CKIH}	0		0		0		0		0	
SR/H0 through H	T_{CKHH0}	0		0		0		0		0	
H1 through H	T_{CKHH1}	0		0		0		0		0	
DIN/H2 through H	T_{CKHH2}	0		0		0		0		0	
DIN/H2	T_{CKDI}	0		0		0		0		0	
EC	T_{CKEC}	0		0		0		0		0	
SR, going Low (inactive)	T_{CKR}	0		0		0		0		0	
ne	T_{CH}	3.0		2.8		2.5		2.3		2.1	
ne	T_{CL}	3.0		2.8		2.5		2.3		2.1	
rect											
	T_{RPW}	3.0		2.8		2.5		2.3		2.3	
inputs via S/R, going High to Q	T_{RIO}		3.7		3.2		2.8		2.7		
eset											
SR Pulse Width	T_{MRW}		19.8		17.3		15.0		14.0		
GSR input to any Q	T_{MRQ}	See page 14 for T_{RRI} values per device.									
uency (MHz) (for export control)	F_{TOG} (MHz)		166		179		200		217		

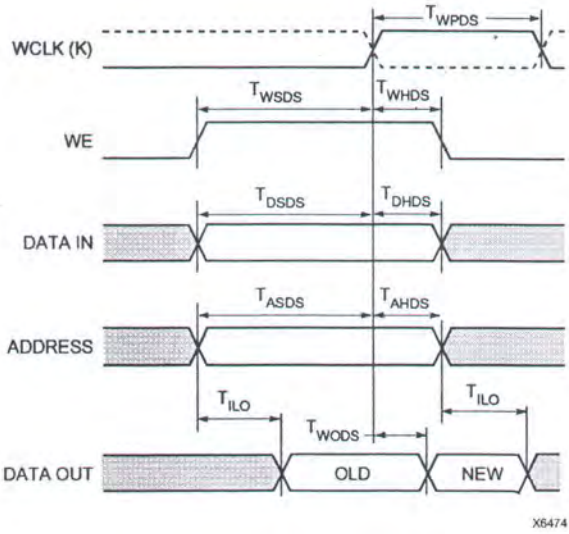
	32x1	T_{WCTS}	9.0		8.4		7.7		7.4		7.4
se width (active edge)	16x2	T_{WPS}	4.5		4.2		3.9		3.7		3.7
	32x1	T_{WPTS}	4.5		4.2		3.9		3.7		3.7
up time before clock K	16x2	T_{ASS}	2.2		2.0		1.7		1.7		1.6
	32x1	T_{ASTS}	2.2		2.0		1.7		1.7		1.7
d time after clock K	16x2	T_{AHS}	0		0		0		0		0
	32x1	T_{AHTS}	0		0		0		0		0
me before clock K	16x2	T_{DSS}	2.0		1.9		1.7		1.7		1.7
	32x1	T_{DSTS}	2.5		2.3		2.1		2.1		2.1
e after clock K	16x2	T_{DHS}	0		0		0		0		0
	32x1	T_{DHTS}	0		0		0		0		0
me before clock K	16x2	T_{WSS}	2.0		1.8		1.6		1.6		1.6
	32x1	T_{WSTS}	1.8		1.7		1.5		1.5		1.5
e after clock K	16x2	T_{WHS}	0		0		0		0		0
	32x1	T_{WHTS}	0		0		0		0		0
fter clock K	16x2	T_{WOS}		6.8		6.3		5.8		5.8	
	32x1	T_{WOTS}		8.1		7.5		6.9		6.7	
ation											
d cycle time	16x2	T_{RC}	4.5		3.1		2.6		2.6		2.6
	32x1	T_{RCT}	6.5		5.5		3.8		3.8		3.8
fter address change (no e)	16x2	T_{ILO}		1.6		1.5		1.3		1.2	
	32x1	T_{IHO}		2.7		2.4		2.2		2.0	
up time before clock K	16x2	T_{ICK}	1.1		1.0		0.9		0.8		0.8
	32x1	T_{IHCK}	2.2		1.9		1.7		1.6		1.5

se width (active edge)	16x1	T_{WPDS}	4.5		4.2		3.9		3.7		3.7
up time before clock K	16x1	T_{ASDS}	2.5		2.0		1.7		1.7		1.6
d time after clock K	16x1	T_{AHDS}	0		0		0		0		0
me before clock K	16x1	T_{DSDS}	2.5		2.3		2.0		2.0		2.0
ne after clock K	16x1	T_{DHDS}	0		0		0		0		0
me before clock K	16x1	T_{WSDS}	1.8		1.7		1.6		1.6		1.6
e after clock K	16x1	T_{WHDS}	0		0		0		0		0
fter clock K	16x1	T_{WODS}		7.8		7.3		6.7		6.7	

L CLB RAM Synchronous (Edge-Triggered) Write Timing



X6461



X6474

Skew Clock to Output us- ip Flop	T_{ICKOF}	XC4002XL	1.2	7.1	6.1	5.4	5.1		n
		XC4005XL	1.3	7.7	6.6	5.8	5.4		n
		XC4010XL	1.4	8.2	7.1	6.2	5.8		n
		XC4013XL	1.5	8.6	7.4	6.5	6.1	5.6	n
		XC4020XL	1.6	9.0	7.8	6.8	6.4		n
		XC4028XL	1.8	9.4	8.1	7.1	6.7		n
		XC4036XL	2.0	9.8	8.5	7.4	7.0	6.4	n
		XC4044XL	2.1	10.3	8.9	7.8	7.4		n
		XC4052XL	2.2	10.7	9.3	8.3	7.9		n
		XC4062XL	2.3	11.3	9.7	8.5	8.1	7.3	n
		XC4085XL	2.5	12.2	10.5	9.5	9.0		n
LOW option add	T_{SLOW}	All Devices	0.5	3.0	2.5	2.0	1.7	1.6	n

-to-out minimum delay is measured with the fastest route and the lightest load, Clock-to-out maximum delay is asured using the farthest distance and a reference load of one clock pin (IK or OK) per IOB as well as driving all cessible CLB flip-flops. For designs with a smaller number of clock loads, the pad-to-IOB clock pin delay as determined the static timing analyzer (TRCE) can be added to the AC parameter Tokpof and used as a worst-case pin-to-pin ck-to-out delay for clocked outputs for FAST mode configurations.
t timing is measured at ~50% V_{CC} threshold with 50 pF external capacitive load. For different loads, see Figure 1.

e Load Factor

ws the relationship between I/O output delay acitance. It allows a user to adjust the speci-elay if the load capacitance is different than example, if the actual load capacitance is 2.5 ns to the specified delay. If the load capac-pF, subtract 0.8 ns from the specified output

sable over the specified operating conditions d temperature and is independent of the out-control.

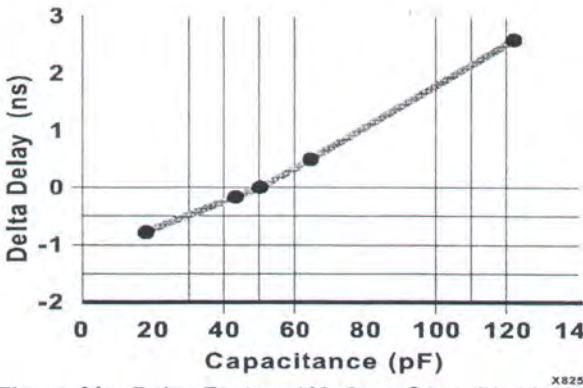


Figure 61: Delay Factor at Various Capacitive Loac

Clock to Output using Flop. Values are for BUF-5, and 6.	T_{ICKEOF}	XC4002XL	1.0	6.6	5.7	5.1	4.8		n
		XC4005XL	1.2	6.9	6.1	5.5	5.2		n
		XC4010XL	1.2	7.2	6.2	5.5	5.3		n
		XC4013XL	1.3	7.4	6.4	5.6	5.3	4.8	n
		XC4020XL	1.3	7.6	6.5	5.9	5.6		n
		XC4028XL	1.2	7.8	6.7	5.9	5.6		n
		XC4036XL	1.2	8.1	7.0	6.1	5.8	5.2	n
		XC4044XL	1.1	8.5	7.3	6.4	6.0		n
		XC4052XL	1.2	9.0	7.8	6.8	6.6		n
		XC4062XL	1.2	9.9	8.6	7.5	7.0	6.3	n
		XC4085XL	1.3	10.8	9.4	8.5	7.9		n

-to-out minimum delay is measured with the fastest route and the lightest load, Clock-to-out maximum delay is easured using the farthest distance and a reference load of one clock pin (IK or OK) per IOB as well as driving all cessible CLB flip-flops. For designs with a smaller number of clock loads, the pad-to-IOB clock pin delay as determined the static timing analyzer (TRCE) can be added to the AC parameter Tokpof and used as a worst-case pin-to-pin ock-to-out delay for clocked outputs for FAST mode configurations.
is measured at ~50% V_{CC} threshold with 50 pF external capacitive load. For different loads, see Figure 1.

. Output Flip-Flop, Clock to Out, BUFGE #s 3, 4, 7, and 8

		Speed Grade	All	-3	-2	-1	-09	-08	Uni
Description	Symbol	Device	Min	Max	Max	Max	Max	Max	
Clock to Output using Flop. Values are for BUF-7, and 8.	T_{ICKEOF}	XC4002XL	1.3	7.8	6.8	5.9	5.3		ns
		XC4005XL	1.5	8.1	7.1	6.5	6.1		ns
		XC4010XL	1.6	8.5	7.4	6.6	6.3		ns
		XC4013XL	1.6	8.8	7.6	6.7	6.4	5.7	ns
		XC4020XL	1.7	9.1	7.9	7.2	6.8		ns
		XC4028XL	1.7	9.4	8.2	7.2	6.9		ns
		XC4036XL	1.8	9.7	8.5	7.5	7.2	6.4	ns
		XC4044XL	1.9	10.1	8.8	7.8	7.3		ns
		XC4052XL	2.0	10.5	9.1	8.1	7.9		ns
		XC4062XL	2.0	10.9	9.5	8.6	8.1	7.3	ns
		XC4085XL	2.2	11.8	10.3	9.3	8.8		ns

-to-out minimum delay is measured with the fastest route and the lightest load, Clock-to-out maximum delay is easured using the farthest distance and a reference load of one clock pin (IK or OK) per IOB as well as driving all cessible CLB flip-flops. For designs with a smaller number of clock loads, the pad-to-IOB clock pin delay as determined the static timing analyzer (TRCE) can be added to the AC parameter Tokpof and used as a worst-case pin-to-pin ck-to-out delay for clocked outputs for FAST mode configurations.
it timing is measured at ~50% V_{CC} threshold with 50 pF external capacitive load. For different loads, see Figure 1.

	T_{PSN}/T_{PHN}	XC4002XL	2.5 / 1.5	2.2 / 1.3	1.9 / 1.2	1.7 / 1.0		n
Skew Clock and IFF		XC4005XL	1.2 / 2.6	1.1 / 2.2	0.9 / 2.0	0.8 / 1.7		n
Skew Clock and FCL		XC4010XL	1.2 / 3.0	1.1 / 2.6	0.9 / 2.3	0.8 / 2.0		n
		XC4013XL	1.2 / 3.2	1.1 / 2.8	0.9 / 2.4	0.8 / 2.1	0.8 / 2.1	n
		XC4020XL	1.2 / 3.7	1.1 / 3.2	0.9 / 2.8	0.8 / 2.4		n
		XC4028XL	1.2 / 4.4	1.1 / 3.8	0.9 / 3.3	0.8 / 2.9		n
		XC4036XL	1.2 / 5.5	1.1 / 4.8	0.9 / 4.1	0.8 / 3.6	0.8 / 3.6	n
		XC4044XL	1.2 / 5.8	1.1 / 5.0	0.9 / 4.4	0.8 / 3.8		n
		XC4052XL	1.2 / 7.1	1.1 / 6.2	0.9 / 5.4	0.8 / 4.7		n
		XC4062XL	1.2 / 7.0	1.1 / 6.1	0.9 / 5.3	0.8 / 4.6	0.8 / 4.6	n
		XC4085XL	1.2 / 9.4	1.1 / 8.2	0.9 / 7.1	0.8 / 6.2		n
y	T_{PSP}/T_{PHP}	XC4002XL	8.4 / 0.0	7.3 / 0.0	6.3 / 0.0	5.5 / 0.0		n
Skew Clock and IFF		XC4005XL	10. 5 / 0.0	9.1 / 0.0	7.9 / 0.0	6.9 / 0.0		n
Skew Clock and FCL		XC4010XL	11.1 / 0.0	9.7 / 0.0	8.4 / 0.0	7.3 / 0.0		n
		XC4013XL*	6.1 / 1.0	5.3 / 1.0	4.6 / 1.0	4.0 / 1.0	3.7 / 0.5	n
		XC4020XL	11.9 / 1.0	10.3 / 1.0	9.0 / 1.0	7.8 / 1.0		n
		XC4028XL	12.3 / 1.0	10.7 / 1.0	9.3 / 1.0	8.1 / 1.0		n
		XC4036XL*	6.4 / 1.0	5.6 / 1.0	4.8 / 1.0	4.2 / 1.0	4.0/ 0.8	n
		XC4044XL	13.1 / 1.0	11.4 / 1.0	9.9 / 1.0	8.6 / 1.0		n
		XC4052XL	11.9 / 1.0	10.3 / 1.0	9.0 / 1.0	7.8 / 1.0		n
		XC4062XL*	6.7 / 1.2	5.8 / 1.2	5.1 / 1.2	4.4 / 1.2	4.2/ 1.0	n
		XC4085XL	12.9 / 1.2	11.2 / 1.2	9.8 / 1.2	8.5 / 1.2		n
	T_{PSD}/T_{PHD}	XC4002XL	6.8 / 0.0	6.0 / 0.0	5.2 / 0.0	4.5 / 0.0		n
Skew Clock and IFF		XC4005XL	8.8 / 0.0	7.6 / 0.0	6.6 / 0.0	5.6 / 0.0		n
		XC4010XL	9.0 / 0.0	7.8 / 0.0	6.8 / 0.0	5.8 / 0.0		n
		XC4013XL*	6.4 / 0.0	6.0 / 0.0	5.6 / 0.0	4.8 / 0.0	4.8 / 0.0	n
		XC4020XL	8.8 / 0.0	7.6 / 0.0	6.6 / 0.0	6.2 / 0.0		n
		XC4028XL	9.3 / 0.0	8.1 / 0.0	7.0 / 0.0	6.4 / 0.0		n
		XC4036XL*	6.6 / 0.0	6.2 / 0.0	5.8 / 0.0	5.3 / 0.0	5.3 / 0.0	n
		XC4044XL	10.6 / 0.0	9.2 / 0.0	8.0 / 0.0	6.8 / 0.0		n
		XC4052XL	11.2 / 0.0	9.7 / 0.0	8.4 / 0.0	7.0 / 0.0		n
		XC4062XL*	6.8 / 0.0	6.4 / 0.0	6.0 / 0.0	5.5 / 0.0	5.5 / 0.0	n
		XC4085XL	12.7 / 0.0	11.0 / 0.0	9.6 / 0.0	8.4 / 0.0		n

p-Flop or Latch

setup time is measured with the fastest route and the lightest load.

hold time is measured using the furthest distance and a reference load of one clock pin per IOB as well as driving all cessible CLB flip-flops. For designs with a smaller number of clock loads, the pad-to-IOB clock pin delay as determined the static timing analyzer (TRCE) can be used as a worst-case pin-to-pin no-delay input hold specification.

XC4013XL, XC4036XL, and 4062XL have significantly faster partial and full delay setup times than other devices.

		XC4002XL	2.8 / 1.5	2.5 / 1.3	2.2 / 1.2	1.9 / 1.0	
y Clock and IFF	T_{PSEN}/T_{PHEN}	XC4005XL	1.2 / 4.1	1.1 / 3.6	0.9 / 3.1	0.8 / 2.7	
y Clock and	T_{PFSEN}/T_{PFHEN}	XC4010XL	1.2 / 4.4	1.1 / 3.8	0.9 / 3.3	0.8 / 2.9	
		XC4013XL	1.2 / 4.7	1.1 / 4.1	0.9 / 3.6	0.8 / 3.1	0.5 / 2.7
		XC4020XL	1.2 / 4.6	1.1 / 4.0	0.9 / 3.5	0.8 / 3.0	
		XC4028XL	1.2 / 5.3	1.1 / 4.6	0.9 / 4.0	0.8 / 3.5	
		XC4036XL	1.2 / 6.7	1.1 / 5.8	0.9 / 5.1	0.8 / 4.4	0.5 / 3.7
		XC4044XL	1.2 / 6.5	1.1 / 5.7	0.9 / 4.9	0.8 / 4.3	
		XC4052XL	1.2 / 6.7	1.1 / 5.8	0.9 / 5.1	0.8 / 4.4	
		XC4062XL	1.2 / 8.4	1.1 / 7.3	0.9 / 6.3	0.8 / 5.5	0.5 / 4.7
		XC4085XL	1.2 / 8.7	1.1 / 7.5	0.9 / 6.6	0.8 / 5.7	
ıy		XC4002XL	8.1 / 0.9	7.0 / 0.8	6.1 / 0.7	5.3 / 0.6	
y Clock and IFF	T_{PSEP}/T_{PHEP}	XC4005XL	9.0 / 0.0	8.5 / 0.0	8.0 / 0.0	7.5 / 0.0	
y Clock and	T_{PFSEP}/T_{PFHEP}	XC4010XL	11.9 / 0.0	10.4 / 0.0	9.0 / 0.0	8.0 / 0.0	
		XC4013XL*	6.4 / 0.0	5.9 / 0.0	5.4 / 0.0	4.9 / 0.0	4.4 / 0.0
		XC4020XL	10.8 / 0.0	10.3 / 0.0	9.8 / 0.0	9.0 / 0.0	
		XC4028XL	14.0 / 0.0	12.2 / 0.0	10.6 / 0.0	9.8 / 0.0	
		XC4036XL*	7.0 / 0.0	6.6 / 0.0	6.2 / 0.0	5.2 / 0.0	4.7 / 0.0
		XC4044XL	14.6 / 0.0	12.7 / 0.0	11.0 / 0.0	10.8 / 0.0	
		XC4052XL	16.4 / 0.0	14.3 / 0.0	12.4 / 0.0	11.4 / 0.0	
		XC4062XL*	9.0 / 0.8	8.6 / 0.8	8.2 / 0.8	7.0 / 0.8	6.3 / 0.5
		XC4085XL	16.7 / 0.0	14.5 / 0.0	12.6 / 0.0	11.6 / 0.0	
		XC4002XL	6.7 / 0.0	5.8 / 0.0	5.1 / 0.0	4.4 / 0.0	
y Clock and IFF	T_{PSED}/T_{PHED}	XC4005XL	10.8 / 0.0	9.4 / 0.0	8.2 / 0.0	7.1 / 0.0	
		XC4010XL	10.3 / 0.0	9.0 / 0.0	7.8 / 0.0	6.8 / 0.0	
		XC4013XL*	10.0 / 0.0	8.7 / 0.0	7.6 / 0.0	6.6 / 0.0	6.0 / 0.0
		XC4020XL	12.0 / 0.0	10.4 / 0.0	9.1 / 0.0	7.9 / 0.0	
		XC4028XL	12.6 / 0.0	11.0 / 0.0	9.5 / 0.0	8.3 / 0.0	
		XC4036XL*	12.2 / 0.0	10.6 / 0.0	9.2 / 0.0	8.0 / 0.0	7.2 / 0.0
		XC4044XL	13.8 / 0.0	12.0 / 0.0	10.5 / 0.0	9.1 / 0.0	
		XC4052XL	14.1 / 0.0	12.3 / 0.0	10.7 / 0.0	9.3 / 0.0	
		XC4062XL*	13.1 / 0.0	11.4 / 0.0	9.9 / 0.0	8.6 / 0.0	7.8 / 0.0
		XC4085XL	17.9 / 0.0	15.6 / 0.0	13.6 / 0.0	11.8 / 0.0	

ip-Flop or Latch, FCL = Fast Capture Latch

setup time is measured with the fastest route and the lightest load.

hold time is measured using the furthest distance and a reference load of one clock pin per IOB as well as driving all ccessible CLB flip-flops. For designs with a smaller number of clock loads, the pad-to-IOB clock pin delay as determined r the static timing analyzer (TRCE) can be used as a worst-case pin-to-pin no-delay input hold specification.

e XC4013XL, XC4036XL, and 4062XL have significantly faster partial and full delay setup times than other devices.

Clock and	T_{PSEN}/T_{PHEN}	XC4005XL	1.2 / 4.1	1.1 / 3.6	0.9 / 3.1	0.8 / 2.7		
	T_{PFSEN}/T_{PFHEN}	XC4010XL	1.2 / 4.4	1.1 / 3.8	0.9 / 3.3	0.8 / 2.9		
Clock and		XC4013XL	1.2 / 4.7	1.1 / 4.1	0.9 / 3.6	0.8 / 3.1	0.5 / 2.7	
		XC4020XL	1.2 / 4.6	1.1 / 4.0	0.9 / 3.5	0.8 / 3.0		
		XC4028XL	1.2 / 5.3	1.1 / 4.6	0.9 / 4.0	0.8 / 3.5		
		XC4036XL	1.2 / 6.7	1.1 / 5.8	0.9 / 5.1	0.8 / 4.4	0.5 / 3.7	
		XC4044XL	1.2 / 6.5	1.1 / 5.7	0.9 / 4.9	0.8 / 4.3		
		XC4052XL	1.2 / 6.7	1.1 / 5.8	0.9 / 5.1	0.8 / 4.4		
		XC4062XL	1.2 / 8.4	1.1 / 7.3	0.9 / 6.3	0.8 / 5.5	0.5 / 4.7	
		XC4085XL	1.2 / 8.7	1.1 / 7.5	0.9 / 6.6	0.8 / 5.7		
y		XC4002XL	7.3 / 1.5	6.4 / 1.3	5.5 / 1.2	4.8 / 1.0		
Clock and	T_{PSEP}/T_{PHEP}	XC4005XL	8.4 / 0.0	7.9 / 0.0	7.4 / 0.0	7.2 / 0.0		
	T_{PFSEP}/T_{PFHEP}	XC4010XL	10.3 / 0.0	9.0 / 0.0	7.8 / 0.0	7.4 / 0.0		
Clock and		XC4013XL*	5.4 / 0.0	4.9 / 0.0	4.4 / 0.0	4.3 / 0.0	4.0 / 0.0	
		XC4020XL	9.8 / 0.0	9.3 / 0.0	8.8 / 0.0	8.5 / 0.0		
		XC4028XL	12.7 / 0.0	11.0 / 0.0	9.6 / 0.0	9.3 / 0.0		
		XC4036XL*	6.4 / 0.8	5.9 / 0.8	5.4 / 0.8	5.0 / 0.8	4.6 / 0.2	
		XC4044XL	13.8 / 0.0	12.0 / 0.0	10.4 / 0.0	10.2 / 0.0		
		XC4052XL	14.5 / 0.0	12.7 / 0.0	11.0 / 0.0	10.7 / 0.0		
		XC4062XL*	8.4 / 1.5	7.9 / 1.5	7.4 / 1.5	6.8 / 1.5	6.2 / 0.0	
		XC4085XL	14.5 / 0.0	12.7 / 0.0	11.0 / 0.0	10.8 / 0.0		
		XC4002XL	5.9 / 0.0	5.2 / 0.0	4.5 / 0.0	3.9 / 0.0		
Clock and	T_{PSED}/T_{PHED}	XC4005XL	10.8 / 0.0	9.4 / 0.0	8.2 / 0.0	7.1 / 0.0		
		XC4010XL	10.3 / 0.0	9.0 / 0.0	7.8 / 0.0	6.8 / 0.0		
		XC4013XL*	10.0 / 0.0	8.7 / 0.0	7.6 / 0.0	6.6 / 0.0	6.0 / 0.0	
		XC4020XL	12.0 / 0.0	10.4 / 0.0	9.1 / 0.0	7.9 / 0.0		
		XC4028XL	12.6 / 0.0	11.0 / 0.0	9.5 / 0.0	8.3 / 0.0		
		XC4036XL*	12.2 / 0.0	10.6 / 0.0	9.2 / 0.0	8.0 / 0.0	7.2 / 0.0	
		XC4044XL	13.8 / 0.0	12.0 / 0.0	10.5 / 0.0	9.1 / 0.0		
		XC4052XL	14.1 / 0.0	12.3 / 0.0	10.7 / 0.0	9.3 / 0.0		
		XC4062XL*	13.1 / 0.0	11.4 / 0.0	9.9 / 0.0	8.6 / 0.0	7.8 / 0.0	
		XC4085XL	17.9 / 0.0	15.6 / 0.0	13.6 / 0.0	11.8 / 0.0		

p Flop or Latch. FCL = Fast Capture Latch

setup time is measured with the fastest route and the lightest load.

hold time is measured using the furthest distance and a reference load of one clock pin per IOB as well as driving all cessible CLB flip-flops. For designs with a smaller number of clock loads, the pad-to-IOB clock pin delay as determined the static timing analyzer (TRCE) can be used as a worst-case pin-to-pin no-delay input hold specification.

XC4013XL, XC4036XL, and 4062XL have significantly faster partial and full delay setup times than other devices.

clock (IK) active edge		XC4013, 36, 62XL	2.2	1.9	1.6	1.6	1.6	n
		Balance of Family	2.2	1.9	1.6	1.6		r
s								
k (IK), no delay	T_{PICK}	XC4002XL	2.6	2.3	2.0	2.0		n
		XC4013, 36, 62XL	1.7	1.5	1.3	1.3	1.2	n
		Balance of Family	1.7	1.5	1.3	1.3		n
k (IK), via transparent Fast ch, no delay	T_{PICKF}	XC4002XL	3.2	2.9	2.5	2.4		n
		XC4013, 36, 62XL	2.3	2.0	1.8	1.7	1.6	n
		Balance of Family	2.3	2.0	1.8	1.7		n
Capture Latch Enable (OK),	T_{POCK}	XC4013, 36, 62XL	1.2	1.0	0.9	0.9	0.9	n
		Balance of Family	1.2	1.0	0.9	0.9		n
es		All Devices	0	0	0	0	0	
Reset								
SR Pulse Width	T_{MRW}	All devices	19.8	17.3	15.0	14.0	14.0	n
Reset			Max	Max	Max	Max	Max	
GSR input to any Q	T_{RRI}*	XC4002XL	9.8	8.5	7.4	7.0		n
		XC4005XL	11.3	9.8	8.5	8.1		n
		XC4010XL	13.9	12.1	10.5	10.0		n
		XC4013XL	15.9	13.8	12.0	11.4	10.9	n
		XC4020XL	18.6	16.1	14.0	13.3		n
		XC4028XL	20.5	17.9	15.5	14.3		n
		XC4036XL	22.5	19.6	17.0	16.2	16.2	n
		XC4044XL	25.1	21.9	19.0	18.1		n
		XC4052XL	27.2	23.6	20.5	19.5		n
		XC4062XL	29.1	25.3	22.0	20.9	20.4	n
		XC4085XL	34.4	29.9	26.0	24.7		n

nimum Amount of Time to Assure Valid Data. IFF = Input Flip-Flop or Latch, FCL = Fast Capture Latch.

		XC4013, 36, 62XL	3.1	2.7	2.4	2.2	2.1
		Balance of Family	3.1	2.7	2.4	2.2	
2 via transparent FCL and in-	T_{PFLI}	X4002XL	5.4	4.7	4.1	3.9	
o delay		XC4013, 36, 62XL	3.7	3.3	2.8	2.7	2.5
		Balance of Family	3.7	3.3	2.8	2.7	
o I1, I2 (flip-flop)	T_{IKRI}	All devices	1.7	1.5	1.3	1.2	1.2
o I1, I2 (latch enable, active	T_{IKLI}	All devices	1.8	1.6	1.4	1.3	1.3
e (OK) active edge to I1, I2	T_{OKLI}	XC4002XL	5.2	4.6	4.0	3.8	
nsparent standard input latch)		XC4013, 36, 62XL	3.6	3.1	2.7	2.6	2.5
		Balance of Family	3.6	3.1	2.7	2.6	

ip-Flop or Latch, FCL = Fast Capture Latch

	T_{CH}	3.0		2.8		2.5		2.3		2.1	
	T_{CL}	3.0		2.8		2.5		2.3		2.1	
n Delays											
o Pad	T_{OKPOF}		5.0		4.3		3.8		3.5		3
o Pad	T_{OPF}		4.1		3.6		3.1		3.0		2
ad hi-Z (slew-rate independent)	T_{TSHZ}		4.0		3.5		3.0		2.9		2
ad active and valid	T_{TSONF}		4.4		3.8		3.3		3.3		3
o Pad via Fast Output MUX	T_{OFPF}		5.5		4.8		4.2		4.0		3
to Pad via Fast MUX	T_{OKFPF}		5.1		4.5		3.9		3.7		3
Hold Times											
o clock (OK) setup time	T_{OOK}	0.5		0.4		0.3		0.3		0.3	
o clock (OK) hold time	T_{OKO}	0.0		0.0		0.0		0.0		0.0	
e (EC) to clock (OK) setup time	T_{ECOK}	0.0		0.0		0.0		0.0		0.0	
e (EC) to clock (OK) hold time	T_{OKEC}	0.3		0.2		0.1		0.0		0.0	
Reset											
SR pulse width	T_{MRW}	19.8		17.3		15.0		14.0		14.0	
GSR input to any Pad	T_{RPO}*										
4002XL			14.3		12.5		10.9		10.3		
4005XL			15.9		13.8		12.0		11.4		
4010XL			18.5		16.1		14.0		13.3		
4013XL			20.5		17.8		15.5		14.7		14
4020XL			23.2		20.1		17.5		16.6		
4028XL			25.1		21.9		19.0		17.6		
4036XL			27.1		23.6		20.5		19.4		19
4044XL			29.7		25.9		22.5		21.4		
4052XL			31.7		27.6		24.0		22.8		
4062XL			33.7		29.3		25.5		24.2		23
4085XL			39.0		33.9		29.5		28.0		
Adjustment											
LOW option add	T_{SLOW}		3.0		2.5		2.0		1.7		1.

iming is measured at ~50% V_{CC} threshold, with 50 pF external capacitive loads.
nimum Amount of Time to Assure Valid Data.

P7	P99	P96	P138	E3	P154	P199	68
P8	P100	P97	P139	C1	P155	P200	71
-	-	-	P140	C2	P156	P201	74
-	-	-	P141	D3	P157	P202	77
P9	P1	P98	P142	B1	P158	P203	80
P10	P2	P99	P143	B2	P159	P204	83
P11	P3	P100	P144	C3	P160	P205	-
P12	P4	P1	P1	C4	P1	P2	-
P13	P5	P2	P2	B3	P2	P4	86
P14	P6	P3	P3	A1	P3	P5	89
-	-	-	P4	A2	P4	P6	92
-	-	-	P5	C5	P5	P7	95
P15	P7	P4	P6	B4	P6	P8	98
P16	P8	P5	P7	A3	P7	P9	101
-	-	-	P8	C6	P10	P14	-
-	-	-	P9	B5	P11	P15	104
-	-	-	P10	B6	P12	P16	107
P17	P9	P6	P11	A5	P13	P17	110
P18	P10	P7	P12	C7	P14	P18	113
-	-	-	P13	B7	P15	P21	116
-	P11	P8	P14	A6	P16	P22	119
P19	P12	P9	P15	A7	P17	P23	122
P20	P13	P10	P16	A8	P18	P24	125
P21	P14	P11	P17	C8	P19	P25	-
P22	P15	P12	P18	B8	P20	P26	-
P23	P16	P13	P19	C9	P21	P27	128
P24	P17	P14	P20	B9	P22	P28	131
-	P18	P15	P21	A9	P23	P29	134
-	-	-	P22	B10	P24	P30	137
P25	P19	P16	P23	C10	P25	P33	140
P26	P20	P17	P24	A10	P26	P34	143
-	-	-	P25	A11	P27	P35	146
-	-	-	P26	B11	P28	P36	149
-	-	-	P27	C11	P29	P37	-
P27	P21	P18	P28	B12	P32	P42	152
-	P22	P19	P29	A13	P33	P43	155
-	-	-	P30	A14	P34	P44	158
-	-	-	P31	C12	P35	P45	161
P28	P23	P20	P32	B13	P36	P46	164
P29	P24	P21	P33	B14	P37	P47	167
P30	P25	P22	P34	A15	P38	P48	170
P31	P26	P23	P35	C13	P39	P49	-
P32	P27	P24	P36	A16	P40	P50	173
P33	P28	P25	P37	C14	P41	P55	-
P34	P29	P26	P38	B15	P42	P56	174
P35	P30	P27	P39	B16	P43	P57	175
P36	P31	P28	P40	D14	P44	P58	178
-	-	-	P41	C15	P45	P59	181
-	-	-	P42	D15	P46	P60	184
-	P32	P29	P43	E14	P47	P61	187
P37	P33	P30	P44	C16	P48	P62	190

GND	P43	P41	P38	P55	J14	P61	P79	
I/O	P44	P42	P39	P56	J15	P62	P80	
I/O	P45	P43	P40	P57	J16	P63	P81	
I/O	-	P44	P41	P58	K16	P64	P82	
I/O	-	P45	P42	P59	K15	P65	P83	
I/O	P46	P46	P43	P60	K14	P66	P86	
I/O	P47	P47	P44	P61	L16	P67	P87	
I/O	-	-	-	P62	M16	P68	P88	
I/O	-	-	-	P63	L15	P69	P89	
GND	-	-	-	P64	L14	P70	P90	
I/O	P48	P48	P45	P65	P16	P73	P95	
I/O	P49	P49	P46	P66	M14	P74	P96	
I/O	-	-	-	P67	N15	P75	P97	
I/O	-	-	-	P68	P15	P76	P98	
I/O	P50	P50	P47	P69	N14	P77	P99	
I/O, SGCK3 †, GCK4 ††	P51	P51	P48	P70	R16	P78	P100	
GND	P52	P52	P49	P71	P14	P79	P101	
DONE	P53	P53	P50	P72	R15	P80	P103	
VCC	P54	P54	P51	P73	P13	P81	P106	
$\overline{\text{PROGRAM}}$	P55	P55	P52	P74	R14	P82	P108	
I/O (D7)	P56	P56	P53	P75	T16	P83	P109	
I/O, PGCK3†, GCK5††	P57	P57	P54	P76	T15	P84	P110	
I/O	-	-	-	P77	R13	P85	P111	
I/O	-	-	-	P78	P12	P86	P112	
I/O (D6)	P58	P58	P55	P79	T14	P87	P113	
I/O	-	P59	P56	P80	T13	P88	P114	
GND	-	-	-	P81	P11	P91	P119	
I/O	-	-	-	P82	R11	P92	P120	
I/O	-	-	-	P83	T11	P93	P121	
I/O (D5)	P59	P60	P57	P84	T10	P94	P122	
I/O ($\overline{\text{CS0}}$)	P60	P61	P58	P85	P10	P95	P123	
I/O	-	P62	P59	P86	R10	P96	P126	
I/O	-	P63	P60	P87	T9	P97	P127	
I/O (D4)	P61	P64	P61	P88	R9	P98	P128	
I/O	P62	P65	P62	P89	P9	P99	P129	
VCC	P63	P66	P63	P90	R8	P100	P130	
GND	P64	P67	P64	P91	P8	P101	P131	
I/O (D3)	P65	P68	P65	P92	T8	P102	P132	
I/O ($\overline{\text{RS}}$)	P66	P69	P66	P93	T7	P103	P133	
I/O	-	P70	P67	P94	T6	P104	P134	
I/O	-	-	-	P95	R7	P105	P135	
I/O (D2)	P67	P71	P68	P96	P7	P106	P138	
I/O	P68	P72	P69	P97	T5	P107	P139	
I/O	-	-	-	P98	R6	P108	P140	
I/O	-	-	-	P99	T4	P109	P141	
GND	-	-	-	P100	P6	P110	P142	
I/O (D1)	P69	P73	P70	P101	T3	P113	P147	
I/O ($\overline{\text{RCLK}}$, RDY/$\overline{\text{BUSY}}$)	P70	P74	P71	P102	P5	P114	P148	
I/O	-	-	-	P103	R4	P115	P149	
I/O	-	-	-	P104	R3	P116	P150	
I/O (D0, DIN)	P71	P75	P72	P105	P4	P117	P151	

-	-	-	P119	L2	P132	P172	20
-	-	-	P120	L1	P133	P173	23
P81	P85	P82	P121	K3	P134	P174	26
P82	P86	P83	P122	K2	P135	P175	29
-	P87	P84	P123	K1	P137	P178	32
-	P88	P85	P124	J1	P138	P179	35
P83	P89	P86	P125	J2	P139	P180	38
P84	P90	P87	P126	J3	P140	P181	41
P1	P91	P88	P127	H2	P141	P182	-

nal XC4005E/XL Package Pins

Not Connected Pins				
-	-	-	-	-

PQ208

Not Connected Pins				
P1	P3	P10	P11	P12
P19	P20	P31	P32	P38
P40	P41	P51	P52	P53
P63	P64	P65	P66	P72
P84	P85	P91	P92	P93
P102	P104	P105	P107	P115
P117	P118	P124	P125	P136
P143	P144	P145	P146	P155
P157	P158	P167	P168	P169
P176	P177	P188	P189	P195
P197	P198	P206	P207	P208

6/5/97

160	61	77		113			129	145			160			160						
192					113		129		145		160		192	192						
224					113		129		145		160		193	205						
256						129				160		193		205	256	256	256			
288						129				160		193				256	288	288	288	
320						129				160		193				256	289	320	320	
352												193				256		352	352	
384												193				256			352	384
448																			352		448

ser I/O Chart for XC4000E FPGAs

Max I/O	Maximum User I/O by Package Type															
	PC84	PQ100	VQ100	PG120	TQ144	PG156	PQ160	PG191	HQ208	PQ208	PG223	BG225	HQ240	PQ240	PG299	
80	61	77	77	80												
112	61	77			112	112	112			112						
128	61				113	125	128			128						
144	61						129	144		144						
160	61						129	160	160	160		160				
192							129		160	160	192	192	192	192		
224									160		192		193			
256											192		193		256	

ser I/O Chart for XC4000EX FPGAs

Max I/O	Maximum User Accessible I/O by Package Type						
	HQ208	HQ240	PG299	HQ304	BG352	PG411	BG432
256	160	193	256	256	256		
288		193		256	288	288	288

Package Type

PC = Plastic Lead Chip Carrier
PQ = Plastic Quad Flat Pack
VQ = Very Thin Quad Flat Pack
TQ = Thin Quad Flat Pack

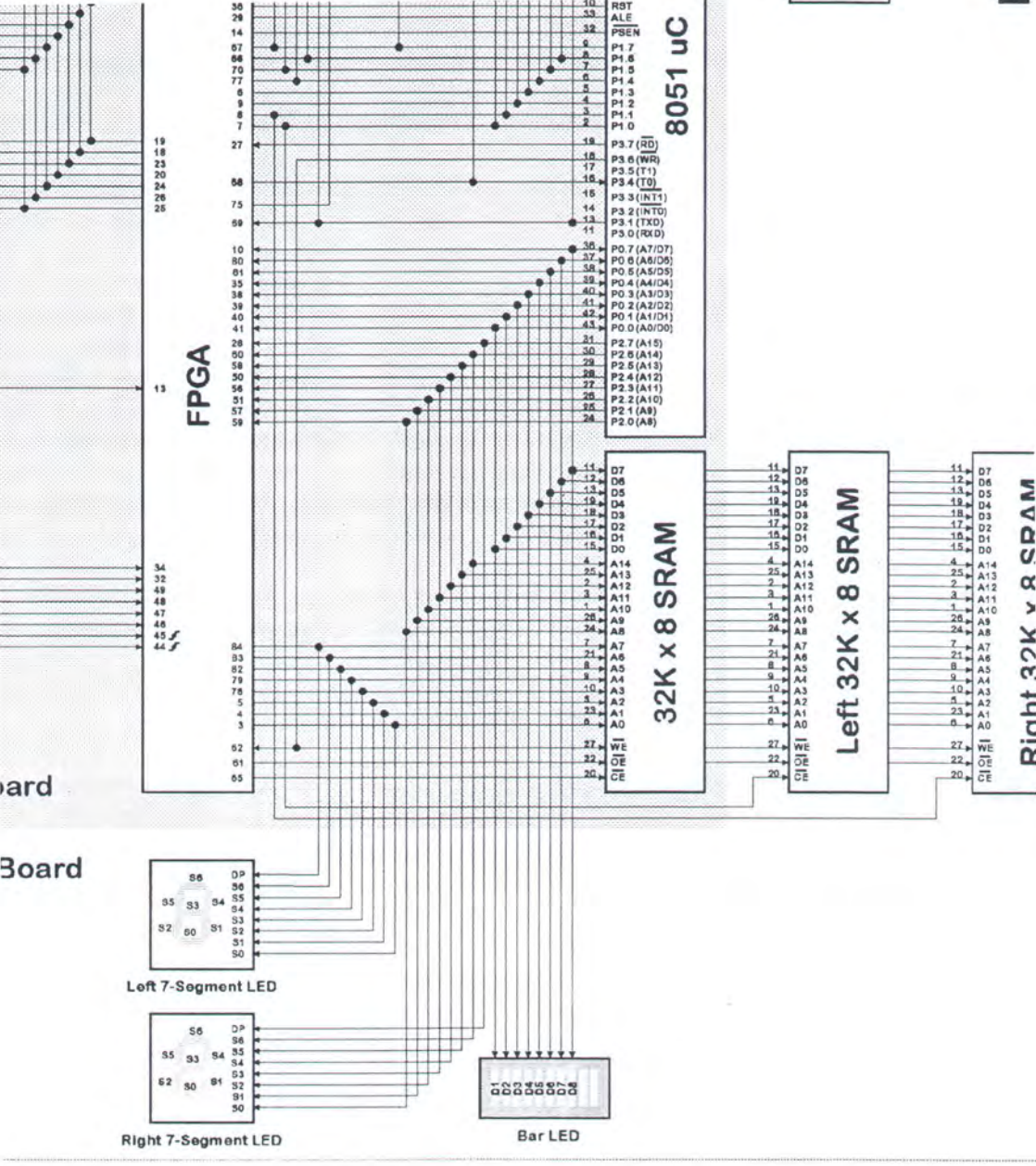
BG = Ball Grid Array
PG = Ceramic Pin Grid Array
HQ = High Heat Dissipation Quad Flat Pack
MQ = Metal Quad Flat Pack
CB = Top Brazed Ceramic Quad Flat Pack

X9020

n Control

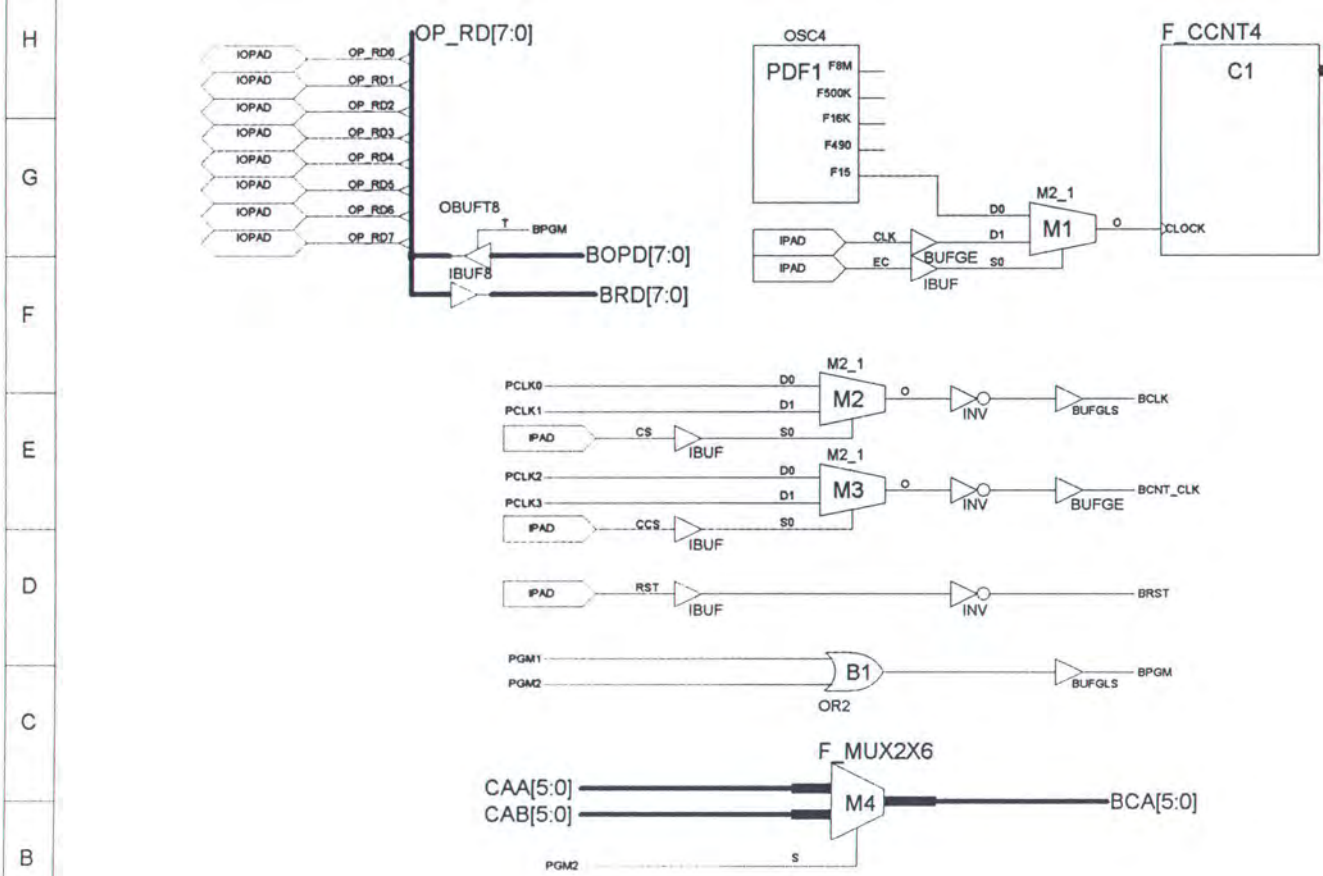
ion	Description
1998 (1.5)	Updated XC4000XL timing and added XC4002XL
/99	Updated pin diagrams.

	S5	RED1					XS Board LED segment; VGA color signal
	S6	HSYNCB					XS Board LED segment; VGA horiz. sync.
	S3	GREEN1					XS Board LED segment; VGA color signal
	S4	RED0					XS Board LED segment; VGA color signal
	S2	GREEN0					XS Board LED segment; VGA color signal
	S0	BLUE0					XS Board LED segment; VGA color signal
	S1	BLUE1					XS Board LED segment; VGA color signal
					P3.7 (RD_)		uC read line
	RDPB				P2.7		Right LED decimal-point; uC I/O port
					ALEB		uC address-latch-enable
							Serial EEPROM chip-enable
						PC_D6	PC parallel port data output
						PC_D7	PC parallel port data output
	DB5		D4		P0.4		LED; RAM data line; uC muxed address/data line
					RST		uC reset
	RESETB				XTAL1		Pushbutton; uC clock
	DB4		D3		P0.3		LED; RAM data line; uC muxed address/data line
	DB3		D2		P0.2		LED; RAM data line; uC muxed address/data line
	DB2		D1		P0.1		LED; RAM data line; uC muxed address/data line
	DB1		D0		P0.0		LED; RAM data line; uC muxed address/data line
				CCLK		PC_D0	Codec control line; PC parallel port data output
				CDIN		PC_D1	Codec control line; PC parallel port data output
				CSB		PC_D2	Codec control line; PC parallel port data output
						PC_D3	PC parallel port data output
						PC_D4	PC parallel port data output
						PC_D5	PC parallel port data output
	RSB4		A12		P2.4		Right LED segment; RAM address line; uC I/O port
	RSB2		A10		P2.2		Right LED segment; RAM address line; uC I/O port
							Power supply ground
?3.3V							5V/3.3V power supply (4000E/4000XL)
	PROGRAM						XS40 configuration control
	RSB3		A11		P2.3		Right LED segment; RAM address line; uC I/O port
	RSB1		A9		P2.1		Right LED segment; RAM address line; uC I/O port
	RSB5		A13		P2.5		Right LED segment; RAM address line; uC I/O port
	RSB0		A8		P2.0		Right LED segment; RAM address line; uC I/O port
	RSB6		A14		P2.6		Right LED segment; RAM address line; uC I/O port
			OEB				RAM output-enable
			WEB		P3.6 (WR_)		RAM write-enable; uC I/O port
			CEB				XS Board RAM chip-enable
DIPSW7				LRCK	P1.6	PC_S5	DIP switch; codec left-right channel switch; uC I/O port; PC
	SPAREB	VSYNCB			P1.7		Pushbutton; VGA vert. sync.; uC I/O port
		KB_CLK			P3.4 (T0)		PS/2 keyboard clock; uC I/O port
DIPSW8		KB_DATA			P3.1 (TXD)	PC_S6	DIP switch; PS/2 keyboard serial data; uC I/O port; PC para
DIPSW6				SDIN	P1.5	PC_S3	DIP switch; codec serial input data; uC I/O port; PC parallel
							JTAG TDI; DIN
							JTAG TDO; DOUT
							JTAG TCK; CCLK
						PC_S7	JTAG TDO; DOUT; PC parallel port status input
DIPSW5				SCLK	P1.4	PC_S4	DIP switch; codec serial I/O clock; uC I/O port; PC parallel
	LSB3		A3				Left LED segment; RAM address line
	LSB4		A4				Left LED segment; RAM address line
	DB7		D6		P0.6		LED; RAM data line; uC muxed address/data line
	DB6		D5		P0.5		LED; RAM data line; uC muxed address/data line
	LSB5		A5				Left LED segment; RAM address line
	LSB6		A6				Left LED segment; RAM address line
	LDPB		A7				Left LED decimal-point; RAM address line

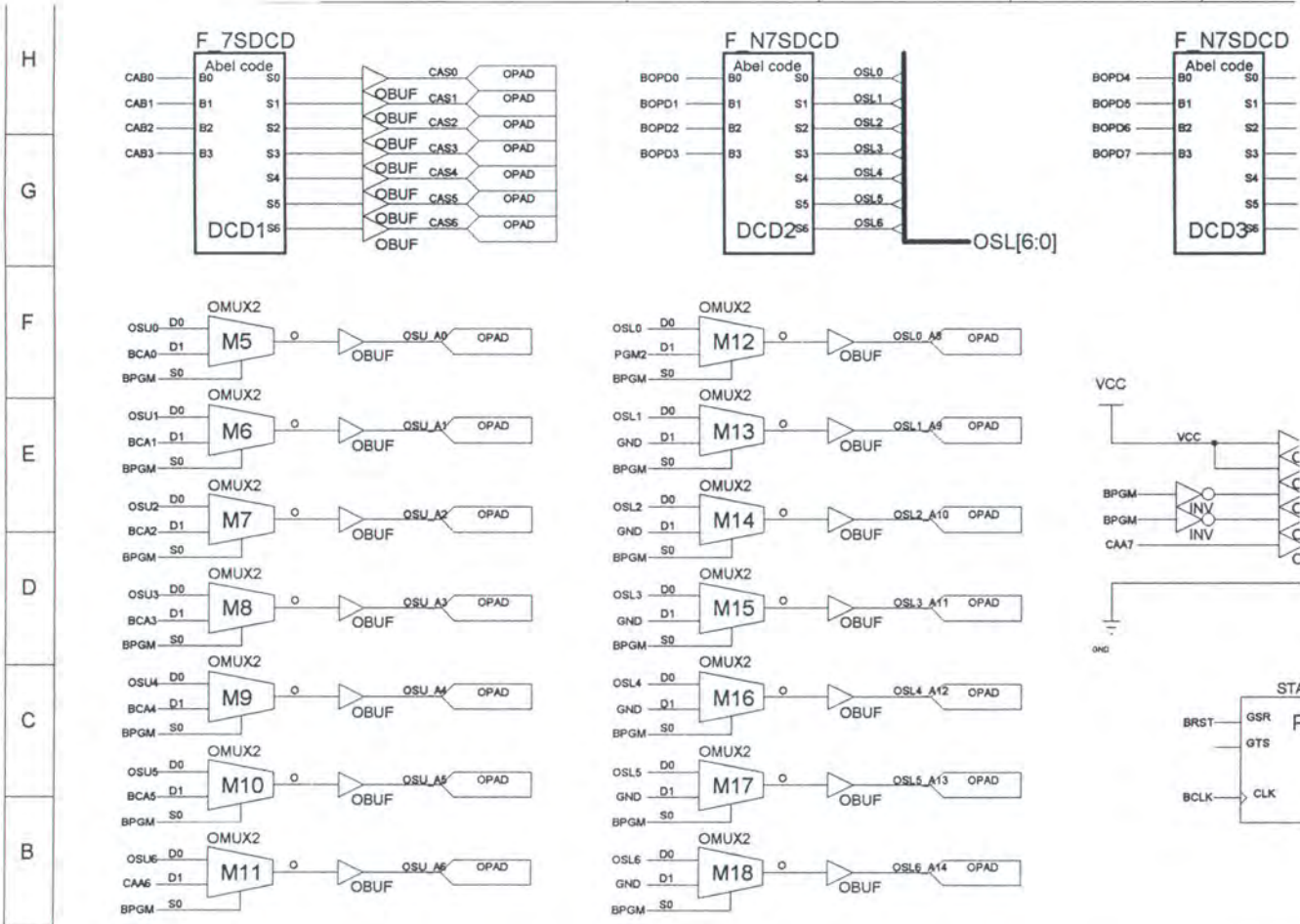

FPGA
8051 uC
32K x 8 SRAM
Left 32K x 8 SRAM
Right 32K x 8 SRAM
Board
Left 7-Segment LED
Right 7-Segment LED
Bar LED

```
----- Control Pins ------------------------------------
-- Master Reset Input (active LO) from XS-T Push Btn RESET
 RST          LOC=P37 ;
-- Program Flag Input (active HI) on XS-T DIP Sw 7 -------
 PGM          LOC=P66 ;
-- Enable(HI)/Disable(LO) Input Update on XS-T DIP Sw 8 --
 UPDT_IN      LOC=P69 ;

------ Interface Pins ----------------------------------
-- uC Reset, used to deactivate uC (active HI) ----------
 RST_UC       LOC=P36 ;
--- RAM's Write Enable (active LO) ----------------------
 WE_RAM       LOC=P62 ;
--- RAM's Output Enable (active LO) ---------------------
 OE_RAM       LOC=P61 ;
--- RAM's Chip Enable (active LO) -----------------------
 CE_RAM       LOC=P65 ;

------ 8 Bit Input Data Pins ---------------------------
Input Data bit 0-7 from PC Parallel Port D0-D7 ----------
 IPD<0>       LOC=P44 ;
 IPD<1>       LOC=P45 ;
 IPD<2>       LOC=P46 ;
 IPD<3>       LOC=P47 ;
 IPD<4>       LOC=P48 ;
 IPD<5>       LOC=P49 ;
 NET IPD<6>   LOC=P32 ;       # located on <MD0>
 NET IPD<7>   LOC=P34 ;       # located on <MD2>


========= O U T P U T  P I N S ==========================
------ 7-S Display of Lower Word I/O Address Pins --------
--- Segment 0-6 of XS-B 7-Segment ----------------------
 CAS0         LOC=P25 ;
 CAS1         LOC=P26 ;
 CAS2         LOC=P24 ;
 CAS3         LOC=P20 ;
 CAS4         LOC=P23 ;
 CAS5         LOC=P18 ;
 CAS6         LOC=P19 ;

------ I/O & RAM Address Pins --------------------------
I/O & RAM Address bit 7 ---------------------------------
```

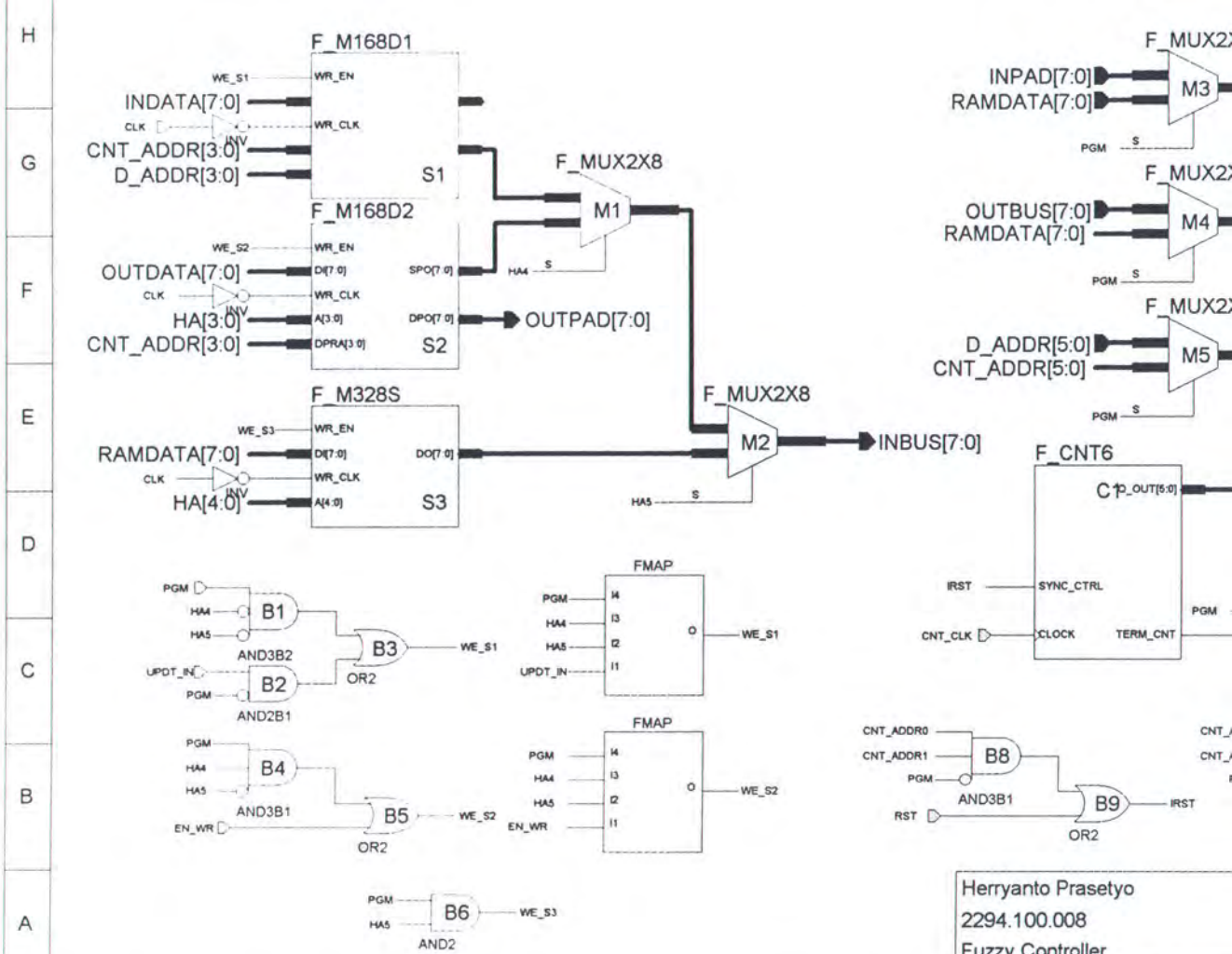
```
---- 7-S Disp of Lwr Word Out Pins / I/O & RAM Addr Pins -
- Segment 0-6 XS-T R-7-Segment / I/O & RAM Address bit 8-1
OSL0_A8      LOC=P59 ;
OSL1_A9      LOC=P57 ;
OSL2_A10     LOC=P51 ;
OSL3_A11     LOC=P56 ;
OSL4_A12     LOC=P50 ;
OSL5_A13     LOC=P58 ;
OSL6_A14     LOC=P60 ;

----- 7-S Disp of Upr Word Out Pins / I/O & RAM Addr Pins -
-- Segment 0-6 XS-T L-7-Segment / I/O & RAM Address bit 0-6
OSU_A0       LOC=P3 ;
OSU_A1       LOC=P4 ;
OSU_A2       LOC=P5 ;
OSU_A3       LOC=P78 ;
OSU_A4       LOC=P79 ;
OSU_A5       LOC=P82 ;
OSU_A6       LOC=P83 ;
```



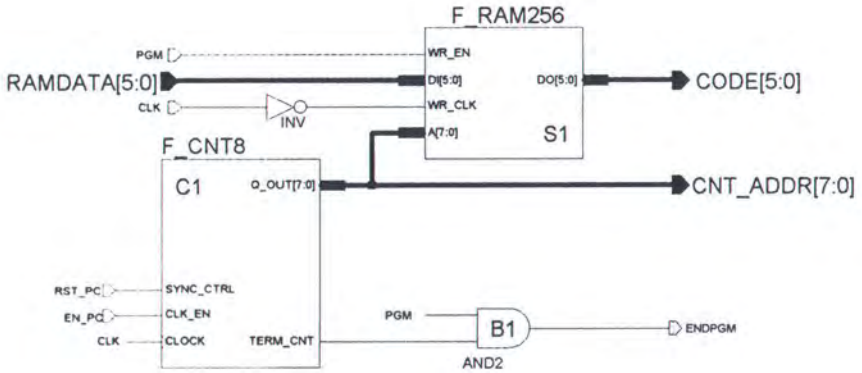
Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: Top-Level Schema	



Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: Top-Level Schema	

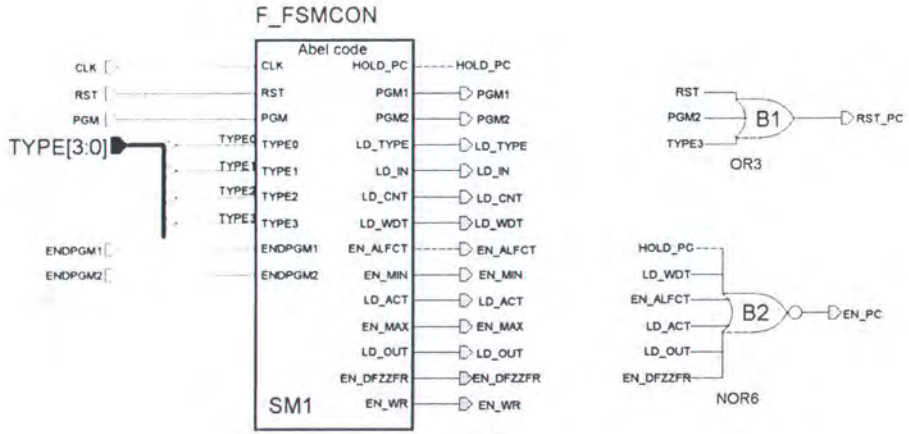


Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: I/O & Fix Paramet	

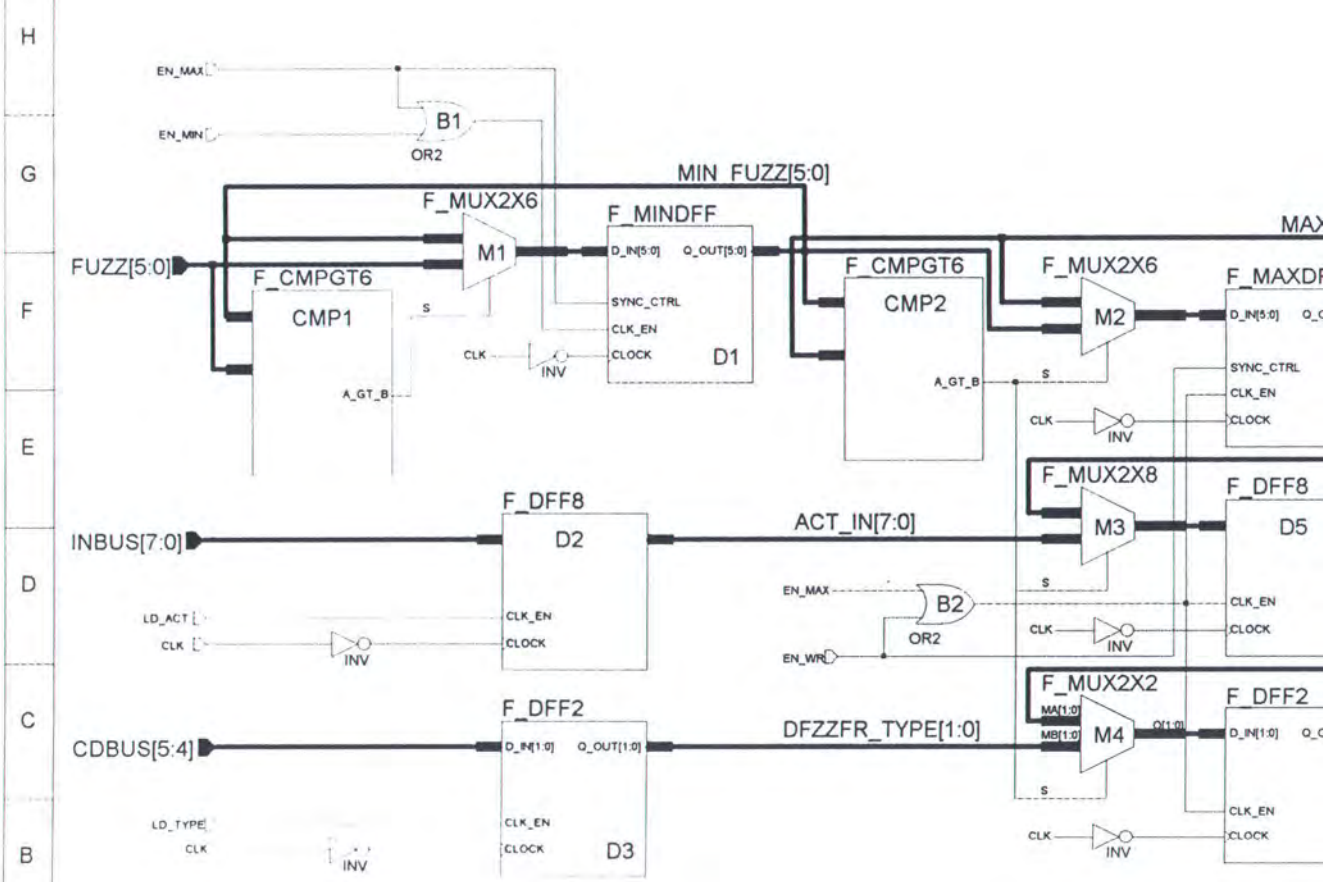


MILIK PERPUSTAKAAN ITS

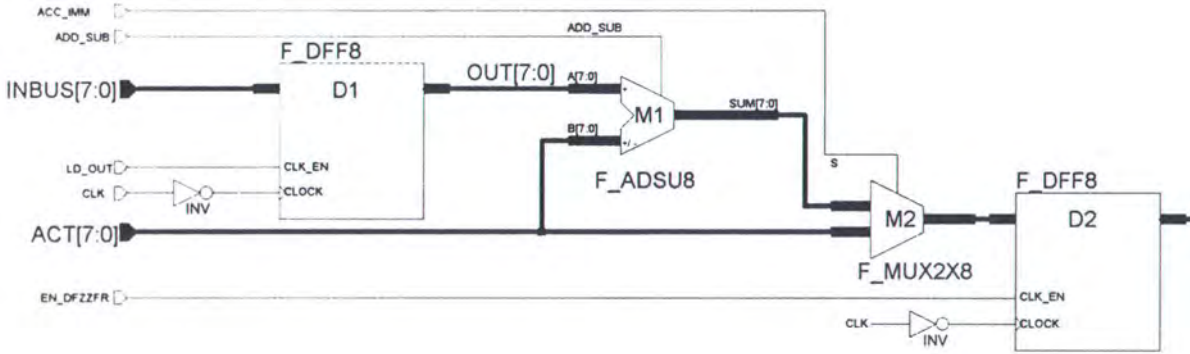
Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: Program Storage	



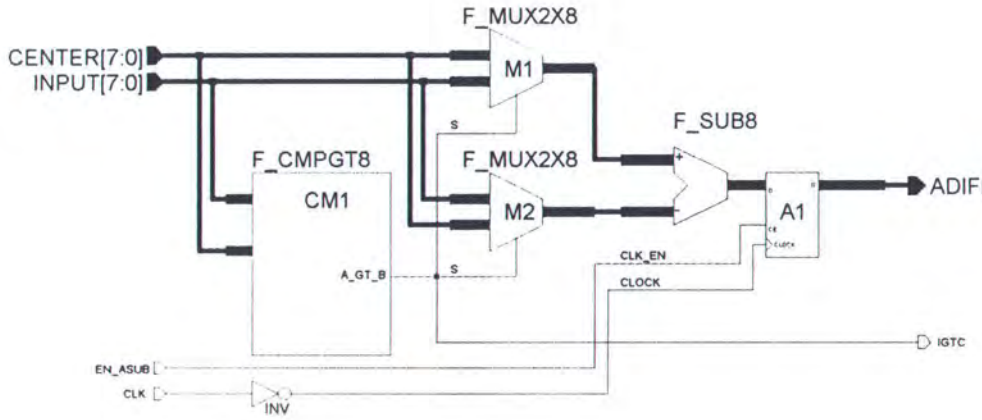
Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: FSM Logic Control	



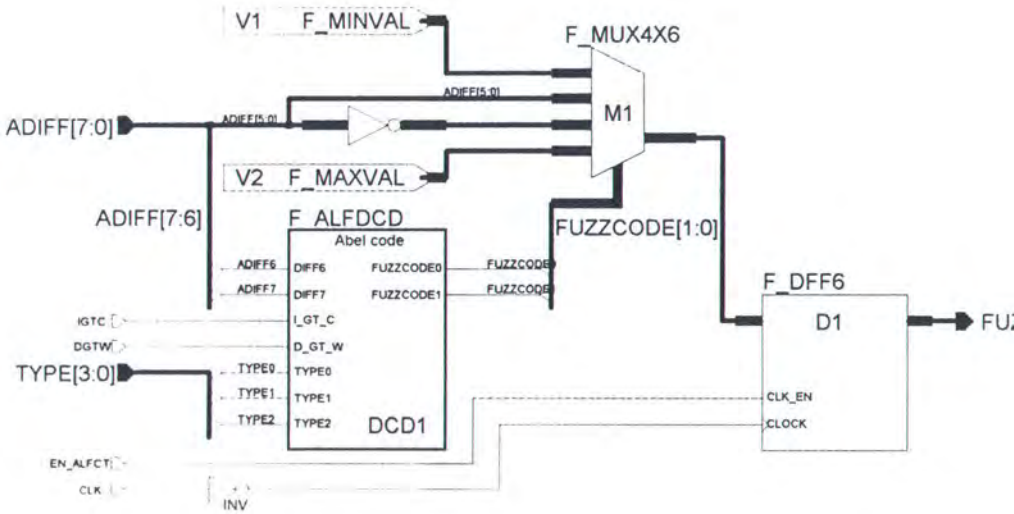
Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: Rule Evaluator Mo	



Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: Defuzzifier Module	



Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: 8 Bit Registerd Ab	



Herryanto Prasetyo	
2294.100.008	
Fuzzy Controller	
Description: Alfa Cut Calculato	

```
 ---------- output ports ----------------------------------
ld_PC    PIN istype 'com'; // Used to hold PC and Set Min DFF
m1       PIN istype 'com'; // Rule programming
m2       PIN istype 'com'; // Parameter programming
_Type    PIN istype 'com'; // Type FF's enable signal
_In      PIN istype 'com'; // Input FF's enable signal
_Cnt     PIN istype 'com'; // Center FF's enable signal
_Wdt     PIN istype 'com'; // Width FF's enable signal
_AlfCt   PIN istype 'com'; // Alfa Cut Calc. enable signal
_Min     PIN istype 'com'; // Minimum FF's enable signal
_Act     PIN istype 'com'; // Action Val FF's enable signal
_Max     PIN istype 'com'; // Maximum FF's enable signal
_Out     PIN istype 'com'; // Output FF's enable signal
_Dfzzfr  PIN istype 'com'; // Defuzzifier FF's enable signal
_Wr      PIN istype 'com'; // Enable write signal

 ---------- node for state machine code's registers ------
..R0  NODE istype 'reg'; // State Machine's code registers

 ---------- sets ----------------------------------------
ype  = [Type3..Type0];    // set of Type
     = [R3..R0];          // set of R

eclarations

 ---------- USER DEFINED ENCODED state machine: Sreg0 ----
reg0 = R;
       St_Init       = 0;
       St_Pgm1       = 1;
       St_Pgm2       = 2;
       St_Fetch      = 3;
       St_Ld_In      = 4;
       St_Ld_Cnt     = 5;
       St_Ld_Wdt     = 6;
       St_En_AlfCt   = 7;
       St_En_Min     = 8;
       St_Ld_Act     = 9;
       St_En_Max     = 10;
       St_Ld_Out     = 11;
       St_En_Dfzzfr  = 12;
       St_Wr_Out     = 13;
```

```
/ ---------- Programming State, filling RAMs -------------
tate St_Pgm1:                        // Filling Program Storage
     Pgm1 = 1;
     IF (EndPgm1) THEN St_Pgm2 // If finished, continue to..
     ELSE St_Pgm1;                   // fill Parameter Storage

tate St_Pgm2:                         // Filling Parameter Storage
     Pgm2 = 1;
     IF (EndPgm2) THEN St_Fetch // If finished, continue to.
     ELSE St_Pgm2;                   // start fetching code

// ---------- Fetching Code State, filling Type's FF -------
tate St_Fetch:                          // Fetching opr's type
     Ld_Type = 1;
     IF (Type == ^b0000) THEN       // End of rules reffering
        St_Ld_Out                     // specific output
     ELSE IF (Type == ^b0001) THEN // End of rule
        St_Ld_Act
     ELSE IF (Type == ^b1000) THEN // End of rules, reset PC
        St_Fetch                      // fetching first rule
     ELSE St_Ld_In;                   // Evaluating term

// ---------- Evaluating Terms State ----------------------
State St_Ld_In:          // Load Input value to Input's FF
     Ld_In = 1;
     GoTo St_Ld_Cnt;     // Continue with Center value

State St_Ld_Cnt: // Load Center value to Center's FF..
     Ld_Cnt = 1;// calculating absolute difference
     GoTo St_Ld_Wdt;     // Continue with Width value

State St_Ld_Wdt: // Load Width value and enable ASUB8.
     Ld_Wdt = 1;// calculating Fuzzy value of term
     GoTo St_En_AlfCt;   // Continue with enabling Alfa Cut

State St_En_AlfCt:       // Enable Alfa Cut Calculator, .....
     En_AlfCt = 1;       // choosing minimum value in Min Comp
     GoTo St_En_Min;     // Continue with enabling Min Comp

State St_En_Min: // Enable Minimum Comparator, .......
     En_Min = 1;// choosing maximum value in Max Comp
     GoTo St_Fetch;      // Go back to fetch state
```

```
State St_En_Dfzzfr:       // Enable Defuzzifier FF
     En_Dfzzfr = 1;
     GoTo St_Wr_Out;      // Continue to update output value

State St_Wr_Out: // Updating output in I/O Registers
     En_Wr = 1;
     GoTo St_Fetch;       // Back to fetch state & reset Max

// ---------- end of state machine - Sreg0 -----------------

end F_FSMCON
```

```
---------- Constant ----------
     = ^b010; // Symetrical Inclusive code
     = ^b011; // Symetrical Exclusive code
     = ^b100; // Left Inclusive code
     = ^b101; // Left Exclusive code
     = ^b110; // Right Inclusive code
     = ^b111; // Right Exclusive code
     = ^b00;  // MUX's code of minimum of fuzzy value
     = ^b01;          // MUX's code of difference value
F    = ^b10;          // MUX's code of inverted difference val
     = ^b11;          // MUX's code of maximum of fuzzy value

h_Table(     [Type, I_Gt_C, D_Gt_W, Diff7, Diff6] -> FuzzC
------------------- Symetrical Inclusive -------------------
               [ SI ,  .X.  ,   1   ,  .X. ,  .X. ] -> MIN;
               [ SI ,  .X.  ,   0   ,  .X. ,  .X. ] -> NDIFF;
------------------- Symetrical Exclusive -------------------
               [ SE ,  .X.  ,   1   ,   1  ,  .X. ] -> MAX;
               [ SE ,  .X.  ,   1   ,  .X. ,   1  ] -> MAX;
               [ SE ,  .X.  ,   1   ,   0  ,   0  ] -> DIFF;
               [ SE ,  .X.  ,   0   ,  .X. ,  .X. ] -> MIN;
------------------- Left Inclusive -------------------------
               [ LI ,   0   ,  .X.  ,  .X. ,  .X. ] -> MAX;
               [ LI ,   1   ,   1   ,  .X. ,  .X. ] -> MIN;
               [ LI ,   1   ,   0   ,  .X. ,  .X. ] -> NDIFF;
------------------- Left Exclusive -------------------------
               [ LE ,   0   ,  .X.  ,  .X. ,  .X. ] -> MIN;
               [ LE ,   1   ,   1   ,   1  ,  .X. ] -> MAX;
               [ LE ,   1   ,   1   ,  .X. ,   1  ] -> MAX;
               [ LE ,   1   ,   1   ,   0  ,   0  ] -> DIFF;
               [ LE ,   1   ,   0   ,  .X. ,  .X. ] -> MIN;
------------------- Right Inclusive ------------------------
               [ RI ,   0   ,   1   ,  .X. ,  .X. ] -> MIN;
               [ RI ,   0   ,   0   ,  .X. ,  .X. ] -> NDIFF;
               [ RI ,   1   ,  .X.  ,  .X. ,  .X. ] -> MAX;
------------------- Right Exclusive ------------------------
               [ RE ,   0   ,   1   ,   1  ,  .X. ] -> MAX;
               [ RE ,   0   ,   1   ,  .X. ,   1  ] -> MAX;
               [ RE ,   0   ,   1   ,   0  ,   0  ] -> DIFF;
               [ RE ,   0   ,   0   ,  .X. ,  .X. ] -> MIN;
               [ RE ,   1   ,  .X.  ,  .X. ,  .X. ] -> MIN;
 F_AlfDcd
```

```
5: Truth_Table(B -> S)
6:          0       -> ^b1110111; "0
7:          1       -> ^b0010010; "1
8:          2       -> ^b1011101; "2
9:          3       -> ^b1011011; "3
0:          4       -> ^b0111010; "4
1:          5       -> ^b1101011; "5
2:          6       -> ^b1101111; "6
3:          7       -> ^b1010010; "7
4:          8       -> ^b1111111; "8
5:          9       -> ^b1111011; "9
6:          10      -> ^b1111110; "A
7:          11      -> ^b0101111; "b
8:          12      -> ^b1100101; "C
9:          13      -> ^b0011111; "d
0:          14      -> ^b1101101; "E
1:          15      -> ^b1101100; "F
2:
3: end F_7SDcd
```

```
with_table(B -> S)
        0       -> ^b0001000; "0
        1       -> ^b1101101; "1
        2       -> ^b0100010; "2
        3       -> ^b0100100; "3
        4       -> ^b1000101; "4
        5       -> ^b0010100; "5
        6       -> ^b0010000; "6
        7       -> ^b0101101; "7
        8       -> ^b0000000; "8
        9       -> ^b0000100; "9
        10      -> ^b0000001; "A
        11      -> ^b1010000; "b
        12      -> ^b0011010; "C
        13      -> ^b1100000; "d
        14      -> ^b0010010; "E
        15      -> ^b0010011; "F

end F_n7SDcd
```

- FIDEData.h (6)
- FIDEData.cpp (11)
- FIDEParDlg.h (4)
- FIDEParDlg.cpp (18)
- FIDETrmV.h (2)
- FIDETrmV.cpp (6)
- FIDESIM.h (3)
- FIDESIM.cpp (9)
- FIDEDoc.h (2)
- FIDEDoc.cpp (12)
- FIDEView.h (2)
- FIDEView.cpp (8)
- FIDEFile.h (2)
- FIDEFile.cpp (6)

```
DtIO();                    // default constructor
ECLARE_SERIAL(CDtIO)
irtual ~CDtIO();           // default destructor

te:
String strName;     // input/output's name
YTE nInit;          // input/output's initial value

c:
/ Overloading operator '=' to copy CDtIO
DtIO& operator =(CDtIO &src);

/ Format CDtIO data in a string
oid Format(CString& str);

/ Used to save to / loading internal parameter from file
irtual void Serialize(CArchive& ar);

/ Get functions
String GetName() const { return strName; };
YTE GetInit() const    { return nInit;   };

/ Set functions
oid SetName(CString name) { strName = name; };
oid SetInit(BYTE init)    { nInit   = init; };



//////////////////////////////////////////////////////////////////
tTrm Class : store Term's Input, Label, Center, Width & Type
//////////////////////////////////////////////////////////////////
mbership function's codes :
I = 2,      Symmetrical Inclusive
E = 3,      Symmetrical Exclusive
I = 4,      Left Inclusive
E = 5,      Left Exclusive
I = 6,      Right Inclusive
E = 7       Right Exclusive        */

 CDtTrm : public CObject

c:
DtTrm();                   // default constructor
ECLARE_SERIAL(CDtTrm)
irtual ~CDtTrm();          // default destructor

te:
String strInput;    // input associated with term
String strName;     // term's name
String strCenter;   // center value
String strWidth;    // width value
YTE nType;          // Membership Function's type
```

```
CString GetWidth()  const { return strWidth; };
BYTE GetType()      const { return nType;    };

// Set functions
void SetInput(CString input)   { strInput  = input;  };
void SetName(CString name)     { strName   = name;   };
void SetCenter(CString center) { strCenter = center; };
void SetWidth(CString width)   { strWidth  = width;  };
void SetType(BYTE type)        { nType     = type;   };



//////////////////////////////////////////////////////////////////////
DtR1 Class : store Rule's Output, Action & Type
//////////////////////////////////////////////////////////////////////
efuzzification's types :
IMM = 0, Immediate          -> Output = Action Value
ACM = 1, Accumulate, Min    -> Output = Action Value - Output
ACP = 2  Accumulate, Plus   -> Output = Action Value + Output */

s CDtR1 : public CObject

lic:
CDtR1();                // default constructor
DECLARE_SERIAL(CDtR1)
virtual ~CDtR1();       // default destructor
CStringArray saTerms;   // competing terms

ote:
CString strOutput;  // output influenced
CString strAction;  // action value
BYTE nType;         // type of defuzzification process

lic:
// Overloading = to copy CDtR1
CDtR1& operator =(CDtR1 &src);

// Format CDtR1 data in a string
void Format(CString& str);

// Format Terms in a string
void FormatTerms(CString& str);

// Get defuzzification's label from the numeric code
CString GetDFZCode(BYTE type) const;

// Used to save to / loading from file
virtual void Serialize(CArchive& ar);

// Get functions
CString GetOutput()   const { return strOutput; };
CString GetAction()   const { return strAction; };
BYTE GetType()        const { return nType;     };
```

```
////////////////////////////////////////////////////////////////////
nstant values for rule's code
////////////////////////////////////////////////////////////////////
 int nEOR_CODE    = 1;
 int nEORRSO_CODE = 0;


////////////////////////////////////////////////////////////////////
nstant values for creating HEX file
////////////////////////////////////////////////////////////////////
 int nBASE_ADDRESS = 0;
 int nDATA_RECORD  = 0;
 int nINFO_RECORD  = 1;


////////////////////////////////////////////////////////////////////
tCnvrtd Class : store converted Fuzzy Parameters
////////////////////////////////////////////////////////////////////
 CDtCnvrtd : public CObject

c:
DtCnvrtd();
irtual ~CDtCnvrtd();
/ Converted Term Parameters
StringArray saTrmLbl;  // Term's Label for search usage
ByteArray baTrmTyp;    // Membership Function code (2 - 7)
ByteArray baTrmInp;    // Input Address (0 - 3 & 16 - 19 (feedback))
ByteArray baTrmCnt;    // Center Address (0 - 63, fixed / floating)
ByteArray baTrmWdt;    // Width Address (0 - 63, fixed / floating)

te:
String strLabel[nPARSUM];  // store parameters' label
YTE nValue[nPARSUM];       // store parameters' value
YTE nRule[nRULESUM];       // store converted rule

c:
/ Reset CDtCnvrtd data
oid ResetContents();

/ Find label in StrLabel, return index if success, -1 if fail
nt FindLabel(CString Label) const;

/ Find label in StrLabel, if fail, add one, if StrLabel is full, return -
nt FindOrAddLabel(CString Label);

/ Find term label in saTrmLbl, return index of -1 if fail
nt FindTrmLbl(CString TrmLbl) const;

/ Copy Term data to nRule, return TRUE if succeed, FALSE if fail
OOL CopyTerm(int TrmIdx, int &RleIdx);

/ Get functions
```

```
umeration for FSM State
////////////////////////////////////////////////////////////////////////
lef enum

St_Init      = 0, // initial state
St_Fetch     = 3, // fetching operating code
St_Ld_In     = 4, // load Input value
St_Ld_Cnt    = 5, // load Center value
St_Ld_Wdt    = 6, // load Width value
St_En_AlfCt  = 7, // enable Alfa Cut Calculator
St_En_Min    = 8, // enable Minimum Comparator
St_Ld_Act    = 9, // load Action value
St_En_Max    = 10, // enable Maximum Comparator
St_Ld_Out    = 11, // load Output value
St_En_Dfzzfr = 12, // enable Defuzzifier's regs
St_Wr_Out    = 13  // update Output value
ateType;


////////////////////////////////////////////////////////////////////////
onstant values for Alfa Cut Calculator
////////////////////////////////////////////////////////////////////////
: int nMIN_FUZZ = 0x00;
: int nMAX_FUZZ = 0x3F;


////////////////////////////////////////////////////////////////////////
mulation miscellanous
////////////////////////////////////////////////////////////////////////
: int nSIM_INC      = 50;   // increment size for array storage
: int nMASK_TYPE    = 0x0F; // 4 byte Type mask
: int nMASK_CS      = 0x40; // (Master) Clock Select Mask
: int nMASK_CCS     = 0x20; // Counter (I/O) Clock Select Mask
: int nMASK_UPDT_IN = 0x08; // Update Input Flag Mask


////////////////////////////////////////////////////////////////////////
tSIM Class : store simulation data
////////////////////////////////////////////////////////////////////////
: CDtSIM : public CObject

c:
DtSIM();
ECLARE_SERIAL(CDtSIM)
irtual ~CDtSIM();

te:
/ Misc. Data
nt i;                    // common index
tateType xPState;        // FSM's Previous State storage
tateType xState;         // FSM's Present State storage

/ Simulation Clocks
ByteArray baMCLK;        // Master clock
```

```
ByteArray baCAB;        // I/O Counter
ByteArray baCDBUS;      // Rule Code addressed by PC
ByteArray baINBUS;      // I/O & Fix Par Addressed by CDBUS
ByteArray baTYPE;       // Operation Type
ByteArray baINPUT;      // Input value for current Term
ByteArray baCENTER;     // Center value for current Term
ByteArray baWIDTH;      // Width value for current Term
ByteArray baFUZZ;       // Fuzzy value of current Term
ByteArray baMIN_FUZZ;   // Minimum Fuzzy value of competing Terms
ByteArray baMAX_FUZZ;   // Maximum Fuzzy value of competing Rules
ByteArray baACT;        // Action value of winning Rules
ByteArray baDTYPE;      // Defuzzification Type of winning Rules
ByteArray ba0UTBUS;     // Defuzzifier 0utput value

/ Simulation Miscellaneous
ByteArray baTrmCnt;     // Term number on the current Rule
ByteArray baRleCnt;     // Rule number
StringArray saComment;  // State comments

c:
/ FLC calculation procedures
/ --- used to calculate State progress
00L UpdateState(int idx);
/ --- used to calculate program counter
oid UpdateCAA(int idx);
/ --- used to calculate a term's fuzzy value
00L UpdateFUZZ(int idx);
/ --- used to calculate a rule's fuzzy value
oid UpdateMIN_FUZZ(int idx);
/ --- used to calculate winning rule's fuzzy value
oid UpdateMAX_FUZZ(int idx);
/ --- used to calculate an output value
00L Update0UTBUS(int idx);

/ Used to save to / loading simulation data from file
irtual void Serialize(CArchive& ar);

/ Get functions
nt GetInput(int idx, int Number);
nt Get0utput(int idx, int Number);
nt GetSize()                const { return baMCLK.GetSize(); };
nt GetIndex()               const { return i;                };
tateType GetState()         const { return xState;           };
tateType GetPState()        const { return xPState;          };
YTE GetCS(int idx)          const { return baCS[idx];        };
YTE GetCCS(int idx)         const { return baCCS[idx];       };
YTE GetUPDT_IN(int idx)     const { return baUPDT_IN[idx];   };
YTE GetMCLK(int idx)        const { return baMCLK[idx];      };
YTE GetI0CLK(int idx)       const { return baI0CLK[idx];     };
YTE GetCAA(int idx)         const { return baCAA[idx];       };
YTE GetCAB(int idx)         const { return baCAB[idx];       };
YTE GetCDBUS(int idx)       const { return baCDBUS[idx];     };
YTE GetINBUS(int idx)       const { return baINBUS[idx];     };
YTE GetTYPE(int idx)        const { return baTYPE[idx];      };
```

```
oid SetIndex(int idx)              { i              = idx;    };
oid SetState(StateType State)      { xState         = State;  };
oid SetPState(StateType PState)    { xPState        = PState; };
oid SetCS(BYTE CS)                 { baCS[i]        = CS;     };
oid SetCCS(BYTE CCS)               { baCCS[i]       = CCS;    };
oid SetUPDT_IN(BYTE UPDT_IN)       { baUPDT_IN[i]   = UPDT_IN;};
oid SetMCLK(BYTE MCLK)             { baMCLK[i]      = MCLK;   };
oid SetIOCLK(BYTE IOCLK)           { baIOCLK[i]     = IOCLK;  };
oid SetCAA(BYTE CAA)               { baCAA[i]       = CAA;    };
oid SetCAB(BYTE CAB)               { baCAB[i]       = CAB;    };
oid SetCDBUS(BYTE CDBUS)           { baCDBUS[i]     = CDBUS;  };
oid SetINBUS(BYTE INBUS)           { baINBUS[i]     = INBUS;  };
oid SetTYPE(BYTE TYPE)             { baTYPE[i]      = TYPE;   };
oid SetINPUT(BYTE INPUT)           { baINPUT[i]     = INPUT;  };
oid SetCENTER(BYTE CENTER)         { baCENTER[i]    = CENTER; };
oid SetWIDTH(BYTE WIDTH)           { baWIDTH[i]     = WIDTH;  };
oid SetFUZZ(BYTE FUZZ)             { baFUZZ[i]      = FUZZ;   };
oid SetMIN_FUZZ(BYTE MIN_FUZZ)     { baMIN_FUZZ[i]  = MIN_FUZZ;};
oid SetMAX_FUZZ(BYTE MAX_FUZZ)     { baMAX_FUZZ[i]  = MAX_FUZZ;};
oid SetACT(BYTE ACT)               { baACT[i]       = ACT;    };
oid SetDTYPE(BYTE DTYPE)           { baDTYPE[i]     = DTYPE;  };
oid SetOUTBUS(BYTE OUTBUS)         { baOUTBUS[i]    = OUTBUS; };
oid SetTrmCnt(BYTE TrmCnt)         { baTrmCnt[i]    = TrmCnt; };
oid SetRleCnt(BYTE RleCnt)         { baRleCnt[i]    = RleCnt; };
oid SetComment(CString Comment)    { saComment[i]   = Comment;};

f // !defined(AFX_FIDEDATA_H__E3BD56B3_95C8_11D2_9847_DF5D05B84D02__INCLU
```

```
IMENT_SERIAL(CDtI0, C0bject, 0)

////////////////////////////////////////////////////////////////////////
onstruction/Destruction
////////////////////////////////////////////////////////////////////////

0::CDtI0()

// there is nothing to do


0::~CDtI0()

// there is nothing to do


verloading operator '=' to copy CDtI0
0& CDtI0::operator =(CDtI0 &src)

this->strName = src.strName;
this->nInit = src.nInit;
return *this;


ormat CDtI0 data in a string
CDtI0::Format(CString& str)

str.Format(_T("%s  %i"), strName, nInit);


sed to save to / loading internal parameter from file
CDtI0::Serialize(CArchive& ar)

C0bject::Serialize(ar);
f (ar.IsStoring())
{
    ar << strName;
    ar << nInit;
}
lse
{
    ar >> strName;
    ar >> nInit;
}


////////////////////////////////////////////////////////////////////////
tTrm Class : store Term's Input, Label, Center, Width & Type
////////////////////////////////////////////////////////////////////////

MENT_SERIAL(CDtTrm, C0bject, 0)
```

```
his->strInput  = src.strInput;
his->strName   = src.strName;
his->strCenter = src.strCenter;
his->strWidth  = src.strWidth;
his->nType     = src.nType;
eturn *this;


rmat CDtTrm data in a string
CDtTrm::Format(CString& str)

tr.Format(_T("%s  IS  %s  %s  %s  %s"),
trInput, strName, strCenter, strWidth, GetMFCode(nType));


t membership function's label from the numeric code
ng CDtTrm::GetMFCode(BYTE type) const

witch (type)
{
ase 2 : return _T("Symmetrical Inclusive");
ase 3 : return _T("Symmetrical Exclusive");
ase 4 : return _T("Left Inclusive");
ase 5 : return _T("Left Exclusive");
ase 6 : return _T("Right Inclusive");
ase 7 : return _T("Right Exclusive");
efault :
   assert(true == false); // should never get there
}
eturn _T("Unknown");


ed to save to / loading from file
CDtTrm::Serialize(CArchive& ar)

Object::Serialize(ar);
f (ar.IsStoring())
{
   ar << strInput;
   ar << strName;
   ar << strCenter;
   ar << strWidth;
   ar << nType;
}
lse
{
   ar >> strInput;
   ar >> strName;
   ar >> strCenter;
   ar >> strWidth;
   ar >> nType;
```

```
// there is nothing to do


verloading = to copy CDtR1
1& CDtR1::operator =(CDtR1 &src)

his->saTerms.Copy(src.saTerms);
his->str0utput = src.str0utput;
his->strAction = src.strAction;
his->nType    = src.nType;
eturn *this;


ormat CDtR1 data in a string
 CDtR1::Format(CString& str)

String temp;
str.Format(_T("IF  %s"), saTerms.GetAt(0));
f (saTerms.GetSize())
    for (int i = 1; i < saTerms.GetSize(); i++)
    {
        temp.Format(_T("  AND  %s"), saTerms.GetAt(i));
        str += temp;
    }
emp.Format(_T(" THEN %s %s %s"), str0utput, strAction, GetDFZCode(nType));
str += temp;


ormat Terms in a string
 CDtR1::FormatTerms(CString& str)

String temp;
str.Format(_T("%s"), saTerms[0]);
f (saTerms.GetSize() > 1)
    for (int i = 1; i < saTerms.GetSize(); i++)
    {
        temp.Format(_T("  %s"), saTerms[i]);
        str += temp;
    }


et defuzzification's label from the numeric code
ng CDtR1::GetDFZCode(BYTE type) const

witch(type)
{
ase 0 : return _T("IMM");
ase 1 : return _T("ACM");
ase 2 : return _T("ACP");
efault :
    assert(true == false); // should never get there
}
eturn _T("Unknown");
```

```
    ar >> nType;


////////////////////////////////////////////////////////////////////////
tCnvrtd Class : store converted Fuzzy Parameters
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
nstruction/Destruction
////////////////////////////////////////////////////////////////////////

vrtd::CDtCnvrtd()

esetContents();


vrtd::~CDtCnvrtd()

aTrmLbl.RemoveAll();
aTrmTyp.RemoveAll();
aTrmInp.RemoveAll();
aTrmCnt.RemoveAll();
aTrmWdt.RemoveAll();


////////////////////////////////////////////////////////////////////////
blic Functions
////////////////////////////////////////////////////////////////////////

set CdtCnvrtd data
CDtCnvrtd::ResetContents()

or (int i = 0; i < nPARSUM; i++)

    if (i < 4)
        strLabel[i] = _T("INPUT");
    else if ((i > 15) && (i < 20))
        strLabel[i] = _T("Output");
    else
        strLabel[i] = _T("");
    nValue[i]   = 0;

or (i = 0; i < nRULESUM; i++)
    nRule[i] = 0;


nd label in StrLabel, return index if success, -1 if fail
)tCnvrtd::FindLabel(CString Label) const

abel.MakeUpper();
or (int i = 0; i < nPARSUM; i++)
    if (strLabel[i] == Label)
```

```
        else
        {
            strLabel[n] = Label; // add new label
            // new label should be value so add new value too
            nValue[n] = (BYTE) atoi((LPCTSTR) Label);
        }
    }
}
return n;


ind term label in saTrmLbl, return index of -1 if fail
CDtCnvrtd::FindTrmLbl(CString TrmLbl) const

TrmLbl.MakeUpper();
for (int i = 0; i < saTrmLbl.GetSize(); i++)
    if (saTrmLbl[i] == TrmLbl)
        return i;
return -1;


opy Term data to nRule, return TRUE if succeed, FALSE if fail
CDtCnvrtd::CopyTerm(int TrmIdx, int &RleIdx)

f (RleIdx+7 > nRULESUM) // rule overflow, added 4 for EOR & EORRS0 code
    return FALSE;
nRule[RleIdx++] = baTrmTyp[TrmIdx]; // Membership Function code
nRule[RleIdx++] = baTrmInp[TrmIdx]; // Input's address
nRule[RleIdx++] = baTrmCnt[TrmIdx]; // Center's address
nRule[RleIdx++] = baTrmWdt[TrmIdx]; // Width's address
return TRUE;



////////////////////////////////////////////////////////////////////////
DtSIM Class : store simulation data
////////////////////////////////////////////////////////////////////////

MENT_SERIAL(CDtSIM, CObject, 0)

////////////////////////////////////////////////////////////////////////
nstruction/Destruction
////////////////////////////////////////////////////////////////////////

M::CDtSIM()

    = 0;                // first data begin at 0
State = St_Init;        // initial state
etSize(100);            // initial simulation data's size


M::~CDtSIM()

aMCLK.RemoveAll();
```

```
baINPUT.RemoveAll();
baCENTER.RemoveAll();
baWIDTH.RemoveAll();
baFUZZ.RemoveAll();
baMIN_FUZZ.RemoveAll();
baMAX_FUZZ.RemoveAll();
baACT.RemoveAll();
baDTYPE.RemoveAll();
ba0UTBUS.RemoveAll();
baTrmCnt.RemoveAll();
baRleCnt.RemoveAll();
saComment.RemoveAll();


////////////////////////////////////////////////////////////////////////
ultiple-Line Get-Set Functions
////////////////////////////////////////////////////////////////////////
CDtSIM::GetInput(int idx, int Number)

switch(Number)
{
    case 0  : return baInput0[idx];
    case 1  : return baInput1[idx];
    case 2  : return baInput2[idx];
    default : return baInput3[idx];
}


CDtSIM::Get0utput(int idx, int Number)

switch(Number)
{
    case 0  : return ba0utput0[idx];
    case 1  : return ba0utput1[idx];
    case 2  : return ba0utput2[idx];
    default : return ba0utput3[idx];
}


CDtSIM::SetSize(int Size)

baMCLK.SetSize(Size);
baI0CLK.SetSize(Size);
baCS.SetSize(Size);
baCCS.SetSize(Size);
baUPDT_IN.SetSize(Size);
baInput0.SetSize(Size);
baInput1.SetSize(Size);
baInput2.SetSize(Size);
baInput3.SetSize(Size);
ba0utput0.SetSize(Size);
ba0utput1.SetSize(Size);
ba0utput2.SetSize(Size);
ba0utput3.SetSize(Size);
```

```
CDtSIM::SetInput(int Number, BYTE Value)

witch(Number)
{
    case 0 : baInput0[i] = Value; break;
    case 1 : baInput1[i] = Value; break;
    case 2 : baInput2[i] = Value; break;
    case 3 : baInput3[i] = Value; break;
}


CDtSIM::Set0utput(int Number, BYTE Value)

witch(Number)
{
    case 0 : ba0utput0[i] = Value; break;
    case 1 : ba0utput1[i] = Value; break;
    case 2 : ba0utput2[i] = Value; break;
    case 3 : ba0utput3[i] = Value; break;
}


////////////////////////////////////////////////////////////////////////
rtual Function 0verride
////////////////////////////////////////////////////////////////////////

ed to save to / loading simulation data from file
CDtSIM::Serialize(CArchive& ar)

0bject::Serialize(ar);

aMCLK.Serialize(ar);
aI0CLK.Serialize(ar);
aCS.Serialize(ar);
aCCS.Serialize(ar);
aUPDT_IN.Serialize(ar);
aInput0.Serialize(ar);
aInput1.Serialize(ar);
aInput2.Serialize(ar);
aInput3.Serialize(ar);
a0utput0.Serialize(ar);
a0utput1.Serialize(ar);
a0utput2.Serialize(ar);
a0utput3.Serialize(ar);
aCAA.Serialize(ar);
aCAB.Serialize(ar);
aCDBUS.Serialize(ar);
aINBUS.Serialize(ar);
aTYPE.Serialize(ar);
aINPUT.Serialize(ar);
aCENTER.Serialize(ar);
aWIDTH.Serialize(ar);
```

```
xPState = xState;
switch(xState)
{
    // --- Term Evaluating Stage
    case St_Ld_In      : xState = St_Ld_Cnt; break;
    case St_Ld_Cnt     : xState = St_Ld_Wdt; break;
    case St_Ld_Wdt     : xState = St_En_AlfCt; break;
    case St_En_AlfCt   : xState = St_En_Min; break;
    case St_En_Min     : xState = St_Fetch; break;

    // --- End Of Rule Stage
    case St_Ld_Act     : xState = St_En_Max; break;
    case St_En_Max     : xState = St_Fetch; break;

    // --- End Of Rule(s) Referring Specific Output Stage
    case St_Ld_Out     : xState = St_En_Dfzzfr; break;
    case St_En_Dfzzfr  : xState = St_Wr_Out; break;
    case St_Wr_Out     : xState = St_Fetch; break;

    // --- Fetching Stage
    case St_Init       : xState = St_Fetch; break;
    case St_Fetch      :
        switch(baTYPE[idx-1])
        {
            case 0  : xState = St_Ld_Out; break;
            case 1  : xState = St_Ld_Act; break;
            case 8  : xState = St_Fetch; break;
            default : xState = St_Ld_In;
        }
        break;
    // --- ERROR : unrecognized State
    default            : return FALSE;
}
return TRUE;


sed to calculate program counter
 CDtSIM::UpdateCAA(int idx)

// increment PC only on specific State
switch(xState)
{
    case St_Init       :
    case St_En_AlfCt   :
    case St_En_Min     :
    case St_En_Max     :
    case St_En_Dfzzfr  :
    case St_Wr_Out     : baCAA[idx] = baCAA[idx-1]; break;
    default            : baCAA[idx] = baCAA[idx-1]+1;
}
// EORs code / initial stage, reset PC
if (((xState == St_Fetch) && (baTYPE[idx-1] == 8)) ||
    (xPState == St_Init) || (baCAA[idx] == nRULESUM))
```

```
   nADIFF = baCENTER[idx] - baINPUT[idx];


/ Comparator, F_CMPGT8 in HDL design
/ --- difference greater than width flag
YTE nDGTW = (nADIFF > baWIDTH[idx])? 1:0;

/ Alfa Cut Decoder, F_ALFCUT in HDL design
/ --- 6 bit difference
YTE nDIFF  = nADIFF&nMAX_FUZZ;
/ --- complement of nADIFF
YTE nNDIFF = ~nDIFF&nMAX_FUZZ;
/ --- ADIFF greater than MAX_FUZZ flag
YTE nDGTF =  (nADIFF > nMAX_FUZZ)? 1:0;

witch(baTYPE[idx])

   // --- Symmetrical Inclusive Membership Function
   case 2  :
       baFUZZ[idx] = (nDGTW)? nMIN_FUZZ:nNDIFF;
       break;
   // --- Symmetrical Inclusive Membership Function
   case 3  :
       baFUZZ[idx] = (nDGTW)? nDIFF:nMIN_FUZZ;
       if ((nDGTW) && (nDGTF))
           baFUZZ[idx] = nMAX_FUZZ; // saturation value
       break;
   // --- Symmetrical Exclusive Membership Function
   case 4  :
       if (!nIGTC)
           baFUZZ[idx] = nMAX_FUZZ;
       else
           if (nDGTW)
               baFUZZ[idx] = nMIN_FUZZ;
           else
               baFUZZ[idx] = nNDIFF;
       break;
   // --- Left Inclusive Membership Function
   case 5  :
       if (!nIGTC)
           baFUZZ[idx] = nMIN_FUZZ;
       else
           if (nDGTW)
           {
               baFUZZ[idx] = nDIFF;
               if (nDGTF)
                   baFUZZ[idx] = nMAX_FUZZ; // saturation value
           }
           else
               baFUZZ[idx] = nMIN_FUZZ;
       break;
   // --- Left Exclusive Membership Function
   case 6  :
       if (nIGTC)
```

MILIK PERPUSTAKAAN
ITS

```
            }
            else
                baFUZZ[idx] = nMIN_FUZZ;
        break;
    // --- ERROR : unrecognized Membership Function
    default :
        baFUZZ[idx] = nMIN_FUZZ;
        return FALSE;

eturn TRUE;


ed to calculate a rule's fuzzy value
CDtSIM::UpdateMIN_FUZZ(int idx)

aMIN_FUZZ[idx] = (baFUZZ[idx] < baMIN_FUZZ[idx-1]) ?
    baFUZZ[idx] : baMIN_FUZZ[idx-1];


ed to calculate winning rule's fuzzy value
CDtSIM::UpdateMAX_FUZZ(int idx)

f (baMIN_FUZZ[idx] >= baMAX_FUZZ[idx-1])

    baMAX_FUZZ[idx] = baMIN_FUZZ[idx];
    baDTYPE[idx]    = baCDBUS[idx-4-4*baCS[idx]]>>4;
    baACT[idx]      = baINBUS[idx-2];

lse

    baMAX_FUZZ[idx] = baMAX_FUZZ[idx-1];
    baDTYPE[idx]    = baDTYPE[idx-1];
    baACT[idx]      = baACT[idx-1];


ed to calculate an output value
CDtSIM::UpdateOUTBUS(int idx)

witch(baDTYPE[idx])

    // --- IMM : Immediate          : Output = Action Value
    case 0  :
        baOUTBUS[idx] = baACT[idx];
        break;
    // --- ACM : Accumulate, Minus : Output = prev. Output - Action Value
    case 1  :
        baOUTBUS[idx] = baINBUS[idx] - baACT[idx];
        break;
    // --- ACP : Accumulate, Plus  : Output = prev. Output + Action Value
    case 3  :
        baOUTBUS[idx] = baINBUS[idx] + baACT[idx];
        break;
    // --- ERROR : unrecognized Defuzzification Type
```

MILIK PERPUST
ITS

```
nstruction
c:
Dlg Input();
~CDlgInput();
CTypedPtrArray<CObArray, CDtIO*> m_xaInput; // used to fill m_lbInput

alog Data
//{{AFX_DATA(CDlgInput)
enum { IDD = IDD_INPUTS };
CListBox    m_lbInput;
BYTE    m_nInit;
CString m_strName;
//}}AFX_DATA


verrides
// ClassWizard generate virtual function overrides
//{{AFX_VIRTUAL(CDlgInput)
ublic:
virtual BOOL OnSetActive();
virtual BOOL OnApply();
rotected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

plementation
cted:
// Generated message map functions
//{{AFX_MSG(CDlgInput)
virtual BOOL OnInitDialog();
afx_msg void OnBUTTONAddInput();
afx_msg void OnBUTTONInsertInput();
afx_msg void OnBUTTONRemoveInput();
afx_msg void OnBUTTONUpdateInput();
afx_msg void OnDblclkLISTInput();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

te:
oid AddToListBox(CDtIO* pDtIn, int nSel = -1);
OOL Valid(BOOL add);


////////////////////////////////////////////////////////////////////////////
lgOutput dialog

 CDlgOutput : public CPropertyPage

DECLARE_DYNCREATE(CDlgOutput)

nstruction
c:
DlgOutput();
```

```
virtual BOOL OnApply();
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

mplementation
ected:
// Generated message map functions
//{{AFX_MSG(CDlgOutput)
virtual BOOL OnInitDialog();
afx_msg void OnBUTTONAddOutput();
afx_msg void OnBUTTONInsertOutput();
afx_msg void OnBUTTONRemoveOutput();
afx_msg void OnBUTTONUpdateOutput();
afx_msg void OnDblclkLISTOutput();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

ate:
void AddToListBox(CDtIO* pDtOut, int nSel = -1);
BOOL Valid(BOOL Add);


////////////////////////////////////////////////////////////////////////////
DlgTerm dialog

s CDlgTerm : public CPropertyPage

DECLARE_DYNCREATE(CDlgTerm)

onstruction
lic:
CDlgTerm();
~CDlgTerm();
CTypedPtrArray<CObArray, CDtTrm*> m_xaTerm; // used to fill m_lbTerm
CStringArray m_saInput;     // used to fill m_cbInput, m_cbCenter & m_cbWi
CStringArray m_saOutput;    // used to fill m_cbInput, m_cbCenter & m_cbWi

alog Data
//{{AFX_DATA(CDlgTerm)
enum { IDD = IDD_TERMS };
CComboBox   m_cbType;
CComboBox   m_cbInput;
CComboBox   m_cbWidth;
CComboBox   m_cbCenter;
CListBox    m_lbTerm;
CString m_strCenter;
CString m_strName;
CString m_strWidth;
//}}AFX_DATA


errides
/ ClassWizard generate virtual function overrides
```

```
fx_msg void OnButtonUpdateTerm();
irtual BOOL OnInitDialog();
fx_msg void OnDblclkLISTTerm();
/}}AFX_MSG
ECLARE_MESSAGE_MAP()

te:
oid AddToListBox(CDtTrm* pDtTrm, int nSel = -1);
OOL AllSet(BOOL Update);
OOL Valid();


////////////////////////////////////////////////////////////////////////////
lgRule dialog

 CDlgRule : public CPropertyPage

ECLARE_DYNCREATE(CDlgRule)

nstruction
c:
DlgRule();
CDlgRule();
TypedPtrArray<CObArray, CDtRl*> m_xaRule;  // used to fill m_lbRule
StringArray m_saTerm;       // used to fill m_cbTerm
StringArray m_saInput;      // used to fill m_cbAction
StringArray m_saOutput;     // used to fill m_cbOutput & m_cbAction
DtRl m_xTemp;  // used to store temporary data to fill Rules Edit Box

alog Data
/{{AFX_DATA(CDlgRule)
num { IDD = IDD_RULES };
ComboBox    m_cbType;
ComboBox    m_cbTerm;
ComboBox    m_cbOutput;
ListBox     m_lbRule;
ComboBox    m_cbAction;
String m_strTerms;
String m_strAction;
/}}AFX_DATA


errides
/ ClassWizard generate virtual function overrides
/{{AFX_VIRTUAL(CDlgRule)
ublic:
irtual BOOL OnSetActive();
irtual BOOL OnKillActive();
irtual LRESULT OnWizardNext();
rotected:
irtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
/}}AFX_VIRTUAL

plementation
```

```
BOOL Valid();

/////////////////////////////////////////////////////////////////////////////
FIDEParDlg

s CFIDEParDlg : public CPropertySheet

DECLARE_DYNAMIC(CFIDEParDlg)

onstruction
ic:
CFIDEParDlg(UINT nIDCaption, CWnd* pParentWnd = NULL, UINT iSelectPage = 
CFIDEParDlg(LPCTSTR pszCaption, CWnd* pParentWnd = NULL, UINT iSelectPage


tributes
ic:
CDlgInput m_InpDlg;
CDlgOutput m_OutDlg;
CDlgTerm m_TrmDlg;
CDlgRule m_RleDlg;

perations
ic:

verrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CFIDEParDlg)
//}}AFX_VIRTUAL

mplementation
.c:
virtual ~CFIDEParDlg();

// Generated message map functions
ected:
//{{AFX_MSG(CFIDEParDlg)
    // NOTE - the ClassWizard will add and remove member functions here.
//}}AFX_MSG
DECLARE_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
IDE Dialog helper functions

ValidLabel(CString str);
AddToComboBox(CString str, CComboBox &Combo, int nSel = -1);

FX_INSERT_LOCATION}}
crosoft Visual C++ will insert additional declarations immediately before
ious line.

f // !defined(AFX_FIDEPARDLG_H__8724F848_0C45_11D3_9DCE_00E04CAACD92__INC
```

```
nput::CDlgInput() : CPropertyPage(CDlgInput::IDD)

/{{AFX_DATA_INIT(CDlgInput)
_nInit = 0;
_strName = _T("");
/}}AFX_DATA_INIT


nput::~CDlgInput()



CDlgInput::DoDataExchange(CDataExchange* pDX)

PropertyPage::DoDataExchange(pDX);
/{{AFX_DATA_MAP(CDlgInput)
DX_Control(pDX, IDC_LIST_Input, m_lbInput);
DX_Text(pDX, IDC_EDIT_InitialInput, m_nInit);
DX_Text(pDX, IDC_EDIT_InputLabel, m_strName);
/}}AFX_DATA_MAP



_MESSAGE_MAP(CDlgInput, CPropertyPage)
/{{AFX_MSG_MAP(CDlgInput)
N_BN_CLICKED(IDC_BUTTON_AddInput, OnBUTTONAddInput)
N_BN_CLICKED(IDC_BUTTON_InsertInput, OnBUTTONInsertInput)
N_BN_CLICKED(IDC_BUTTON_RemoveInput, OnBUTTONRemoveInput)
N_BN_CLICKED(IDC_BUTTON_UpdateInput, OnBUTTONUpdateInput)
N_LBN_DBLCLK(IDC_LIST_Input, OnDblclkLISTInput)
/}}AFX_MSG_MAP
ESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
lgInput internal implementation

CDlgInput::AddToListBox(CDtI0 *pDtIn, int nSel)

/ Add new CDtI0 to the listbox.
String str;
DtIn->Format(str);
f (nSel == -1)

   nSel = m_lbInput.AddString(str);
   m_lbInput.SetCurSel(m_lbInput.GetCount()-1);

lse

   m_lbInput.InsertString(nSel, str);
   m_lbInput.SetCurSel(nSel);
```

```
}

// Check whether the label is valid
if (!ValidLabel(m_strName))
{
    AfxMessageBox(IDS_ILLEGAL_LABEL);
    return FALSE;
}

return TRUE;


////////////////////////////////////////////////////////////////////////////
DlgInput virtual functions override

 CDlgInput::OnSetActive()

// Set wizard mode's buttons
CPropertySheet* parent = (CPropertySheet*) GetParent();
parent->SetWizardButtons(PSWIZB_BACK | PSWIZB_NEXT);

return CPropertyPage::OnSetActive();


 CDlgInput::OnApply()

// Update Add Input listbox in Term & Rule Page
CFIDEParDlg* parent = (CFIDEParDlg*) GetParent();
parent->m_TrmDlg.m_saInput.RemoveAll();
parent->m_RleDlg.m_saInput.RemoveAll();

int count = m_xaInput.GetSize();
parent->m_TrmDlg.m_saInput.SetSize(count);
parent->m_RleDlg.m_saInput.SetSize(count);

for (int i = 0; i < count; i++)
{
    parent->m_TrmDlg.m_saInput.SetAt(i, m_xaInput.ElementAt(i)->GetName()
    parent->m_RleDlg.m_saInput.SetAt(i, m_xaInput.ElementAt(i)->GetName()
}

return CPropertyPage::OnApply();


////////////////////////////////////////////////////////////////////////////
DlgInput message handlers

 CDlgInput::OnInitDialog()

CPropertyPage::OnInitDialog();

// Copy all of the entries from m_xaInput to the listbox
m_lbInput.ResetContent();
```

```
pDtIn->SetName(m_strName);
pDtIn->SetInit(m_nInit);

// Add an Input to m_xaInput
m_xaInput.Add(pDtIn);
AddToListBox(pDtIn);

// Activate Apply button
SetModified();


CDlgInput::OnBUTTONInsertInput()

if (UpdateData() != TRUE)
    return;

if (!Valid(TRUE))
    return;

// Check whether an Input has been selected
int nSel = m_lbInput.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_INPUT);
    return;
}

// Create a new CDtIO object
CDtIO* pDtIn = new CDtIO;
pDtIn->SetName(m_strName);
pDtIn->SetInit(m_nInit);

// Insert an Input to m_xaInput
m_xaInput.InsertAt(nSel, pDtIn);
AddToListBox(pDtIn, nSel);

// Activate Apply button
SetModified();


CDlgInput::OnBUTTONRemoveInput()

// Check whether an Input has been selected
int nSel = m_lbInput.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_INPUT);
    return;
}

// Delete an Input on m_xaInput
delete m_xaInput[nSel];
m_xaInput.RemoveAt(nSel);
m_lbInput.DeleteString(nSel);
```

```
    AfxMessageBox(IDS_INPUT);
    return;


/ Update an existing Input on m_xaInput
DtIO* pDtIn = m_xaInput.ElementAt(nSel);
DtIn->SetName(m_strName);
DtIn->SetInit(m_nInit);

/ Update Input List
_lbInput.DeleteString(nSel);
ddToListBox(pDtIn, nSel);

/ Activate Apply button
etModified();


CDlgInput::OnDblclkLISTInput()

nt nSel = m_lbInput.GetCurSel();
_nInit = m_xaInput.ElementAt(nSel)->GetInit();
_strName = m_xaInput.ElementAt(nSel)->GetName();
OOL check = UpdateData(FALSE);
ssert(check);


////////////////////////////////////////////////////////////////////////////
lgOutput property page

MENT_DYNCREATE(CDlgOutput, CPropertyPage)

utput::CDlgOutput() : CPropertyPage(CDlgOutput::IDD)

/{{AFX_DATA_INIT(CDlgOutput)
_nInit = 0;
_strName = _T("");
/}}AFX_DATA_INIT


utput::~CDlgOutput()


CDlgOutput::DoDataExchange(CDataExchange* pDX)

PropertyPage::DoDataExchange(pDX);
/{{AFX_DATA_MAP(CDlgOutput)
DX_Control(pDX, IDC_LIST_Output, m_lbOutput);
DX_Text(pDX, IDC_EDIT_InitialOutput, m_nInit);
DX_Text(pDX, IDC_EDIT_OutputLabel, m_strName);
/}}AFX_DATA_MAP
```

```
pt0ut->Format(str);
f (nSel == -1)

   nSel = m_lb0utput.AddString(str);
   m_lb0utput.SetCurSel(m_lb0utput.GetCount()-1);

lse

   m_lb0utput.InsertString(nSel, str);
   m_lb0utput.SetCurSel(nSel);



CDlg0utput::Valid(B00L Add)

/ 0utput is limited in 4
f (Add)

   if (m_xa0utput.GetSize() == 4)
   {
       AfxMessageBox(IDS_2_MANY_0UTPUT);
       return FALSE;
   }

   // Check for duplicate label
   for (int i = 0; i < m_xa0utput.GetSize(); i++)
       if (m_xa0utput.ElementAt(i)->GetName() == m_strName)
       {
           AfxMessageBox(IDS_DUPLICATE_0UTPUT);
           return FALSE;
       }


/ Check whether the label is valid
f (!ValidLabel(m_strName))

   AfxMessageBox(IDS_ILLEGAL_LABEL);
   return FALSE;


eturn TRUE;


////////////////////////////////////////////////////////////////////////////
lg0utput virtual functions override

CDlg0utput::0nSetActive()

/ Set wizard mode's buttons
PropertySheet* parent = (CPropertySheet*) GetParent();
arent->SetWizardButtons(PSWIZB_BACK | PSWIZB_NEXT);

eturn CPropertyPage::0nSetActive();
```

```
return CPropertyPage::OnApply();


////////////////////////////////////////////////////////////////////////////
Dlg0utput message handlers

 CDlg0utput::0nInitDialog()

CPropertyPage::0nInitDialog();

// Copy all of the entries from m_xa0utput to m_lb0utput listbox.
m_lb0utput.ResetContent();
for (int i = 0; i < m_xa0utput.GetSize(); i++)
    AddToListBox(m_xa0utput[i]);

return TRUE;  // return TRUE unless you set the focus to a control
              // EXCEPTI0N: 0CX Property Pages should return FALSE


 CDlg0utput::0nBUTT0NAdd0utput()

if (UpdateData() != TRUE)
    return;

if (!Valid(TRUE))
    return;

// Create a new CDtI0 object
CDtI0* pDt0ut = new CDtI0;
pDt0ut->SetName(m_strName);
pDt0ut->SetInit(m_nInit);

// Add an 0utput to m_xa0utput
m_xa0utput.Add(pDt0ut);
AddToListBox(pDt0ut);

// Activate Apply button
SetModified();


 CDlg0utput::0nBUTT0NInsert0utput()

if (UpdateData() != TRUE)
    return;

if (!Valid(TRUE))
    return;

// Check whether an 0utput has been selected
int nSel = m_lb0utput.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_0UTPUT);
```

```
// Check whether an Output has been selected
int nSel = m_lbOutput.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_OUTPUT);
    return;
}

// Delete an Output on m_xaOutput
delete m_xaOutput[nSel];
m_xaOutput.RemoveAt(nSel);
m_lbOutput.DeleteString(nSel);

// Activate Apply button
SetModified();


 CDlgOutput::OnBUTTONUpdateOutput()

if (UpdateData() != TRUE)
    return;

if (!Valid(FALSE))
    return;

// Check whether an Output has been selected
int nSel = m_lbOutput.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_OUTPUT);
    return;
}

// Update an existing Output on m_xaOutput
CDtIO* pDtOut = m_xaOutput.ElementAt(nSel);
pDtOut->SetName(m_strName);
pDtOut->SetInit(m_nInit);

// Update Output List
m_lbOutput.DeleteString(nSel);
AddToListBox(pDtOut, nSel);

// Activate Apply button
SetModified();


 CDlgOutput::OnDblclkLISTOutput()

int nSel = m_lbOutput.GetCurSel();
m_nInit = m_xaOutput.ElementAt(nSel)->GetInit();
m_strName = m_xaOutput.ElementAt(nSel)->GetName();
BOOL check = UpdateData(FALSE);
assert(check);
```

```
 CDlgTerm::DoDataExchange(CDataExchange* pDX)

CPropertyPage::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CDlgTerm)
DDX_Control(pDX, IDC_COMBO_TermType, m_cbType);
DDX_Control(pDX, IDC_COMBO_TermInput, m_cbInput);
DDX_Control(pDX, IDC_COMBO_TermWidth, m_cbWidth);
DDX_Control(pDX, IDC_COMBO_TermCenter, m_cbCenter);
DDX_Control(pDX, IDC_LIST_Term, m_lbTerm);
DDX_CBString(pDX, IDC_COMBO_TermCenter, m_strCenter);
DDX_Text(pDX, IDC_EDIT_TermLabel, m_strName);
DDX_CBString(pDX, IDC_COMBO_TermWidth, m_strWidth);
//}}AFX_DATA_MAP



N_MESSAGE_MAP(CDlgTerm, CPropertyPage)
//{{AFX_MSG_MAP(CDlgTerm)
ON_BN_CLICKED(IDC_BUTTON_AddTerm, OnBUTTONAddTerm)
ON_BN_CLICKED(IDC_BUTTON_InsertTerm, OnBUTTONInsertTerm)
ON_BN_CLICKED(IDC_BUTTON_RemoveTerm, OnBUTTONRemoveTerm)
ON_BN_CLICKED(IDC_BUTTON_UpdateTerm, OnBUTTONUpdateTerm)
ON_LBN_DBLCLK(IDC_LIST_Term, OnDblclkLISTTerm)
//}}AFX_MSG_MAP
MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
DlgTerm internal implementation

 CDlgTerm::AddToListBox(CDtTrm *pDtTrm, int nSel)

// Add new CDtTrm to the listbox
CString str;
pDtTrm->Format(str);
if (nSel == -1)
{
    nSel = m_lbTerm.AddString(str);
    m_lbTerm.SetCurSel(m_lbTerm.GetCount()-1);
}
else
{
    m_lbTerm.InsertString(nSel, str);
    m_lbTerm.SetCurSel(nSel);
}


 CDlgTerm::AllSet(BOOL Update)

// Check whether the label is valid
if (!ValidLabel(m_strName))
{
```

```
    AfxMessageBox(IDS_INPUT);
    return FALSE;
}

// Check whether a Type has been selected
nSel = m_cbType.GetCurSel();
if (nSel == CB_ERR)
{
    AfxMessageBox(IDS_MF_TYPE);
    return FALSE;
}

return TRUE;


CDlgTerm::Valid()

// Validate CDtTrm's data
CFIDEParDlg* parent = (CFIDEParDlg*) GetParent();
for (int i = 0; i < m_xaTerm.GetSize(); i++)
{
    // ----- validate Input data
    if (m_cbInput.FindStringExact(-1,
        m_xaTerm.ElementAt(i)->GetInput()) == CB_ERR)
    {
        AfxMessageBox(IDS_UNRECOGNIZED_INPUT);
        return FALSE;
    }
    // ----- validate Center data
    if ((m_cbCenter.FindStringExact(-1,
         m_xaTerm.ElementAt(i)->GetCenter()) == CB_ERR) &&
        (m_xaTerm.ElementAt(i)->GetCenter()[0] > '9'))
    {
        AfxMessageBox(IDS_UNRECOGNIZED_FLOATING);
        return FALSE;
    }
    // ----- validate Width data
    if ((m_cbWidth.FindStringExact(-1,
         m_xaTerm.ElementAt(i)->GetWidth()) == CB_ERR) &&
        (m_xaTerm.ElementAt(i)->GetWidth()[0] > '9'))
    {
        AfxMessageBox(IDS_UNRECOGNIZED_FLOATING);
        return FALSE;
    }
}

return TRUE;

////////////////////////////////////////////////////////////////////////////
DlgTerm virtual functions override

CDlgTerm::OnSetActive()

/ Update listboxes contents
```

```
for (i = 0; i < m_saOutput.GetSize(); i++)
    AddToComboBox(m_saOutput[i], m_cbWidth);
// ----- highlight cell in Type combo box
m_cbType.SetCurSel(m_cbType.GetCount()-1);
// ----- copy all entries from m_xaTerm to m_lbTerm listbox.
m_lbTerm.ResetContent();
for (i = 0; i < m_xaTerm.GetSize(); i++)
    AddToListBox(m_xaTerm[i]);

// Set wizard mode's buttons
CPropertySheet* parent = (CPropertySheet*) GetParent();
parent->SetWizardButtons(PSWIZB_BACK | PSWIZB_NEXT);

return CPropertyPage::OnSetActive();


 CDlgTerm::OnApply()

// Update Add Term listbox in Rule page
CFIDEParDlg* parent = (CFIDEParDlg*) GetParent();
parent->m_RleDlg.m_saTerm.RemoveAll();
int count = m_xaTerm.GetSize();
parent->m_RleDlg.m_saTerm.SetSize(count);
for (int i = 0; i < count; i++)
    parent->m_RleDlg.m_saTerm.SetAt(i,
    m_xaTerm.ElementAt(i)->GetName());

return CPropertyPage::OnApply();


 CDlgTerm::OnKillActive()

// Validate terms' data
if (!Valid())
    return FALSE;

return CPropertyPage::OnKillActive();


ULT CDlgTerm::OnWizardNext()

// Validate terms' data
if (!Valid())
    return FALSE;

return CPropertyPage::OnWizardNext();


////////////////////////////////////////////////////////////////////////////
DlgTerm message handlers

 CDlgTerm::OnInitDialog()

CPropertyPage::OnInitDialog();
```

MILIK PERPUSTAKAAN
ITS

```
n_cbInput.GetLBText(m_cbInput.GetCurSel(), temp);
pDtTrm->SetInput(temp);
pDtTrm->SetCenter(m_strCenter);
pDtTrm->SetWidth(m_strWidth);
pDtTrm->SetType(2 + m_cbType.GetCurSel());

// Add a Term to m_xaTerm
n_xaTerm.Add(pDtTrm);
AddToListBox(pDtTrm);

// Activate Apply button
SetModified();


CDlgTerm::OnBUTTONInsertTerm()

if (UpdateData() != TRUE)
    return;

if (!AllSet(FALSE))
    return;

// Check whether a Term has been selected
int nSel = m_lbTerm.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_TERM);
    return;
}

// Create a new CDtTrm object
CDtTrm* pDtTrm = new CDtTrm;
pDtTrm->SetName(m_strName);
CString temp;
n_cbInput.GetLBText(m_cbInput.GetCurSel(), temp);
pDtTrm->SetInput(temp);
pDtTrm->SetCenter(m_strCenter);
pDtTrm->SetWidth(m_strWidth);
pDtTrm->SetType(2 + m_cbType.GetCurSel());

// Insert a Term to m_xaTerm
n_xaTerm.InsertAt(nSel, pDtTrm);
AddToListBox(pDtTrm, nSel);

// Activate Apply button
SetModified();


CDlgTerm::OnBUTTONRemoveTerm()

// Check whether a Term has been selected
int nSel = m_lbTerm.GetCurSel();
if (nSel == LB_ERR)
{
```

```
    return;

if (!AllSet(TRUE))
    return;

// Check whether a Term has been selected
int nSel = m_lbTerm.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_TERM);
    return;
}

// Update an existing Term on m_xaTerm
CDtTrm* pDtTrm = m_xaTerm.ElementAt(nSel);
pDtTrm->SetName(m_strName);
CString temp;
m_cbInput.GetLBText(m_cbInput.GetCurSel(), temp);
pDtTrm->SetInput(temp);
pDtTrm->SetCenter(m_strCenter);
pDtTrm->SetWidth(m_strWidth);
pDtTrm->SetType(2 + m_cbType.GetCurSel());

m_lbTerm.DeleteString(nSel);
AddToListBox(pDtTrm, nSel);

// Activate Apply button
SetModified();


 CDlgTerm::OnDblclkLISTTerm()

int nSel = m_lbTerm.GetCurSel();
int i = m_cbInput.SelectString(-1,
    LPCTSTR(m_xaTerm.ElementAt(nSel)->GetInput()));
assert (i != CB_ERR);
m_strName   = m_xaTerm.ElementAt(nSel)->GetName();
m_strCenter = m_xaTerm.ElementAt(nSel)->GetCenter();
m_strWidth  = m_xaTerm.ElementAt(nSel)->GetWidth();
m_cbType.SetCurSel(m_xaTerm.ElementAt(nSel)->GetType()-2);
BOOL check = UpdateData(FALSE);
assert(check);


/////////////////////////////////////////////////////////////////////////////
DlgRule property page

EMENT_DYNCREATE(CDlgRule, CPropertyPage)

Rule::CDlgRule() : CPropertyPage(CDlgRule::IDD)

//{{AFX_DATA_INIT(CDlgRule)
m_strTerms = _T("");
m_strAction = _T("");
```

```
DDX_CBString(pDX, IDC_COMBO_RuleAction, m_strAction);
//}}AFX_DATA_MAP



N_MESSAGE_MAP(CDlgRule, CPropertyPage)
//{{AFX_MSG_MAP(CDlgRule)
ON_BN_CLICKED(IDC_BUTTON_AddRuleTerm, OnBUTTONAddRuleTerm)
ON_BN_CLICKED(IDC_BUTTON_ClearRuleTerm, OnBUTTONClearRuleTerm)
ON_BN_CLICKED(IDC_BUTTON_AddRule, OnBUTTONAddRule)
ON_BN_CLICKED(IDC_BUTTON_InsertRule, OnBUTTONInsertRule)
ON_BN_CLICKED(IDC_BUTTON_RemoveRule, OnBUTTONRemoveRule)
ON_BN_CLICKED(IDC_BUTTON_UpdateRule, OnBUTTONUpdateRule)
ON_LBN_DBLCLK(IDC_LIST_RuleList, OnDblclkLISTRuleList)
//}}AFX_MSG_MAP
MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
DlgTerm internal implementation

 CDlgRule::AddToListBox(CDtRl *pDtRl, int nSel)

// Add new CDtRl to the listbox
CString str;
pDtRl->Format(str);
if (nSel == -1)
{
    nSel = m_lbRule.AddString(str);
    m_lbRule.SetCurSel(m_lbRule.GetCount()-1);
}
else
{
    m_lbRule.InsertString(nSel, str);
    m_lbRule.SetCurSel(nSel);
}


 CDlgRule::AllSet()

// Check whether an Output has been selected
int nSel = m_cbOutput.GetCurSel();
if (nSel == CB_ERR)
{
    AfxMessageBox(IDS_OUTPUT);
    return FALSE;
}

// Check whether a Type has been selected
nSel = m_cbType.GetCurSel();
if (nSel == CB_ERR)
{
    AfxMessageBox(IDS_DFZ_TYPE);
    return FALSE;
}
```

```
        }
    // ----- validate Output data
    if (m_cbOutput.FindStringExact(-1,
        m_xaRule.ElementAt(i)->GetOutput()) == CB_ERR)
    {
        AfxMessageBox(IDS_UNRECOGNIZED_OUTPUT);
        return FALSE;
    }
    // ----- validate Action data
    if ((m_cbAction.FindStringExact(-1,
         m_xaRule.ElementAt(i)->GetAction()) == CB_ERR) &&
        (m_xaRule.ElementAt(i)->GetAction()[0] > '9'))
    {
        AfxMessageBox(IDS_UNRECOGNIZED_FLOATING);
        return FALSE;
    }
}
return TRUE;


////////////////////////////////////////////////////////////////////////////
DlgTerm virtual functions override

CDlgRule::OnSetActive()

// Update listboxes contents
BOOL check = UpdateData(FALSE);
// ----- copy all entries from m_saTerm to m_cbTerm
m_cbTerm.ResetContent();
for (int i = 0; i < m_saTerm.GetSize(); i++)
    AddToComboBox(m_saTerm[i], m_cbTerm);
// ----- copy all entries from m_saOutput to m_cbOutput
m_cbOutput.ResetContent();
for (i = 0; i < m_saOutput.GetSize(); i++)
    AddToComboBox(m_saOutput[i], m_cbOutput);
// ----- copy all entries from m_saInput & m_saOutput to m_cbAction
m_cbAction.ResetContent();
for (i = 0; i < m_saInput.GetSize(); i++)
    AddToComboBox(m_saInput[i], m_cbAction);
for (i = 0; i < m_saOutput.GetSize(); i++)
    AddToComboBox(m_saOutput[i], m_cbAction);
// ----- highlight cell in Type combo box
m_cbType.SetCurSel(m_cbType.GetCount()-1);
// ---------- copy all entries from m_xaRule to m_lbRule
m_lbRule.ResetContent();
for (i = 0; i < m_xaRule.GetSize(); i++)
    AddToListBox(m_xaRule[i]);

// Set wizard mode's buttons
CPropertySheet* parent = (CPropertySheet*) GetParent();
parent->SetWizardButtons(PSWIZB_BACK | PSWIZB_FINISH);

return CPropertyPage::OnSetActive();
```

```
////////////////////////////////////////////////////////////////////////////
DlgRule message handlers

 CDlgRule::OnInitDialog()

CPropertyPage::OnInitDialog();

return TRUE;  // return TRUE unless you set the focus to a control
              // EXCEPTION: OCX Property Pages should return FALSE


 CDlgRule::OnBUTTONAddRuleTerm()

if (UpdateData() != TRUE)
    return;

// Check whether a Term has been selected
int nSel = m_cbTerm.GetCurSel();
if (nSel == CB_ERR)
{
    AfxMessageBox(IDS_TERM);
    return;
}

// Check for duplicate Term
CString str;
m_cbTerm.GetLBText(nSel, str);
for (int i = 0; i < m_xTemp.saTerms.GetSize(); i++)
    if (m_xTemp.saTerms[i] == str)
    {
        AfxMessageBox(IDS_DUPLICATE_TERM_RULE);
        return;
    }

// Add a Term to m_Temp.Terms
m_xTemp.saTerms.Add(str);
m_xTemp.FormatTerms(m_strTerms);

// Force redraw to update Terms
if (UpdateData(FALSE))
    SetModified();  // activate Apply button


 CDlgRule::OnBUTTONClearRuleTerm()

// Remove all temporary entry on m_Temp.Terms
m_xTemp.saTerms.RemoveAll();
m_strTerms = _T("");

// Force redraw to update Terms
if (UpdateData(FALSE))
    SetModified();  // activate Apply button
```

```
    pDtRl->saTerms[i] = m_xTemp.saTerms[i];
pDtRl->SetAction(m_strAction);
pDtRl->SetType(m_cbType.GetCurSel());

// Add a Rule to m_xaRule
m_xaRule.Add(pDtRl);
AddToListBox(pDtRl);

// Activate Apply button
SetModified();


 CDlgRule::OnBUTTONInsertRule()

if (UpdateData() != TRUE)
    return;

if (!AllSet())
    return;

// Check whether a Rule has been selected
int nSel = m_lbRule.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_RULE);
    return;
}

// Create a new CDtRl object
CDtRl* pDtRl = new CDtRl;
CString temp;
m_cbOutput.GetLBText(m_cbOutput.GetCurSel(), temp);
pDtRl->SetOutput(temp);
pDtRl->saTerms.SetSize(m_xTemp.saTerms.GetSize());
for (int i = 0; i < m_xTemp.saTerms.GetSize(); i++)
    pDtRl->saTerms[i] = m_xTemp.saTerms[i];
pDtRl->SetAction(m_strAction);
pDtRl->SetType(m_cbType.GetCurSel());

// Insert a Rule to m_xaRule
m_xaRule.InsertAt(nSel, pDtRl);
AddToListBox(pDtRl, nSel);

// Activate Apply button
SetModified();


 CDlgRule::OnBUTTONRemoveRule()

// Check whether a Rule has been selected
int nSel = m_lbRule.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_RULE);
```

MILIK PERPUSTAKAAN
ITS

```
if (!AllSet())
    return;

// Check whether a Rule has been selected
int nSel = m_lbRule.GetCurSel();
if (nSel == LB_ERR)
{
    AfxMessageBox(IDS_RULE);
    return;
}

// Update an existing Rule on m_xaRule
CDtRl* pDtRl = m_xaRule.ElementAt(nSel);
CString temp;
m_cbOutput.GetLBText(m_cbOutput.GetCurSel(), temp);
pDtRl->SetOutput(temp);
pDtRl->saTerms.SetSize(m_xTemp.saTerms.GetSize());
for (int i = 0; i < m_xTemp.saTerms.GetSize(); i++)
    pDtRl->saTerms[i] = m_xTemp.saTerms[i];
pDtRl->SetAction(m_strAction);
pDtRl->SetType(m_cbType.GetCurSel());

m_lbRule.DeleteString(nSel);
AddToListBox(pDtRl, nSel);

// Activate Apply button
SetModified();


CDlgRule::OnDblclkLISTRuleList()

int nSel = m_lbRule.GetCurSel();
m_xTemp.saTerms.Copy(m_xaRule.ElementAt(nSel)->saTerms);
m_xTemp.FormatTerms(m_strTerms);
int i = m_cbOutput.SelectString(-1,
    LPCTSTR(m_xaRule.ElementAt(nSel)->GetOutput()));
assert (i != CB_ERR);
m_strAction = m_xaRule.ElementAt(nSel)->GetAction();
m_cbType.SetCurSel(m_xaRule.ElementAt(nSel)->GetType());
BOOL check = UpdateData(FALSE);
assert(check);


////////////////////////////////////////////////////////////////////////////
FIDEParDlg

EMENT_DYNAMIC(CFIDEParDlg, CPropertySheet)

EParDlg::CFIDEParDlg(UINT nIDCaption, CWnd* pParentWnd, UINT iSelectPage)
:CPropertySheet(nIDCaption, pParentWnd, iSelectPage)

AddPage(&m_InpDlg);
AddPage(&m_OutDlg);
AddPage(&m_TrmDlg);
```

```
N_MESSAGE_MAP(CFIDEParDlg, CPropertySheet)
//{{AFX_MSG_MAP(CFIDEParDlg)
    // NOTE - the ClassWizard will add and remove mapping macros here.
//}}AFX_MSG_MAP
MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
FIDEParDlg message handlers

/////////////////////////////////////////////////////////////////////////////
IDE Dialog helper functions

 ValidLabel(CString str)


if (((str[0] >= 'A') && (str[0] <= 'Z')) ||
    ((str[0] >= 'a') && (str[0] <= 'z')))
    return TRUE;
else
    return FALSE;


 AddToComboBox(CString str, CComboBox &Combo, int nSel)

// Fill Combo combobox
if (nSel == -1)
{
    nSel = Combo.AddString(str);
    Combo.SetCurSel(Combo.GetCount()-1);
}
else
{
    Combo.InsertString(nSel, str);
    Combo.SetCurSel(nSel);
}
```

```
; double nY_INC     = 0.07;
; int nMIN_X        = 0;
; int nMAX_X        = 255;
; int nMIN_F        = 0;
; int nMAX_F        = 63;

////////////////////////////////////////////////////////////////////////////
TrmView form view

def __AFXEXT_H__
lude <afxext.h>
if

s CTrmView : public CFormView

ected:
CTrmView();           // protected constructor used by dynamic creation
DECLARE_DYNCREATE(CTrmView)
virtual ~CTrmView();

orm Data
ic:
//{{AFX_DATA(CTrmView)
enum { IDD = IDD_TRMVIEW_FORM };
CComboBox   m_cbIOList;
CStrip  m_stripTrmV;
//}}AFX_DATA

tributes
ic:
CFIDEDoc* GetDocument();

ate:
CString m_strLabel;     // I/O Label
CByteArray m_baTrmIdx; // terms' index

perations
ic:

verrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CTrmView)
ublic:
virtual void OnInitialUpdate();
rotected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

mplementation
ected:
// Get indexed-based OLE_COLOR
ong GetColor(int idx);
// Draw the chart !!
OOL DrawChart(CDtCnvrtd* pCnvrtd, BYTE idx);
```

```
 return (CFIDEDoc*)m_pDocument; }
if

////////////////////////////////////////////////////////////////////////////

AFX_INSERT_LOCATION}}
icrosoft Visual C++ will insert additional declarations immediately befor
vious line.

if // !defined(AFX_FIDETRMV_H__635FDD21_41FB_11D3_BD00_00E04CAACD92__INCL
```

```
View::CTrmView()
: CFormView(CTrmView::IDD)

//{{AFX_DATA_INIT(CTrmView)
    // NOTE: the ClassWizard will add member initialization here
//}}AFX_DATA_INIT


View::~CTrmView()



 CTrmView::DoDataExchange(CDataExchange* pDX)

CFormView::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CTrmView)
DDX_Control(pDX, IDC_COMBO_IO_SLCT, m_cbIOList);
DDX_Control(pDX, IDC_STRIPCTRL_TrmV1, m_stripTrmV);
//}}AFX_DATA_MAP



N_MESSAGE_MAP(CTrmView, CFormView)
//{{AFX_MSG_MAP(CTrmView)
ON_CBN_SELCHANGE(IDC_COMBO_IO_SLCT, OnSelchangeComboIoSlct)
//}}AFX_MSG_MAP
MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
TrmView diagnostics

ef _DEBUG
 CTrmView::AssertValid() const

CFormView::AssertValid();


 CTrmView::Dump(CDumpContext& dc) const

CFormView::Dump(dc);


IDoc* CTrmView::GetDocument() // non-debug version is inline

ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CFIDEDoc)));
return (CFIDEDoc*)m_pDocument;

if //_DEBUG

/////////////////////////////////////////////////////////////////////////////
TrmView internal implementation
et indexed-based OLE_COLOR
 CTrmView::GetColor(int idx)
```

```
CTrmView::DrawChart(CDtCnvrtd* pCnvrtd, BYTE idx)

// --- Reset Chart
for (int i = 0; i < m_stripTrmV.GetVariables(); i++)
    m_stripTrmV.ClearVariable(i);
m_stripTrmV.ClearTrack(0);
m_stripTrmV.ClearAll();

int m_nTrmNum = 0;
CString strTmp;
m_baTrmIdx.RemoveAll();

// Pick terms matching I/O selected by idx
m_baTrmIdx.SetSize(pCnvrtd->saTrmLbl.GetSize());
for (i = 0; i < pCnvrtd->saTrmLbl.GetSize(); i++)
    if (pCnvrtd->baTrmInp[i] == idx)
    {
        m_baTrmIdx[m_nTrmNum] = i; // store term's index
        m_nTrmNum++;               // record terms number
    }

// Chart Preparation
// --- Set stripchart variables and captions numbers
m_stripTrmV.SetVariables(m_nTrmNum);
m_stripTrmV.SetCaptions(m_nTrmNum+1);

// Give I/O Label & Cursor Position
m_stripTrmV.SetCaptionID(0);
m_strLabel = pCnvrtd->GetLabel(idx);
strTmp.Format("%s : %.2X", m_strLabel,
    (int)ceil(m_stripTrmV.GetCursorX()));
m_stripTrmV.SetCaption(strTmp);

// Draw the chart !
for (i = 0; i < m_nTrmNum; i++)
{
    long lColor = GetColor(i);
    // --- Draw the Membership Function's Waveform
    m_stripTrmV.SetVariableID(i);
    m_stripTrmV.SetVariableColor(lColor);
    m_stripTrmV.SetVariableType(0);
    m_stripTrmV.SetVariableLineStyle(0);
    m_stripTrmV.SetVariableLineWidth(3);
    m_stripTrmV.SetVariableLastX(nMIN_X);

    // copy to local variable to shorten syntax
    int nCnt = pCnvrtd->GetValue(pCnvrtd->baTrmCnt[m_baTrmIdx[i]]);
    int nWdt = pCnvrtd->GetValue(pCnvrtd->baTrmWdt[m_baTrmIdx[i]]);

    switch (pCnvrtd->baTrmTyp[m_baTrmIdx[i]])
    {
    // ------ Symmetrical Inclusive MF
    case 2:
        if ((nCnt-nWdt) >= nMIN_X)
```

```
            m_stripTrmV.AddXY(i, nCnt+nWdt, nMIN_F);
            m_stripTrmV.AddXY(i, nMAX_X,    nMIN_F);
        }
        else
            m_stripTrmV.AddXY(i, nMAX_X, nMAX_F-nMAX_X+nCnt);
        break;
// ------ Symmetrical Exclusive MF
case 3:
        if ((nCnt-nWdt) >= nMIN_X)
        {
            if ((nCnt-nMAX_F) >= nMIN_X)
            {
                m_stripTrmV.AddXY(i, nMIN_X, nMAX_F);
                m_stripTrmV.AddXY(i, nMIN_X, nMAX_F);
                m_stripTrmV.AddXY(i, nCnt-nMAX_F, nMAX_F);
            }
            else
            {
                m_stripTrmV.AddXY(i, nMIN_X, nMAX_F-nCnt);
                m_stripTrmV.AddXY(i, nMIN_X, nMAX_F-nCnt);
            }
            m_stripTrmV.AddXY(i, nCnt-nWdt, nWdt);
            m_stripTrmV.AddXY(i, nCnt-nWdt, nMIN_F);
        }
        else
            m_stripTrmV.AddXY(i, nMIN_X, nMIN_F);
            m_stripTrmV.AddXY(i, nMIN_X, nMIN_F);

        if ((nCnt+nWdt) <= nMAX_X)
        {
            m_stripTrmV.AddXY(i, nCnt+nWdt, nMIN_F);
            m_stripTrmV.AddXY(i, nCnt+nWdt, nWdt);
            if ((nCnt+nMAX_F) <= nMAX_X)
            {
                m_stripTrmV.AddXY(i, nCnt+nMAX_F, nMAX_F);
                m_stripTrmV.AddXY(i, nMAX_X,      nMAX_F);
            }
            else
                m_stripTrmV.AddXY(i, nMAX_X, nMAX_F-nMAX_X+nCnt);
        }
        else
            m_stripTrmV.AddXY(i, nMAX_X, nMIN_F);
        break;
// ------ Left Inclusive MF
case 4:
        m_stripTrmV.AddXY(i, nMIN_X, nMAX_F);
        m_stripTrmV.AddXY(i, nMIN_X, nMAX_F);
        m_stripTrmV.AddXY(i, nCnt,   nMAX_F);
        if ((nCnt+nWdt) <= nMAX_X)
        {
            m_stripTrmV.AddXY(i, nCnt+nWdt, nMAX_F-nWdt);
            m_stripTrmV.AddXY(i, nCnt+nWdt, nMIN_F);
            m_stripTrmV.AddXY(i, nMAX_X,    nMIN_F);
        }
```

```
            m_stripTrmV.AddXY(i, nMAX_X, nMAX_F-nMAX_X+nCnt);
        }
        else
            m_stripTrmV.AddXY(i, nMAX_X, nMIN_F);
        break;
// ------ Right Inclusive MF
case 6:
    if ((nCnt-nWdt) >= nMIN_X)
    {
        m_stripTrmV.AddXY(i, nMIN_X, nMIN_F);
        m_stripTrmV.AddXY(i, nMIN_X, nMIN_F);
        m_stripTrmV.AddXY(i, nCnt-nWdt, nMIN_F);
        m_stripTrmV.AddXY(i, nCnt-nWdt, nMAX_F-nWdt);
    }
    else
    {
        m_stripTrmV.AddXY(i, nMIN_X, nMAX_F-nCnt);
        m_stripTrmV.AddXY(i, nMIN_X, nMAX_F-nCnt);
    }
    m_stripTrmV.AddXY(i, nCnt,       nMAX_F);
    m_stripTrmV.AddXY(i, nMAX_X,     nMAX_F);
    break;
// ------ Right Exclusive MF
case 7:
    if ((nCnt-nWdt) >= nMIN_X)
    {
        if ((nCnt-nMAX_F) >= nMIN_X)
        {
            m_stripTrmV.AddXY(i, nMIN_X, nMAX_F);
            m_stripTrmV.AddXY(i, nMIN_X, nMAX_F);
            m_stripTrmV.AddXY(i, nCnt-nMAX_F, nMAX_F);
        }
        else
        {
            m_stripTrmV.AddXY(i, nMIN_X, nMAX_F-nCnt);
            m_stripTrmV.AddXY(i, nMIN_X, nMAX_F-nCnt);
        }
        m_stripTrmV.AddXY(i, nCnt-nWdt,   nWdt);
        m_stripTrmV.AddXY(i, nCnt-nWdt,   nMIN_F);
    }
    else
        m_stripTrmV.AddXY(i, nMIN_X, nMIN_F);
    m_stripTrmV.AddXY(i, nMAX_X, nMIN_F);
    break;
default:
    return FALSE;
}

// --- Draw the caption comment
m_stripTrmV.SetCaptionID(i+1);
m_stripTrmV.SetCaptionColor(lColor);
m_stripTrmV.SetCaptionFontID(2);
m_stripTrmV.SetCaptionAlign(0);
if (i%2 == 0)
```

```
CFormView::0nInitialUpdate();

CFIDEDoc* pDoc = GetDocument();

// Fill combobox
// fill with Inputs' labels
for (int i = 0; i < pDoc->m_xaInput.GetSize(); i++)
    m_cbI0List.AddString(pDoc->m_xaInput.ElementAt(i)->GetName());
// fill with 0utputs' labels
for (i = 0; i < pDoc->m_xa0utput.GetSize(); i++)
    m_cbI0List.AddString(pDoc->m_xa0utput.ElementAt(i)->GetName());
if (m_cbI0List.SetCurSel(0) == CB_ERR)
    return;

B00L check = DrawChart(&(pDoc->m_xConverted), 0);
assert(check);


 CTrmView::0nSelchangeComboIoSlct()

CFIDEDoc* pDoc = GetDocument();

CString strTmp;
m_cbI0List.GetLBText(m_cbI0List.GetCurSel(), strTmp);
int idx = pDoc->m_xConverted.FindLabel(strTmp);
if (idx == -1)
{
    AfxMessageBox(IDS_UNREC0GNIZED_I0_LABEL);
    return;
}

B00L check = DrawChart(&(pDoc->m_xConverted), idx);
assert(check);


N_EVENTSINK_MAP(CTrmView, CFormView)
//{{AFX_EVENTSINK_MAP(CTrmView)
0N_EVENT(CTrmView, IDC_STRIPCTRL_TrmV1, 1 /* CursorChange */, 0nCursorCha
TRLTrmV1, VTS_R8)
//}}AFX_EVENTSINK_MAP
EVENTSINK_MAP()

 CTrmView::0nCursorChangeSTRIPCTRLTrmV1(double X)

CString strTmp;
CDtCnvrtd* pCnvrtd = &(GetDocument()->m_xConverted);

// Give I/0 Label & Cursor Position
m_stripTrmV.SetCaptionID(0);
strTmp.Format("%s : %.2X", m_strLabel, (int)ceil(X));
m_stripTrmV.SetCaption(strTmp);

for (int i = 0; i < m_stripTrmV.GetVariables(); i++)
{
```

```
def __AFXEXT_H__
lude <afxext.h>
if

s CFrmSIM : public CFormView

ected:
CFrmSIM();           // protected constructor used by dynamic creation
DECLARE_DYNCREATE(CFrmSIM)

ttributes
ic:
CFIDEDoc* GetDocument();

orm Data
ic:
//{{AFX_DATA(CFrmSIM)
enum { IDD = IDD_SIMULATOR };
CRichEditCtrl    m_rectlSIMComment;
CString m_s7S_FM1;
CString m_s7S_DS1;
CString m_s7S_DS2;
CString m_s7S_DS3;
CString m_s7S_DS4;
CString m_s7S_DS5;
CString m_s7S_FM2;
CString m_s7S_FM3;
CString m_s7S_FM4;
CString m_s7S_FM5;
CString m_s7S_OUT1;
CString m_s7S_OUT2;
CString m_s7S_REDM1;
CString m_s7S_REDM2;
CString m_s7S_REDM3;
CString m_s7S_REDM4;
CString m_s7S_REDM5;
CString m_s7S_SM1;
CString m_s7S_SM2;
CString m_s7S_SM3;
CStrip  m_stripClk;
long    m_lDS1;
long    m_lDS2;
long    m_lDS3;
long    m_lDS4;
long    m_lDS5;
BOOL    m_bCCS;
BOOL    m_bCS;
BOOL    m_bUPDT_IN;
CString m_strSIMComment;
double  m_dLStep;
double  m_dSStep;
CString m_s7S_FRC1;
CString m_s7S_FRC2;
```

```
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

mplementation
ected:
virtual ~CFrmSIM();
ef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
if

// Generated message map functions
//{{AFX_MSG(CFrmSIM)
afx_msg void OnBUTTONSIMConReset();
afx_msg void OnBUTTONSIMDataSave();
afx_msg void OnBUTTONSIMDataLoad();
afx_msg void OnCursorChangeSTRIPCTRLClk(double X);
afx_msg void OnClickLEDCTRLDIPSw1();
afx_msg void OnClickLEDCTRLDIPSw2();
afx_msg void OnClickLEDCTRLDIPSw3();
afx_msg void OnChangeLEDCTRLDIPSw4();
afx_msg void OnChangeLEDCTRLDIPSw5();
afx_msg void OnClickTogglectrlCcs();
afx_msg void OnClickTogglectrlCs();
afx_msg void OnClickTogglectrlUpdtIn();
afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
afx_msg void OnCancelMode();
afx_msg void OnBUTTONSIMConSStep();
afx_msg void OnBUTTONSIMConLStep();
afx_msg void OnViewTextLarge();
afx_msg void OnUpdateViewTextLarge(CCmdUI* pCmdUI);
afx_msg void OnViewTextLargest();
afx_msg void OnUpdateViewTextLargest(CCmdUI* pCmdUI);
afx_msg void OnViewTextMedium();
afx_msg void OnUpdateViewTextMedium(CCmdUI* pCmdUI);
afx_msg void OnViewTextSmall();
afx_msg void OnUpdateViewTextSmall(CCmdUI* pCmdUI);
afx_msg void OnViewTextSmallest();
afx_msg void OnUpdateViewTextSmallest(CCmdUI* pCmdUI);
DECLARE_EVENTSINK_MAP()
//}}AFX_MSG
DECLARE_MESSAGE_MAP()



def _DEBUG  // debug version in FIDEView.cpp
ne CFIDEDoc* CFrmSIM::GetDocument()
 return (CFIDEDoc*)m_pDocument; }
if

////////////////////////////////////////////////////////////////////////////

AFX_INSERT_LOCATION}}
```

```
SIM::CFrmSIM()
: CFormView(CFrmSIM::IDD)

//{{AFX_DATA_INIT(CFrmSIM)
m_s?S_FM1 = __T("");
m_s?S_DS1 = __T("");
m_s?S_DS2 = __T("");
m_s?S_DS3 = __T("");
m_s?S_DS4 = __T("");
m_s?S_DS5 = __T("");
m_s?S_FM2 = __T("");
m_s?S_FM3 = __T("");
m_s?S_FM4 = __T("");
m_s?S_FM5 = __T("");
m_s?S_OUT1 = __T("");
m_s?S_OUT2 = __T("");
m_s?S_REDM1 = __T("");
m_s?S_REDM2 = __T("");
m_s?S_REDM3 = __T("");
m_s?S_REDM4 = __T("");
m_s?S_REDM5 = __T("");
m_s?S_SM1 = __T("");
m_s?S_SM2 = __T("");
m_s?S_SM3 = __T("");
m_lDS1 = 0;
m_lDS2 = 0;
m_lDS3 = 0;
m_lDS4 = 0;
m_lDS5 = 0;
m_bCCS = FALSE;
m_bCS = FALSE;
m_bUPDT_IN = FALSE;
m_strSIMComment = _T("");
m_dLStep = 25.0f;
m_dSStep = 2.0f;
m_s?S_FRC1 = __T("");
m_s?S_FRC2 = __T("");
//}}AFX_DATA_INIT


SIM::~CFrmSIM()



 CFrmSIM::DoDataExchange(CDataExchange* pDX)

CFormView::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CFrmSIM)
DDX_Control(pDX, IDC_RICHEDIT_SIMComment, m_rect1SIMComment);
DDX_OCText(pDX, IDC_NUMLEDCTRL_FM1, DISPID(23), m_s?S_FM1);
DDX_OCText(pDX, IDC_NUMLEDCTRL_DIPSw2, DISPID(23), m_s?S_DS2);
DDX_OCText(pDX, IDC_NUMLEDCTRL_DIPSw3, DISPID(23), m_s?S_DS3);
DDX_OCText(pDX, IDC_NUMLEDCTRL_DIPSw4, DISPID(23), m_s?S_DS4);
DDX_OCText(pDX, IDC_NUMLEDCTRL_DIPSw5, DISPID(23), m_s?S_DS5);
```

```
DDX_OCInt(pDX, IDC_LEDCTRL_DIPSw3, DISPID(29), m_lDS3);
DDX_OCInt(pDX, IDC_LEDCTRL_DIPSw4, DISPID(29), m_lDS4);
DDX_OCInt(pDX, IDC_LEDCTRL_DIPSw5, DISPID(29), m_lDS5);
DDX_OCBool(pDX, IDC_TOGGLECTRL_CCS, DISPID(25), m_bCCS);
DDX_OCBool(pDX, IDC_TOGGLECTRL_CS, DISPID(25), m_bCS);
DDX_OCBool(pDX, IDC_TOGGLECTRL_UPDT_IN, DISPID(25), m_bUPDT_IN);
DDX_Text(pDX, IDC_RICHEDIT_SIMComment, m_strSIMComment);
DDX_OCFloat(pDX, IDC_KNOBCTRL_SIMLStep, DISPID(40), m_dLStep);
DDX_OCFloat(pDX, IDC_KNOBCTRL_SIMSStep, DISPID(40), m_dSStep);
DDX_OCText(pDX, IDC_NUMLEDCTRL_FRC1, DISPID(23), m_s?S_FRC1);
DDX_OCText(pDX, IDC_NUMLEDCTRL_FRC2, DISPID(23), m_s?S_FRC2);
//}}AFX_DATA_MAP



_MESSAGE_MAP(CFrmSIM, CFormView)
//{{AFX_MSG_MAP(CFrmSIM)
ON_BN_CLICKED(IDC_BUTTON_SIMConReset, OnBUTTONSIMConReset)
ON_BN_CLICKED(IDC_BUTTON_SIMConSStep, OnBUTTONSIMConSStep)
ON_BN_CLICKED(IDC_BUTTON_SIMConLStep, OnBUTTONSIMConLStep)
ON_COMMAND(ID_VIEW_TEXT_LARGE, OnViewTextLarge)
ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_LARGE, OnUpdateViewTextLarge)
ON_COMMAND(ID_VIEW_TEXT_LARGEST, OnViewTextLargest)
ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_LARGEST, OnUpdateViewTextLargest)
ON_COMMAND(ID_VIEW_TEXT_MEDIUM, OnViewTextMedium)
ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_MEDIUM, OnUpdateViewTextMedium)
ON_COMMAND(ID_VIEW_TEXT_SMALL, OnViewTextSmall)
ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_SMALL, OnUpdateViewTextSmall)
ON_COMMAND(ID_VIEW_TEXT_SMALLEST, OnViewTextSmallest)
ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_SMALLEST, OnUpdateViewTextSmallest)
//}}AFX_MSG_MAP
MESSAGE_MAP()

_EVENTSINK_MAP(CFrmSIM, CFormView)
//{{AFX_EVENTSINK_MAP(CFrmSIM)
ON_EVENT(CFrmSIM, IDC_STRIPCTRL_Clk, 1 /* CursorChange */, OnCursorChangeS
Clk, VTS_R8)
ON_EVENT(CFrmSIM, IDC_LEDCTRL_DIPSw1, -600 /* Click */, OnClickLEDCTRLDIPS
NONE)
ON_EVENT(CFrmSIM, IDC_LEDCTRL_DIPSw2, -600 /* Click */, OnClickLEDCTRLDIPS
NONE)
ON_EVENT(CFrmSIM, IDC_LEDCTRL_DIPSw3, -600 /* Click */, OnClickLEDCTRLDIPS
NONE)
ON_EVENT(CFrmSIM, IDC_LEDCTRL_DIPSw4, 1 /* Change */, OnChangeLEDCTRLDIPSw
ONE)
ON_EVENT(CFrmSIM, IDC_LEDCTRL_DIPSw5, 1 /* Change */, OnChangeLEDCTRLDIPSw
ONE)
ON_EVENT(CFrmSIM, IDC_TOGGLECTRL_CCS, -600 /* Click */, OnClickTogglectrl(
NONE)
ON_EVENT(CFrmSIM, IDC_TOGGLECTRL_CS, -600 /* Click */, OnClickTogglectrlCs
IE)
ON_EVENT(CFrmSIM, IDC_TOGGLECTRL_UPDT_IN, -600 /* Click */, OnClickToggleC
, VTS_NONE)
//}}AFX_EVENTSINK_MAP
```

```
IDEDoc* CFrmSIM::GetDocument() // non-debug version is inline

 ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CFIDEDoc)));
 return (CFIDEDoc*)m_pDocument;

dif //_DEBUG

////////////////////////////////////////////////////////////////////////////
CFrmSIM internal implementation

Take Input's value for calculation
d CFrmSIM::GetInput(CDtSIM* pxSIM, int idx, BYTE Input[4])

 // set common index for setting value
 pxSIM->SetIndex(idx);
 // configuration setting
 // --- CS flag
 if (m_bCS)
     pxSIM->SetCS(1);
 else
     pxSIM->SetCS(0);

 // --- CCS flag
 if (m_bCCS)
     pxSIM->SetCCS(1);
 else
     pxSIM->SetCCS(0);

 // --- UPDT_IN flag
 if (m_bUPDT_IN)
     pxSIM->SetUPDT_IN(1);
 else
     pxSIM->SetUPDT_IN(0);

 BOOL check = UpdateData(TRUE);
 assert(check);
 // Input values
 Input[0] = (BYTE) m_lDS2;
 Input[1] = (BYTE) m_lDS3;
 Input[2] = (BYTE) m_lDS4;
 Input[3] = (BYTE) m_lDS5;


Show calculation result on the graphics
d CFrmSIM::ShowOutput(CDtSIM* pxSIM, int idx)

 CString strTmp;
 BYTE nCon = 0;

 // Configuration Dip Switch & Buttons
 // --- Clock Select
 if (pxSIM->GetCS(idx))
 {
     m_bCS = TRUE;
```

```
}
else
{
    m_bUPDT_IN = FALSE;
    // Input Dip Switches
    m_lDS2 = pxSIM->GetInput(idx, 0); // Input #0
    m_lDS3 = pxSIM->GetInput(idx, 1); // Input #1
    m_lDS4 = pxSIM->GetInput(idx, 2); // Input #2
    m_lDS5 = pxSIM->GetInput(idx, 3); // Input #3
}

// Input 7-Segments
strTmp.Format("%.2X", (BYTE) m_lDS2); // Input #0
m_s7S_DS2 = strTmp;
strTmp.Format("%.2X", (BYTE) m_lDS3); // Input #1
m_s7S_DS3 = strTmp;
strTmp.Format("%.2X", (BYTE) m_lDS4); // Input #2
m_s7S_DS4 = strTmp;
strTmp.Format("%.2X", (BYTE) m_lDS5); // Input #3
m_s7S_DS5 = strTmp;

// Clocks' waveforms
m_stripClk.AddXY(0, idx, pxSIM->GetMCLK(idx));
m_stripClk.AddXY(1, idx, pxSIM->GetI0CLK(idx));
if (idx == 0)
{
    m_stripClk.AddXY(0, idx, pxSIM->GetMCLK(idx));
    m_stripClk.AddXY(1, idx, pxSIM->GetI0CLK(idx));
}

// Storage Module 7-Segments
strTmp.Format("%.2X", pxSIM->GetCAA(idx));
m_s7S_SM1 = strTmp;
strTmp.Format("%.2X", pxSIM->GetCDBUS(idx));
m_s7S_SM2 = strTmp;
strTmp.Format("%.2X", pxSIM->GetINBUS(idx));
m_s7S_SM3 = strTmp;

// Fuzzy Rule Counters' 7-Segments
strTmp.Format("%.1X", pxSIM->GetTrmCnt(idx));
m_s7S_FRC1 = strTmp;
strTmp.Format("%.2X", pxSIM->GetRleCnt(idx));
m_s7S_FRC2 = strTmp;

// Fuzzifier Module's 7-Segments
strTmp.Format("%.1X", pxSIM->GetTYPE(idx));
m_s7S_FM1 = strTmp;
strTmp.Format("%.2X", pxSIM->GetINPUT(idx));
m_s7S_FM2 = strTmp;
strTmp.Format("%.2X", pxSIM->GetCENTER(idx));
m_s7S_FM3 = strTmp;
strTmp.Format("%.2X", pxSIM->GetWIDTH(idx));
m_s7S_FM4 = strTmp;
strTmp.Format("%.2X", pxSIM->GetFUZZ(idx));
```

```
 strTmp.Format("%.2X", pxSIM->GetOutput(idx, pxSIM->GetCAB(idx)));
 m_s7S_OUT2 = strTmp;

 // Simulation Process' Comment
 m_strSIMComment = pxSIM->GetComment(idx);

 BOOL check = UpdateData(FALSE);
 assert(check);



////////////////////////////////////////////////////////////////////////////
CFrmSIM message handlers

d CFrmSIM::OnInitialUpdate()

 CFormView::OnInitialUpdate();

 // Graphic initialization
 // --- rich-edit control text formatting
 m_cf.cbSize     = sizeof(CHARFORMAT);
 m_cf.dwMask     = CFM_SIZE | CFM_BOLD;
 m_cf.dwEffects = CFE_BOLD;
 m_cf.yHeight    = nMEDIUMFONT*20;
 m_rectlSIMComment.SetDefaultCharFormat(m_cf);

 // --- GUI component intialization
 CFIDEDoc* pDoc = GetDocument();
 if(pDoc->m_xSIM.GetIndex()>0)
 {
     for(int i = 0; i < pDoc->m_xSIM.GetIndex(); i++)
     {
         m_stripClk.AddXY(0, i, pDoc->m_xSIM.GetMCLK(i));
         m_stripClk.AddXY(1, i, pDoc->m_xSIM.GetIOCLK(i));
     }
     m_stripClk.SetCursorX(pDoc->m_xSIM.GetIndex()-1);
     ShowOutput(&(pDoc->m_xSIM), pDoc->m_xSIM.GetIndex()-1);
 }
 else
 {
     BYTE baTmp[4];
     pDoc->Simulate(pDoc->m_xSIM.GetIndex(), baTmp);
 ShowOutput(&(pDoc->m_xSIM), 0);
 }


--- Inputs Interfaces ----------------------------------------------
d CFrmSIM::OnClickLEDCTRLDIPSw1()

 BOOL check = UpdateData(TRUE);
 assert(check);

 // (Master) Clock Select Flag
 if ((BYTE)m_lDS1 & nMASK_CS)
```

```
assert(check);


void CFrmSIM::OnClickLEDCTRLDIPSw2()
{
CString strTmp;

BOOL check = UpdateData(TRUE);
assert(check);
// Input #0 Interface 7-Segment
strTmp.Format("%.2X", (BYTE) m_lDS2);
m_s7S_DS2 = strTmp;

check = UpdateData(FALSE);
assert(check);
}


void CFrmSIM::OnClickLEDCTRLDIPSw3()
{
CString strTmp;

BOOL check = UpdateData(TRUE);
assert(check);
// Input #1 Interface 7-Segment
strTmp.Format("%.2X", (BYTE) m_lDS3);
m_s7S_DS3 = strTmp;

check = UpdateData(FALSE);
assert(check);
}


void CFrmSIM::OnChangeLEDCTRLDIPSw4()
{
CString strTmp;

BOOL check = UpdateData(TRUE);
assert(check);
// Input #2 Interface 7-Segment
strTmp.Format("%.2X", (BYTE) m_lDS4);
m_s7S_DS4 = strTmp;

check = UpdateData(FALSE);
assert(check);
}


void CFrmSIM::OnChangeLEDCTRLDIPSw5()
{
CString strTmp;

BOOL check = UpdateData(TRUE);
assert(check);
// Input #3 Interface 7-Segment
strTmp.Format("%.2X", (BYTE) m_lDS5);
m_s7S_DS5 = strTmp;
```

```
d CFrmSIM::0nClickTogglectrlCcs()

 B00L check = UpdateData(TRUE);
 assert(check);
 if (m_bCCS)
     m_lDSl = m_lDSl | nMASK_CCS;
 else
     m_lDSl = m_lDSl & ~nMASK_CCS;

 check = UpdateData(FALSE);
 assert(check);


d CFrmSIM::0nClickTogglectrlUpdtIn()

 B00L check = UpdateData(TRUE);
 assert(check);
 if (m_bUPDT_IN)
     m_lDSl = m_lDSl | nMASK_UPDT_IN;
 else
     m_lDSl = m_lDSl & ~nMASK_UPDT_IN;

 check = UpdateData(FALSE);
 assert(check);


d CFrmSIM::0nCursorChangeSTRIPCTRLClk(double X)

 CFIDEDoc* pDoc = GetDocument();
 Show0utput(&(pDoc->m_xSIM), (int) m_stripClk.GetCursorX());


--- Controls Interfaces ------------------------------------------

Short Step Simulation
d CFrmSIM::0nBUTT0NSIMConSStep()

 CFIDEDoc* pDoc = GetDocument();
 for (int i = 0; i < (int)m_dSStep; i++)
 {
     if (pDoc->m_xSIM.GetIndex() == pDoc->m_xSIM.GetSize())
         pDoc->m_xSIM.SetSize(pDoc->m_xSIM.GetSize()+nSIM_INC);
     GetInput(&(pDoc->m_xSIM), pDoc->m_xSIM.GetIndex(), m_baInput);
     pDoc->Simulate(pDoc->m_xSIM.GetIndex(), m_baInput);
     Show0utput(&(pDoc->m_xSIM), pDoc->m_xSIM.GetIndex());
     pDoc->m_xSIM.SetIndex(pDoc->m_xSIM.GetIndex()+1);
 }


Long step simulation
d CFrmSIM::0nBUTT0NSIMConLStep()

 CFIDEDoc* pDoc = GetDocument();
```

```
 ShowOutput(&(pDoc->m_xSIM), 0);

 // resetting clocks' chart
 m_stripClk.ClearTrack(0);
 m_stripClk.ClearTrack(1);
 m_stripClk.SetLastX(5);


--- Data Interfaces ---------------------------------------------
d CFrmSIM::OnBUTTONSIMDataSave()

 // TODO: Add your control notification handler code here


d CFrmSIM::OnBUTTONSIMDataLoad()

 // TODO: Add your control notification handler code here


d CFrmSIM::OnViewTextLargest()

 m_cf.yHeight    = nLARGESTFONT*20;
 m_rectlSIMComment.SetDefaultCharFormat(m_cf);


d CFrmSIM::OnUpdateViewTextLargest(CCmdUI* pCmdUI)

 pCmdUI->SetCheck((m_cf.yHeight == nLARGESTFONT*20)? 1:0);


d CFrmSIM::OnViewTextLarge()

 m_cf.yHeight    = nLARGEFONT*20;
 m_rectlSIMComment.SetDefaultCharFormat(m_cf);


d CFrmSIM::OnUpdateViewTextLarge(CCmdUI* pCmdUI)

 pCmdUI->SetCheck((m_cf.yHeight == nLARGEFONT*20)? 1:0);


d CFrmSIM::OnViewTextMedium()

 m_cf.yHeight    = nMEDIUMFONT*20;
 m_rectlSIMComment.SetDefaultCharFormat(m_cf);


d CFrmSIM::OnUpdateViewTextMedium(CCmdUI* pCmdUI)

 pCmdUI->SetCheck((m_cf.yHeight == nMEDIUMFONT*20)? 1:0);
```

```
d CFrmSIM::OnUpdateViewTextSmallest(CCmdUI* pCmdUI)

 pCmdUI->SetCheck((m_cf.yHeight == nSMALLESTFONT*20)? 1:0);
```

```
Attributes
lic:
  // Input's name & initial value
  CTypedPtrArray<C0bArray, CDtI0*> m_xaInput;
  // 0utput's name & initial value
  CTypedPtrArray<C0bArray, CDtI0*> m_xa0utput;
  // Term's name, input, type, width & center
  CTypedPtrArray<C0bArray, CDtTrm*> m_xaTerm;
  // Rule's terms, type, output & action value
  CTypedPtrArray<C0bArray, CDtR1*> m_xaRule;
  // Converted fuzzy's parameters & rules
  CDtCnvrtd m_xConverted;
  // Simulation data
  CDtSIM m_xSIM;

0perations
lic:
  // Convert fuzzy parameters & rules to FLC internal code
  B00L Convert();
  // Generate 4 .mem files from converted data for pre-synthesized design
  B00L GenerateMEMFile(int index, CArchive& ar);
  // Generate a .hex file from converted data for post-synthesized design
  B00L GenerateHEXFile(CArchive& ar);

  // Single step simulation
  B00L Simulate(int index, BYTE Input[4]);
  // Save simulation data
  B00L SaveSIMData();

0verrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CFIDEDoc)
  public:
  virtual B00L 0nNewDocument();
  virtual void Serialize(CArchive& ar);
  virtual void DeleteContents();
  virtual B00L 0n0penDocument(LPCTSTR lpszPathName);
  //}}AFX_VIRTUAL

Implementation
lic:
  virtual ~CFIDEDoc();
def _DEBUG
  virtual void AssertValid() const;
  virtual void Dump(CDumpContext& dc) const;
dif

Generated message map functions
tected:
  //{{AFX_MSG(CFIDEDoc)
      // N0TE - the ClassWizard will add and remove member functions here.
      //    D0 N0T EDIT what you see in these blocks of generated code !
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
```

```
GIN_MESSAGE_MAP(CFIDEDoc, CDocument)
  //{{AFX_MSG_MAP(CFIDEDoc)
      // NOTE - the ClassWizard will add and remove mapping macros here.
      //    DO NOT EDIT what you see in these blocks of generated code!
  //}}AFX_MSG_MAP
)_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
 CFIDEDoc construction/destruction

IDEDoc::CFIDEDoc()

  // TODO: add one-time construction code here


IDEDoc::~CFIDEDoc()

  for (int i = 0; i < m_xaInput.GetSize(); i++)
      delete m_xaInput[i];
  m_xaInput.RemoveAll();

  for (i = 0; i < m_xaOutput.GetSize(); i++)
      delete m_xaOutput[i];
  m_xaOutput.RemoveAll();

  for (i = 0; i < m_xaTerm.GetSize(); i++)
      delete m_xaTerm[i];
  m_xaTerm.RemoveAll();

  for (i = 0; i < m_xaRule.GetSize(); i++)
      delete m_xaRule[i];
  m_xaRule.RemoveAll();


OL CFIDEDoc::OnNewDocument()

  if (!CDocument::OnNewDocument())
      return FALSE;

  // Initialize Input data
  CDtIO* pDtIn = new CDtIO();
  pDtIn->SetName(_T("Input1"));
  pDtIn->SetInit(5);
  m_xaInput.Add(pDtIn);

  // Initialize Output data
  CDtIO* pDtOut = new CDtIO();
  pDtOut->SetName(_T("Output1"));
  pDtOut->SetInit(5);
  m_xaOutput.Add(pDtOut);

  // Initialize Term data
  CDtTrm* pDtTrm = new CDtTrm();
```

```
{
    AfxMessageBox(IDS_CONVERSION_ERROR);
    return FALSE;
}

return TRUE;


OL CFIDEDoc::OnOpenDocument(LPCTSTR lpszPathName)

if (!CDocument::OnOpenDocument(lpszPathName))
    return FALSE;

// Convert data
if (!Convert())
{
    AfxMessageBox(IDS_CONVERSION_ERROR);
    return FALSE;
}

return TRUE;


/////////////////////////////////////////////////////////////////////////////
CFIDEDoc serialization

id CFIDEDoc::Serialize(CArchive& ar)

m_xaInput.Serialize(ar);
m_xaOutput.Serialize(ar);
m_xaTerm.Serialize(ar);
m_xaRule.Serialize(ar);


/////////////////////////////////////////////////////////////////////////////
CFIDEDoc diagnostics

fdef _DEBUG
id CFIDEDoc::AssertValid() const

CDocument::AssertValid();


id CFIDEDoc::Dump(CDumpContext& dc) const

CDocument::Dump(dc);

ndif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
CFIDEDoc commands

id CFIDEDoc::DeleteContents()
```

```
Convert fuzzy parameters & rules to FLC internal code
)L CFIDEDoc::Convert()

 CString strTmp;
 m_xConverted.ResetContents();

 // Convert Fuzzy parameters & terms to FLC internal code
 // ---------- Convert Input parameters
 for (int i = 0; i < m_xaInput.GetSize(); i++)
 {
     // ----- Label
     m_xConverted.SetLabel(i, m_xaInput.ElementAt(i)->GetName());
     // ----- Initial Value
     m_xConverted.SetValue(i, m_xaInput.ElementAt(i)->GetInit());
 }

 // ---------- Convert Output parameters
 for (i = 0; i < m_xaOutput.GetSize(); i++)
 {
     // ----- Label
     m_xConverted.SetLabel(i+16, m_xaOutput.ElementAt(i)->GetName());
     // ----- Initial Value
     m_xConverted.SetValue(i+16, m_xaOutput.ElementAt(i)->GetInit());
 }

 // ---------- Convert Term parameters
 int n = m_xaTerm.GetSize();
 m_xConverted.saTrmLbl.SetSize(n, 1);
 m_xConverted.baTrmTyp.SetSize(n, 1);
 m_xConverted.baTrmInp.SetSize(n, 1);
 m_xConverted.baTrmCnt.SetSize(n, 1);
 m_xConverted.baTrmWdt.SetSize(n, 1);
 for (i = 0; i < m_xaTerm.GetSize(); i++)
 {
     // ----- Label
     strTmp = m_xaTerm.ElementAt(i)->GetName();
     strTmp.MakeUpper();
     m_xConverted.saTrmLbl[i] = strTmp;
     // ----- Type
     m_xConverted.baTrmTyp[i] = m_xaTerm.ElementAt(i)->GetType();
     // ----- Input
     n = m_xConverted.FindLabel(m_xaTerm.ElementAt(i)->GetInput());
     assert(n != -1);
     m_xConverted.baTrmInp[i] = n;
     // ----- Center
     n = m_xConverted.Find0rAddLabel(m_xaTerm.ElementAt(i)->GetCenter());
     assert(n != -1);
     m_xConverted.baTrmCnt[i] = n;
     // ----- Width
     n = m_xConverted.Find0rAddLabel(m_xaTerm.ElementAt(i)->GetWidth());
     assert(n != -1);
     m_xConverted.baTrmWdt[i] = n;
```

```
    }
    // ----- Give End of Rule code
    n = m_xaRule.ElementAt(i)->GetType();
    if (n == 2)
        n = 3;
    m_xConverted.SetRule(nRlIdx++, nEOR_CODE | (n << 4));
    n = m_xConverted.FindOrAddLabel(m_xaRule.ElementAt(i)->GetAction());
    assert(n!= -1);
    m_xConverted.SetRule(nRlIdx++, n); // Action Value's address
    // ----- Give End of Rule Reffering Specific Output code ?
    if (i == m_xaRule.GetSize()-1)
        n = 1;
    else if (m_xaRule.ElementAt(i)->GetOutput() !=
             m_xaRule.ElementAt(i+1)->GetOutput())
             n = 1;
    else
        n = 0;
    if (n == 1)
    {
        m_xConverted.SetRule(nRlIdx++, nEORRS0_CODE);
        n = m_xConverted.FindLabel(m_xaRule.ElementAt(i)->GetOutput());
        assert(n!= -1);
        m_xConverted.SetRule(nRlIdx++, n);
    }
}

// If Rule hasn't full yet, give End of Rules code
if (nRlIdx < nRULESUM+1)
    m_xConverted.SetRule(nRlIdx, 8);

return TRUE;


Generate 4 .mem files from converted data for pre-synthesized design
OL CFIDEDoc::GenerateMEMFile(int index, CArchive& ar)

int i;
CTime t = CTime::GetCurrentTime();
CString strTmp = t.Format("; Created on %#c\r\n");
switch (index)
{
case 1:
    ar.WriteString(";\r\n");
    ar.WriteString("; memfile par1.mem for LogiBLOX symbol f_m168d1\r\n"
    ar.WriteString((LPCTSTR) strTmp);
    ar.WriteString(";\r\n");
    ar.WriteString("; Header Section\r\n");
    ar.WriteString("RADIX 10\r\nDEPTH 16\r\nWIDTH 8\r\nDEFAULT 0\r\n");
    ar.WriteString(";\r\n");
    ar.WriteString("; Data Section\r\n");
    ar.WriteString("; Specifies data to be stored in different addresses

    ar.WriteString("; e.g., DATA 0:A, 1:0\r\n");
    ar.WriteString("RADIX 10\r\nDATA");
```

```
    break;
case 2:
    ar.WriteString(";\r\n");
    ar.WriteString("; memfile par2.mem for LogiBL0X symbol f_m168d2\r\n
    ar.WriteString((LPCTSTR) strTmp);
    ar.WriteString(";\r\n");
    ar.WriteString("; Header Section\r\n");
    ar.WriteString("RADIX 10\r\nDEPTH 16\r\nWIDTH 8\r\nDEFAULT 0\r\n");
    ar.WriteString(";\r\n");
    ar.WriteString("; Data Section\r\n");
    ar.WriteString("; Specifies data to be stored in different addresse

    ar.WriteString("; e.g., DATA 0:A, 1:0\r\n");
    ar.WriteString("RADIX 10\r\nDATA");
    for (i = 0; i < nI0PAR/8; i++)
    {
        for (int j = 0; j < 8; j++)
            if (i*8+j == nI0PAR-1)
            {
                strTmp.Format(" %3i", m_xConverted.GetValue(nI0PAR+i*8+
                ar.WriteString(strTmp);
            }
            else
            {
                strTmp.Format(" %3i,", m_xConverted.GetValue(nI0PAR+i*8
                ar.WriteString(strTmp);
            }
        if (i != nI0PAR/8-1)
            ar.WriteString("\r\n    ");
    }
    ar.WriteString("\r\n; end of LogiBL0X memfile");
    break;
case 3:
    ar.WriteString(";\r\n");
    ar.WriteString("; memfile par3.mem for LogiBL0X symbol f_m328s\r\n"
    ar.WriteString((LPCTSTR) strTmp);
    ar.WriteString(";\r\n");
    ar.WriteString("; Header Section\r\n");
    ar.WriteString("RADIX 10\r\nDEPTH 32\r\nWIDTH 8\r\nDEFAULT 0\r\n");
    ar.WriteString(";\r\n");
    ar.WriteString("; Data Section\r\n");
    ar.WriteString("; Specifies data to be stored in different addresse

    ar.WriteString("; e.g., DATA 0:A, 1:0\r\n");
    ar.WriteString("RADIX 10\r\nDATA");
    for (i = 0; i < nFIXPAR/8; i++)
    {
        for (int j = 0; j < 8; j++)
            if (i*8+j == nFIXPAR-1)
            {
                strTmp.Format(" %3i", m_xConverted.GetValue(nFIXPAR+i*8
                ar.WriteString(strTmp);
            }
            else
```

```
    ar.WriteString("; Data Section\r\n");
    ar.WriteString("; Specifies data to be stored in different addresse

    ar.WriteString("; e.g., DATA 0:A, 1:0\r\n");
    ar.WriteString("RADIX 10\r\nDATA");
    for (i = 0; i < nRULESUM/16; i++)
    {
        for (int j = 0; j < 16; j++)
            if (i*16+j == nRULESUM-1)
            {
                strTmp.Format(" %2i", m_xConverted.GetRule(i*16+j));
                ar.WriteString(strTmp);
            }
            else
            {
                strTmp.Format(" %2i,", m_xConverted.GetRule(i*16+j));
                ar.WriteString(strTmp);
            }
        if (i != nRULESUM/16-1)
            ar.WriteString("\r\n    ");
    }
    ar.WriteString("\r\n; end of LogiBL0X memfile");
    break;
default:
    break;
}

return TRUE;


Generate a .hex file from converted data for post-synthesized design
OL CFIDEDoc::GenerateHEXFile(CArchive& ar)

int nAddr     = nBASE_ADDRESS - 16;
int nCheckSum = 0;
CString strTmp;
for (int i = 0; i < nRULESUM/16; i++)
{
    nAddr += 16;
    nCheckSum = 16 + (nAddr&0xFF) + ((nAddr>>8)&0xFF) + nDATA_RECORD;
    for (int j = 0; j < 16; j++)
        nCheckSum += m_xConverted.GetRule(i*16+j);
    strTmp.Format(":10%.4X00", nAddr);
    ar.WriteString((LPCTSTR) strTmp);
    for (j = 0; j < 16; j++)
    {
        strTmp.Format("%.2X", m_xConverted.GetRule(i*16+j));
        ar.WriteString((LPCTSTR) strTmp);
    }
    strTmp.Format("%.2X", (-nCheckSum)&0xFF);
    ar.WriteString((LPCTSTR) strTmp);
    ar.WriteString("\r\n");
}
for (i = 0; i < nPARSUM/16; i++)
```

```
 return TRUE;


Single step simulation
OL CFIDEDoc::Simulate(int idx, BYTE Input[4])

 // Set configuration flags
 BYTE CS      = m_xSIM.GetCS(idx);          // Clock Select
 BYTE CCS     = m_xSIM.GetCCS(idx);         // Counter Clock Select
 BYTE UPDT_IN = m_xSIM.GetUPDT_IN(idx);     // Update Input Flag

 // Set common index for setting all array variables
 m_xSIM.SetIndex(idx);

 // --- Clocks Calculation ----------------------------------------
 // ------ MCLK calculation ------------------------------------------
 if ((idx == 0) || ((idx == 1) && (CS)))    // initial state
     m_xSIM.SetMCLK(0);
 else
     if (m_xSIM.GetMCLK(idx-(1+CS)))        // L0 state
         m_xSIM.SetMCLK(0);
     else
         m_xSIM.SetMCLK(1);                 // HI state

 // ------ I0CLK calculation -----------------------------------------
 if ((idx < 4) || ((idx < 8) && (CCS)))     // initial state
     m_xSIM.SetI0CLK(0);
 else
     if (m_xSIM.GetI0CLK(idx-(4+4*CCS)))    // L0 state
         m_xSIM.SetI0CLK(0);
     else
         m_xSIM.SetI0CLK(1);                // HI state

 // --- I/0 Calculation -------------------------------------------
 // ------ CAB calculation -------------------------------------------
 if (idx == 0)                              // initial state
     m_xSIM.SetCAB(0);
 else
     // increment on positif transition of I0CLK
     if ((m_xSIM.GetI0CLK(idx)) && (!m_xSIM.GetI0CLK(idx-1)))
         m_xSIM.SetCAB(m_xSIM.GetCAB(idx-1)+1);
     // copy CAB from previous value
     else
         m_xSIM.SetCAB(m_xSIM.GetCAB(idx-1));
 // I/0 limitation
 if (m_xSIM.GetCAB(idx) == nI0SUM)
     m_xSIM.SetCAB(0);

 // ------ Input Port update -----------------------------------------
 if (idx == 0)
     // initial state, fill all Inputs with initial value
     for (int i = 0; i < nI0SUM; i++)
         m_xSIM.SetInput(i, m_xConverted.GetValue(i));
```

```
else
    // copy output value from previous value
    for (int i = 0; i < nI0SUM; i++)
        m_xSIM.Set0utput(i, m_xSIM.Get0utput(idx-1, i));

// --- FLC Calculation ----------------------------------------
// ------ State calculation -------------------------------------
if (idx == 0)
    m_xSIM.SetState(St_Init);
else
    // update State on positif transition
    if ((m_xSIM.GetMCLK(idx)) && (!m_xSIM.GetMCLK(idx-1)))
        if (!m_xSIM.UpdateState(idx))
        {
            AfxMessageBox(IDS_UNREC0GNIZED_STATE);
            return FALSE;
        }

// ------ CAA calculation ---------------------------------------
if (idx == 0) // initial state
    m_xSIM.SetCAA(0);
else
    // update CAA on positif transition
    if ((m_xSIM.GetMCLK(idx)) && (!m_xSIM.GetMCLK(idx-1)))
        m_xSIM.UpdateCAA(idx);
    // copy CAA from previous value
    else m_xSIM.SetCAA(m_xSIM.GetCAA(idx-1));

// ------ CDBUS update ------------------------------------------
m_xSIM.SetCDBUS(m_xConverted.GetRule(m_xSIM.GetCAA(idx)));

// ------ INBUS update ------------------------------------------
int i = m_xSIM.GetCDBUS(idx);
// --- read Input value
if (i < nI0SUM)
    m_xSIM.SetINBUS(m_xSIM.GetInput(idx, i));
else
    // --- read 0utput value
    if ((i >= nI0PAR) && (i < nI0PAR+nI0SUM))
        m_xSIM.SetINBUS(m_xSIM.Get0utput(idx, i-nI0PAR));
    else
        // --- read fix parameter value
        m_xSIM.SetINBUS(
        m_xConverted.GetValue(m_xSIM.GetCDBUS(idx)));

// ------ TYPE update -------------------------------------------
if (idx == 0) // initial type
    m_xSIM.SetTYPE(m_xSIM.GetCDBUS(idx)&nMASK_TYPE);
else
    // update TYPE on negatif transition in St_Fetch State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_Fetch))
        m_xSIM.SetTYPE(m_xSIM.GetCDBUS(idx)&nMASK_TYPE);
    // copy TYPE from previous value
```

```
    m_xSIM.SetCENTER(0);
else
    // update CENTER on negatif transition in St_Ld_Cnt State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_Ld_Cnt))
        m_xSIM.SetCENTER(m_xSIM.GetINBUS(idx));
    // copy CENTER from previous value
    else
        m_xSIM.SetCENTER(m_xSIM.GetCENTER(idx-1));

// ------ WIDTH update ------------------------------------------
if (idx == 0)
    m_xSIM.SetWIDTH(0);
else
    // update WIDTH on negatif transition in St_Ld_Wdt State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_Ld_Wdt))
        m_xSIM.SetWIDTH(m_xSIM.GetINBUS(idx));
    // copy WIDTH from previous value
    else
        m_xSIM.SetWIDTH(m_xSIM.GetWIDTH(idx-1));

// ------ FUZZ calculation --------------------------------------
if (idx == 0)
    m_xSIM.SetFUZZ(0);
else
    // update FUZZ on negatif transition in St_En_AlfCt State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_En_AlfCt))
    {
        if (!m_xSIM.UpdateFUZZ(idx))
        {
            AfxMessageBox(IDS_UNRECOGNIZED_MF);
            return FALSE;
        }
    }
    // copy FUZZ from previous value
    else
        m_xSIM.SetFUZZ(m_xSIM.GetFUZZ(idx-1));

// ------ MIN_FUZZ calculation ----------------------------------
if (idx == 0)
    m_xSIM.SetMIN_FUZZ(nMAX_FUZZ);
else
    // update MIN_FUZZ on neg. trans. in St_En_Min / St_En_Max State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_En_Min))
        m_xSIM.UpdateMIN_FUZZ(idx);
    // copy WIDTH from previous value
    else
        m_xSIM.SetMIN_FUZZ(m_xSIM.GetMIN_FUZZ(idx-1));

// ------ MAX_FUZZ, DTYPE & ACT calculation ---------------------
if (idx == 0)
```

```
        m_xSIM.SetDTYPE(m_xSIM.GetDTYPE(idx-1));
        m_xSIM.SetACT(m_xSIM.GetACT(idx-1));
    }

/*/ ------ DTYPE calculation ----------------------------------------
if (idx == 0)
    m_xSIM.SetDTYPE(0);
else
    // update DTYPE on negatif transition in St_En_Max State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_En_Max))
        m_xSIM.SetDTYPE(
          (m_xSIM.GetMAX_FUZZ(idx) == m_xSIM.GetMAX_FUZZ(idx-1)) ?
           m_xSIM.GetDTYPE(idx-1) :
           m_xSIM.GetCDBUS(idx-4-4*m_xSIM.GetCS(idx))>>4);
    // copy DTYPE from previous value
    else
        m_xSIM.SetDTYPE(m_xSIM.GetDTYPE(idx-1));

// ------ ACT calculation ----------------------------------------
if (idx == 0)
    m_xSIM.SetACT(0);
else
    // update ACT on negatif transition in St_En_Max State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_En_Max))
        m_xSIM.SetACT(
          (m_xSIM.GetMAX_FUZZ(idx) == m_xSIM.GetMAX_FUZZ(idx-1)) ?
           m_xSIM.GetACT(idx-1) : m_xSIM.GetINBUS(idx-2));
    // copy ACT from previous value
    else
        m_xSIM.SetACT(m_xSIM.GetACT(idx-1));*/

// ------ 0UTBUS calculation ----------------------------------------
if (idx == 0)
    m_xSIM.Set0UTBUS(0);
else
    // update 0UTBUS on negatif transition in St_En_Dfzzfr State
    if ((!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
        (m_xSIM.GetState() == St_En_Dfzzfr))
    {
        if (!m_xSIM.Update0UTBUS(idx))
        {
            AfxMessageBox(IDS_UNREC0GNIZED_DTYPE);
            return FALSE;
        }
    }
    // copy 0UTBUS from previous value
    else
        m_xSIM.Set0UTBUS(m_xSIM.Get0UTBUS(idx-1));

// ------ Update 0utput registers on St_Wr_0ut state -------------
if ((idx > 0) && (!m_xSIM.GetMCLK(idx)) && (m_xSIM.GetMCLK(idx-1)) &&
    (m_xSIM.GetState() == St_Wr_0ut))
```

```
        m_xSIM.SetTrmCnt(m_xSIM.GetTrmCnt(idx-1)+1);
    else
        // copy term count from previous value
        m_xSIM.SetTrmCnt(m_xSIM.GetTrmCnt(idx-1));

    // reset term count in transition between EOR/EORRS0 & TE stage
    if ((m_xSIM.GetTYPE(idx) > 1) && (m_xSIM.GetTYPE(idx) < 8) &&
        (m_xSIM.GetTYPE(idx-1) < 2))
        m_xSIM.SetTrmCnt(0);
}

// ------ Update Rule# ----------------------------------------
if (idx == 0)
    m_xSIM.SetRleCnt(0);
else
{
    // increase rule count between EOR/EORRS0 & TE stage transition
    if ((m_xSIM.GetTYPE(idx) > 1) && (m_xSIM.GetTYPE(idx) < 8) &&
        (m_xSIM.GetTYPE(idx-1) < 2))
        m_xSIM.SetRleCnt(m_xSIM.GetRleCnt(idx-1)+1);
    else
        // copy rule count from previous value
        m_xSIM.SetRleCnt(m_xSIM.GetRleCnt(idx-1));

    // reset rule count in transition between EORs & TE stage
    if ((m_xSIM.GetTYPE(idx) > 1) && (m_xSIM.GetTYPE(idx) < 8) &&
        (m_xSIM.GetTYPE(idx-1) == 8))
        m_xSIM.SetRleCnt(0);
}

// ------ Adding Comment ----------------------------------------
CString strTmp;
switch(m_xSIM.GetState())
{
    case St_Init      :
        if (!strTmp.LoadString(IDS_ST_INIT))
            return FALSE;
        break;
    case St_Fetch     :
        if (!strTmp.LoadString(IDS_ST_FETCH))
            return FALSE;
        break;

    // --- Term Evaluating Stage
    case St_Ld_In     :
        if (!strTmp.LoadString(IDS_ST_LD_IN))
            return FALSE;
        break;
    case St_Ld_Cnt    :
        if (!strTmp.LoadString(IDS_ST_LD_CNT))
            return FALSE;
        break;
    case St_Ld_Wdt    :
        if (!strTmp.LoadString(IDS_ST_LD_WDT))
```

```
        if (!strTmp.LoadString(IDS_ST_EN_MAX))
            return FALSE;
        break;

    // --- End Of Rule(s) Referring Specific Output Stage
    case St_Ld_Out    :
        if (!strTmp.LoadString(IDS_ST_LD_OUT))
            return FALSE;
        break;
    case St_En_Dfzzfr :
        if (!strTmp.LoadString(IDS_ST_EN_DFZZFR))
            return FALSE;
        break;
    case St_Wr_Out    :
        if (!strTmp.LoadString(IDS_ST_WR_OUT))
            return FALSE;
        break;

    // --- ERROR : unrecognized State
    default           :
        AfxMessageBox(IDS_UNRECOGNIZED_STATE);
        return FALSE;
  }
  m_xSIM.SetComment(strTmp);

  return TRUE;


 Save simulation data
OL CFIDEDoc::SaveSIMData()

  return TRUE;
```

```
nst int nLARGEFONT    = 12;
nst int nLARGESTFONT  = 14;
nst int nSPACESCALER  = 6;

ass CFIDEView : public CScrollView

otected: // create from serialization only
  CFIDEView();
  DECLARE_DYNCREATE(CFIDEView)

 Attributes
blic:
  CFIDEDoc* GetDocument();

ivate:
  LOGFONT m_logFont;

 Operations
otected:
  // Show Fuzzy Parameter
  void ShowFuzzy(CDC *pDC, CFIDEDoc *pDoc);
  // Load Document to Parameter Dialog
  void LoadToDlg(CFIDEParDlg& dlg);
  // Save changes done by Parameter Dialog to Document
  void SaveToDoc(CFIDEParDlg& dlg);
  // Clean unneeded dynamic data created by cancelled Parameter Dialog
  void CleanTemp(CFIDEParDlg& dlg);

 Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CFIDEView)
  public:
  virtual void OnDraw(CDC* pDC);  // overridden to draw this view
  virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
  protected:
  virtual void OnInitialUpdate(); // called first time after construct
  virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
  virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
  virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
  //}}AFX_VIRTUAL

 Implementation
blic:
  virtual ~CFIDEView();
fdef _DEBUG
  virtual void AssertValid() const;
  virtual void Dump(CDumpContext& dc) const;
ndif

otected:

 Generated message map functions
otected:
  //{{AFX_MSG(CFIDEView)
```

```
fndef _DEBUG  // debug version in FIDEView.cpp
line CFIDEDoc* CFIDEView::GetDocument()
 { return (CFIDEDoc*)m_pDocument; }
ndif

////////////////////////////////////////////////////////////////////////////

{{AFX_INSERT_LOCATION}}
 Microsoft Visual C++ will insert additional declarations immediately bef
revious line.

ndif // !defined(AFX_FIDEVIEW_H__8724F824_0C45_11D3_9DCE_00E04CAACD92__IN
```

```
GIN_MESSAGE_MAP(CFIDEView, CScrollView)
  //{{AFX_MSG_MAP(CFIDEView)
  ON_COMMAND(ID_FLC_EDIT, OnFlcEdit)
  ON_COMMAND(ID_FLC_GENERATE_HEXFILES, OnFlcGenerateHexfiles)
  ON_COMMAND(ID_FLC_GENERATE_MEMFILES, OnFlcGenerateMemfiles)
  ON_COMMAND(ID_VIEW_TEXT_LARGE, OnViewTextLarge)
  ON_COMMAND(ID_VIEW_TEXT_LARGEST, OnViewTextLargest)
  ON_COMMAND(ID_VIEW_TEXT_MEDIUM, OnViewTextMedium)
  ON_COMMAND(ID_VIEW_TEXT_SMALL, OnViewTextSmall)
  ON_COMMAND(ID_VIEW_TEXT_SMALLEST, OnViewTextSmallest)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_LARGEST, OnUpdateViewTextLargest)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_LARGE, OnUpdateViewTextLarge)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_MEDIUM, OnUpdateViewTextMedium)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_SMALL, OnUpdateViewTextSmall)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_SMALLEST, OnUpdateViewTextSmallest)
  //}}AFX_MSG_MAP
  // Standard printing commands
  ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_DIRECT, CScrollView::OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_PREVIEW, CScrollView::OnFilePrintPreview)
D_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
 CFIDEView construction/destruction

IDEView::CFIDEView()

  // initialize the fonts
  m_logFont.lfHeight = nMEDIUMFONT*10;
  m_logFont.lfWidth = 0;
  m_logFont.lfEscapement = 0;
  m_logFont.lfOrientation = 0;
  m_logFont.lfWeight = FW_NORMAL;
  m_logFont.lfItalic = 0;
  m_logFont.lfUnderline = 0;
  m_logFont.lfStrikeOut = 0;
  m_logFont.lfCharSet = ANSI_CHARSET;
  m_logFont.lfOutPrecision = OUT_DEFAULT_PRECIS;
  m_logFont.lfClipPrecision = CLIP_DEFAULT_PRECIS;
  m_logFont.lfQuality = PROOF_QUALITY;
  m_logFont.lfPitchAndFamily = VARIABLE_PITCH | FF_ROMAN;
  strcpy(m_logFont.lfFaceName, "Times New Roman");


IDEView::~CFIDEView()



)L CFIDEView::PreCreateWindow(CREATESTRUCT& cs)

  // TODO: Modify the Window class or styles here by modifying
  //  the CREATESTRUCT cs
```

```
    CSize sizeTotal;
    // TODO: calculate the total size of this view
    sizeTotal.cx = sizeTotal.cy = 100;
    SetScrollSizes(MM_TEXT, sizeTotal);
}

////////////////////////////////////////////////////////////////////////////
// CFIDEView printing

BOOL CFIDEView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
}

void CFIDEView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

void CFIDEView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}

////////////////////////////////////////////////////////////////////////////
// CFIDEView diagnostics

#ifdef _DEBUG
void CFIDEView::AssertValid() const
{
    CScrollView::AssertValid();
}

void CFIDEView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}

CFIDEDoc* CFIDEView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CFIDEDoc)));
    return (CFIDEDoc*)m_pDocument;
}
#endif //_DEBUG

////////////////////////////////////////////////////////////////////////////
// CFIDEView internal implementation for Drawing

void CFIDEView::ShowFuzzy(CDC *pDC, CFIDEDoc *pDoc)
{
    // initialize the position of text in the window
    UINT pos = 0;
```

```
pos += m_logFont.lfHeight/nSPACESCALER;

// ---------------------------- Outputs
str = "OUTPUTS";
pDC->TextOut(10, pos, str);
pos += m_logFont.lfHeight/nSPACESCALER;
count = pDoc->m_xaOutput.GetSize();
for (i = 0; i < count; i++)
{
    (pDoc->m_xaOutput.ElementAt(i))->Format(str);
    pDC->TextOut(10, pos, str);
    pos += m_logFont.lfHeight/nSPACESCALER;
}
pos += m_logFont.lfHeight/nSPACESCALER;

// ---------------------------- Terms
str = "TERMS";
pDC->TextOut(10, pos, str);
pos += m_logFont.lfHeight/nSPACESCALER;
count = pDoc->m_xaTerm.GetSize();
for (i = 0; i < count; i++)
{
    (pDoc->m_xaTerm.ElementAt(i))->Format(str);
    pDC->TextOut(10, pos, str);
    pos += m_logFont.lfHeight/nSPACESCALER;
}
pos += m_logFont.lfHeight/nSPACESCALER;

// ---------------------------- Rules
str = "RULES";
pDC->TextOut(10, pos, str);
pos += m_logFont.lfHeight/nSPACESCALER;
count = pDoc->m_xaRule.GetSize();
for (i = 0; i < count; i++)
{
    (pDoc->m_xaRule.ElementAt(i))->Format(str);
    pDC->TextOut(10, pos, str);
    pos += m_logFont.lfHeight/nSPACESCALER;
}

// make the window scrolls
// -------------------- calculate the document size
CSize docSize(100, pos);

// -------------------- calculate the page size
CRect rect;
GetClientRect(&rect);
CSize pageSize(rect.right, rect.bottom);

// -------------------- calculate the line size
CSize lineSize(0, m_logFont.lfHeight/nSPACESCALER);

// -------------------- adjust the scrollers
SetScrollSizes(MM_TEXT, docSize, pageSize, lineSize);
```

```
dlg.m_RleDlg.m_saInput.SetSize(count, 1);
for (int i = 0; i < count; i++)
{
    CDtI0* pDtIn = new CDtI0();
    *pDtIn = *(pDoc->m_xaInput[i]);
    dlg.m_InpDlg.m_xaInput[i] = pDtIn;
    dlg.m_TrmDlg.m_saInput[i] = pDtIn->GetName();
    dlg.m_RleDlg.m_saInput[i] = pDtIn->GetName();
}
dlg.m_InpDlg.m_strName = pDoc->m_xaInput.ElementAt(--i)->GetName();
dlg.m_InpDlg.m_nInit = pDoc->m_xaInput.ElementAt(i)->GetInit();

// load outputs
count = pDoc->m_xa0utput.GetSize();
dlg.m_0utDlg.m_xa0utput.SetSize(count, 1);
dlg.m_TrmDlg.m_sa0utput.SetSize(count, 1);
dlg.m_RleDlg.m_sa0utput.SetSize(count, 1);
for (i = 0; i < count; i++)
{
    CDtI0* pDt0ut = new CDtI0();
    *pDt0ut = *(pDoc->m_xa0utput[i]);
    dlg.m_0utDlg.m_xa0utput[i] = pDt0ut;
    dlg.m_TrmDlg.m_sa0utput[i] = pDt0ut->GetName();
    dlg.m_RleDlg.m_sa0utput[i] = pDt0ut->GetName();
}
dlg.m_0utDlg.m_strName = pDoc->m_xa0utput.ElementAt(--i)->GetName();
dlg.m_0utDlg.m_nInit = pDoc->m_xa0utput.ElementAt(i)->GetInit();

// load terms
count = pDoc->m_xaTerm.GetSize();
dlg.m_TrmDlg.m_xaTerm.SetSize(count, 1);
dlg.m_RleDlg.m_saTerm.SetSize(count, 1); // used for m_Term in rule dlg
for (i = 0; i < count; i++)
{
    CDtTrm* pDtTrm = new CDtTrm();
    *pDtTrm = *(pDoc->m_xaTerm[i]);
    dlg.m_TrmDlg.m_xaTerm[i] = pDtTrm;
    dlg.m_RleDlg.m_saTerm[i] = pDtTrm->GetName(); // used for m_Term in

}
dlg.m_TrmDlg.m_strName   = pDoc->m_xaTerm.ElementAt(--i)->GetName();
dlg.m_TrmDlg.m_strCenter = pDoc->m_xaTerm.ElementAt(i)->GetCenter();
dlg.m_TrmDlg.m_strWidth  = pDoc->m_xaTerm.ElementAt(i)->GetWidth();

// load rules
count = pDoc->m_xaRule.GetSize();
dlg.m_RleDlg.m_xaRule.SetSize(count, 1);
for (i = 0; i < count; i++)
{
    CDtRl* pDtRl = new CDtRl();
    *pDtRl = *(pDoc->m_xaRule[i]);
    dlg.m_RleDlg.m_xaRule[i] = pDtRl;
}
dlg.m_RleDlg.m_xTemp = *(pDoc->m_xaRule.ElementAt(--i));
```

```
  for (i = 0; i < count; i++)
      pDoc->m_xaInput[i] = dlg.m_InpDlg.m_xaInput[i];
  dlg.m_InpDlg.m_xaInput.RemoveAll();

  // save outputs
  // ---------- first, delete old data
  count = pDoc->m_xaOutput.GetSize();
  for (i = 0; i < count; i++)
      delete pDoc->m_xaOutput[i];
  // ---------- then point to new data
  count = dlg.m_OutDlg.m_xaOutput.GetSize();
  pDoc->m_xaOutput.SetSize(count, 1);
  for (i = 0; i < count; i++)
      pDoc->m_xaOutput[i] = dlg.m_OutDlg.m_xaOutput[i];
  dlg.m_OutDlg.m_xaOutput.RemoveAll();

  // save terms
  // ---------- first, delete old data
  count = pDoc->m_xaTerm.GetSize();
  for (i = 0; i < count; i++)
      delete pDoc->m_xaTerm[i];
  // ---------- then point to new data
  count = dlg.m_TrmDlg.m_xaTerm.GetSize();
  pDoc->m_xaTerm.SetSize(count, 1);
  for (i = 0; i < count; i++)
      pDoc->m_xaTerm[i] = dlg.m_TrmDlg.m_xaTerm[i];
  dlg.m_TrmDlg.m_xaTerm.RemoveAll();

  // save rules
  // ---------- first, delete old data
  count = pDoc->m_xaRule.GetSize();
  for (i = 0; i < count; i++)
      delete pDoc->m_xaRule[i];
  // ---------- then point to new data
  count = dlg.m_RleDlg.m_xaRule.GetSize();
  pDoc->m_xaRule.SetSize(count, 1);
  for (i = 0; i < count; i++)
      pDoc->m_xaRule[i] = dlg.m_RleDlg.m_xaRule[i];
  dlg.m_RleDlg.m_xaRule.RemoveAll();

  // mark document as unclean
  pDoc->SetModifiedFlag();


 Clean unneeded dynamic data created by cancelled Parameter Dialog
id CFIDEView::CleanTemp(CFIDEParDlg& dlg)

  // clean temporary inputs
  for (int i = 0; i < dlg.m_InpDlg.m_xaInput.GetSize(); i++)
      delete dlg.m_InpDlg.m_xaInput[i];
  dlg.m_InpDlg.m_xaInput.RemoveAll();

  // clean temporary outputs
  for (i = 0; i < dlg.m_OutDlg.m_xaOutput.GetSize(); i++)
```

```
id CFIDEView::OnViewTextLargest()

  m_logFont.lfHeight = nLARGESTFONT*10;
  Invalidate();


id CFIDEView::OnUpdateViewTextLargest(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_logFont.lfHeight == nLARGESTFONT*10)? 1:0);


id CFIDEView::OnViewTextLarge()

  m_logFont.lfHeight = nLARGEFONT*10;
  Invalidate();


id CFIDEView::OnUpdateViewTextLarge(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_logFont.lfHeight == nLARGEFONT*10)? 1:0);


id CFIDEView::OnViewTextMedium()

  m_logFont.lfHeight = nMEDIUMFONT*10;
  Invalidate();


id CFIDEView::OnUpdateViewTextMedium(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_logFont.lfHeight == nMEDIUMFONT*10)? 1:0);


id CFIDEView::OnViewTextSmall()

  m_logFont.lfHeight = nSMALLFONT*10;
  Invalidate();


id CFIDEView::OnUpdateViewTextSmall(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_logFont.lfHeight == nSMALLFONT*10)? 1:0);


id CFIDEView::OnViewTextSmallest()

  m_logFont.lfHeight = nSMALLESTFONT*10;
  Invalidate();


id CFIDEView::OnUpdateViewTextSmallest(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_logFont.lfHeight == nSMALLESTFONT*10)? 1:0);
```

```
    else
        CleanTemp(ParDlg);

    Invalidate(); // force redraw


void CFIDEView::OnFlcGenerateHexfiles()

    char szDefExt[] = "hex";
    char szFileName[] = "PARAM";
    char szFilter[] =
        "Intel HEX Files (*.hex)|*.hex|All Files (*.*)|*.*||";
    CFileDialog FileDlg(FALSE, szDefExt, szFileName,
        OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT, szFilter);
    int result = FileDlg.DoModal();
    if (result == IDOK)
    {
        CFile f(FileDlg.GetFileName(), CFile::modeCreate | CFile::modeWrite
        CArchive ar(&f, CArchive::store);
        CFIDEDoc* pDoc = GetDocument();
        if (pDoc->GenerateHEXFile(ar))
            AfxMessageBox(IDS_HEX_FILE_GENERATED);
        else
            AfxMessageBox(IDS_HEX_FILE_CANT_GENERATED);
    }
    else
    {
        AfxMessageBox(IDS_HEX_FILE_NOT_GENERATED);
    }


void CFIDEView::OnFlcGenerateMemfiles()

    char szDefExt[] = "mem";
    char szFilter[] =
        "Foundation MEM Files (*.mem)|*.mem|All Files (*.*)|*.*||";
    for (int i = 0; i < 4; i++)
    {
        CString temp;
        temp.Format("PAR%i", i+1);
        if (i == 3)
            temp = _T("RULE");
        CFileDialog FileDlg(FALSE, szDefExt, (LPCTSTR) temp,
            OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT, szFilter);
        int result = FileDlg.DoModal();
        if (result == IDOK)
        {
            CFile f(FileDlg.GetFileName(), CFile::modeCreate | CFile::modeW
            CArchive ar(&f, CArchive::store);
            CFIDEDoc* pDoc = GetDocument();
            if (pDoc->GenerateMEMFile(i+1, ar))
                AfxMessageBox(IDS_MEM_FILE_GENERATED);
            else
```

```
rivate:
   CStringList m_LineList;

/ Operations
ublic:
   CStringList* GetLineList() { return &m_LineList; };

/ Overrides
   // ClassWizard generated virtual function overrides
   //{{AFX_VIRTUAL(CFileDoc)
   public:
   virtual void Serialize(CArchive& ar);   // overridden for document i/o
   virtual BOOL OnOpenDocument(LPCTSTR lpszPathName);
   protected:
   virtual BOOL OnNewDocument();
   //}}AFX_VIRTUAL

/ Implementation
ublic:
   virtual ~CFileDoc();
ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
endif

   // Generated message map functions
rotected:
   //{{AFX_MSG(CFileDoc)
       // NOTE - the ClassWizard will add and remove member functions here
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
}

'/////////////////////////////////////////////////////////////////////////////
' CFileView view

lass CFileView : public CScrollView

rotected:
   CFileView();           // protected constructor used by dynamic creation
   DECLARE_DYNCREATE(CFileView)

rivate:
   CSize m_ViewCharSize; // Dimensions of screen character in device units
   CSize m_DocSize;      // Document size in device units
   CFont* m_pFont;       // Current font
   int m_nFontPoint;     // font size

  Operations
otected:
   void ComputeViewMetrics();
   void ComputeVisibleLines(CDC* pDC, int& nFirst, int& nLast);
```

```
rotected:
   virtual ~CFileView();
ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
endif

   // Generated message map functions
   //{{AFX_MSG(CFileView)
   afx_msg void OnViewTextLargest();
   afx_msg void OnUpdateViewTextLargest(CCmdUI* pCmdUI);
   afx_msg void OnViewTextLarge();
   afx_msg void OnUpdateViewTextLarge(CCmdUI* pCmdUI);
   afx_msg void OnViewTextMedium();
   afx_msg void OnUpdateViewTextMedium(CCmdUI* pCmdUI);
   afx_msg void OnViewTextSmall();
   afx_msg void OnUpdateViewTextSmall(CCmdUI* pCmdUI);
   afx_msg void OnViewTextSmallest();
   afx_msg void OnUpdateViewTextSmallest(CCmdUI* pCmdUI);
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
;

ifndef _DEBUG  // debug version in TextView.cpp
nline CFileDoc* CFileView::GetDocument()
  { return (CFileDoc*)m_pDocument; }
endif

////////////////////////////////////////////////////////////////////////////
/{{AFX_INSERT_LOCATION}}
/ Microsoft Visual C++ will insert additional declarations immediately be
previous line.

endif // !defined(AFX_FIDEFILE_H__8456B4C1_3A3C_11D3_BD00_00E04CAACD92__IN
```

```
FileDoc::CFileDoc()


00L CFileDoc::OnNewDocument()

    // there will be no new document, only for read-only
    return FALSE;


FileDoc::~CFileDoc()



EGIN_MESSAGE_MAP(CFileDoc, CDocument)
    //{{AFX_MSG_MAP(CFileDoc)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
ND_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
/ CFileDoc diagnostics

ifdef _DEBUG
oid CFileDoc::AssertValid() const

    CDocument::AssertValid();


oid CFileDoc::Dump(CDumpContext& dc) const

    CDocument::Dump(dc);

endif //_DEBUG

////////////////////////////////////////////////////////////////////////////
/ CFileDoc serialization

oid CFileDoc::Serialize(CArchive& ar)

    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }


////////////////////////////////////////////////////////////////////////////
/ CFileDoc commands
```

```
  {
      //remove the noise characters at the end of the line
      int nLastCharIndex = strLine.GetLength()-1;
      while (nLastCharIndex >= 0 && strLine[nLastCharIndex] < ' ')
      {
          strLine.SetAt(nLastCharIndex--, '\0');
      }

      //  Add to CStringList
      m_LineList.AddTail(strLine);
  }

  EndWaitCursor();
  return TRUE;


////////////////////////////////////////////////////////////////////////////
 CFileView

PLEMENT_DYNCREATE(CFileView, CScrollView)

ileView::CFileView() : m_ViewCharSize(0,0), m_DocSize(0,0)

  m_pFont      = NULL;
  m_nFontPoint = nMEDIUMFONT*10;


ileView::~CFileView()



GIN_MESSAGE_MAP(CFileView, CScrollView)
  //{{AFX_MSG_MAP(CFileView)
  ON_COMMAND(ID_VIEW_TEXT_LARGEST, OnViewTextLargest)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_LARGEST, OnUpdateViewTextLargest)
  ON_COMMAND(ID_VIEW_TEXT_LARGE, OnViewTextLarge)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_LARGE, OnUpdateViewTextLarge)
  ON_COMMAND(ID_VIEW_TEXT_MEDIUM, OnViewTextMedium)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_MEDIUM, OnUpdateViewTextMedium)
  ON_COMMAND(ID_VIEW_TEXT_SMALL, OnViewTextSmall)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_SMALL, OnUpdateViewTextSmall)
  ON_COMMAND(ID_VIEW_TEXT_SMALLEST, OnViewTextSmallest)
  ON_UPDATE_COMMAND_UI(ID_VIEW_TEXT_SMALLEST, OnUpdateViewTextSmallest)
  //}}AFX_MSG_MAP
D_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
 CFileView drawing

id CFileView::OnDraw(CDC* pDC)

  int nFirstLn, nLastLn;
```

```
  //  and output them to the display context
  CStringList *pLineList = GetDocument()->GetLineList();

  CString strLine;
  POSITION pos;
  while (nFirstLn <= nLastLn)
  {
      if(( pos = pLineList->FindIndex(nFirstLn)) != NULL)
      {
          strLine = pLineList->GetAt(pos);
          pDC->TabbedTextOut(nXPos, nYPos, strLine, 0, NULL, 0);
          nYPos -= CharSize.cy;
          nFirstLn++;
      }
  }

  //  Cleanup and restore original GDI Objects
  if(pPreviousFont)
  {
      pDC->SelectObject(pPreviousFont);
  }


/////////////////////////////////////////////////////////////////////////////
 CFileView diagnostics

fdef _DEBUG
id CFileView::AssertValid() const

  CScrollView::AssertValid();


id CFileView::Dump(CDumpContext& dc) const

  CScrollView::Dump(dc);


ileDoc* CFileView::GetDocument() // non-debug version is inline

  ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CFileDoc)));
  return (CFileDoc*)m_pDocument;

ndif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
 CFileView internal functions

ont* CFileView::GetFont()

  m_pFont = new CFont;
  if(m_pFont)
      m_pFont->CreatePointFont(m_nFontPoint, "Arial");
  return m_pFont;
```

```
  m_ViewCharSize.cx = tm.tmAveCharWidth;

  // convert to device units to minimize round off error
  pDC->LPtoDP(&m_ViewCharSize);

  //  Calculate document size
  CFileDoc* pDoc = GetDocument();
  m_DocSize.cx = 0;
  m_DocSize.cy = m_ViewCharSize.cy *
                  pDoc->GetLineList()->GetCount();

  // loop through the document and find the longest line
  CString Line;
  CSize size;
  POSITION pos = pDoc->GetLineList()->GetHeadPosition();
  while( pos != NULL )
  {
      Line = pDoc->GetLineList()->GetNext( pos );
      size = pDC->GetTextExtent(Line, Line.GetLength());
      m_DocSize.cx = max(size.cx, m_DocSize.cx);
  }

  //  Account for our simple margin
  m_DocSize.cx += 4 * m_ViewCharSize.cx;

  // clean up
  if(pPreviousFont)
  {
      pDC->SelectObject(pPreviousFont);
  }
  pDC->RestoreDC(nSaveDC);
  ::ReleaseDC(NULL,pDC->GetSafeHdc());


id CFileView::ComputeVisibleLines(CDC* pDC, int& nFirst, int& nLast)

  int nLineCount = GetDocument()->GetLineList()->GetCount();

  // Get the viewport origin, convert to logical coordinates
  CPoint pt = pDC->GetViewportOrg();
  pDC->DPtoLP(&pt,1);

  // Get the clipping region, in logical coordinates
  CRect rc;
  pDC->GetClipBox(&rc);

  // Get the logical line height
  CSize CharSize = GetCharSize();

  // Compute the first visible line
  // The algorithm for the first visible line is
  //   · Calculate the distance from the top of the viewport to the top of
 region
  //   · Divide this distance by the height of a line, giving the number o
```

```
CFileView message handlers

id CFileView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)

  ComputeViewMetrics();

  // --- calculate the page size
  CRect rect;
  GetClientRect(&rect);
  CSize pageSize(rect.right, rect.bottom);

  // --- calculate the line size
  CSize lineSize(0, m_ViewCharSize.cy);

  SetScrollSizes(MM_LOENGLISH, GetDocSize(), pageSize, lineSize);
  Invalidate();


id CFileView::OnViewTextLargest()

  m_nFontPoint = nLARGESTFONT*10;
  ComputeViewMetrics();
  Invalidate();


id CFileView::OnUpdateViewTextLargest(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_nFontPoint == nLARGESTFONT*10)? 1:0);


id CFileView::OnViewTextLarge()

  m_nFontPoint = nLARGEFONT*10;
  ComputeViewMetrics();
  Invalidate();


id CFileView::OnUpdateViewTextLarge(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_nFontPoint == nLARGEFONT*10)? 1:0);


id CFileView::OnViewTextMedium()

  m_nFontPoint = nMEDIUMFONT*10;
  ComputeViewMetrics();
  Invalidate();


id CFileView::OnUpdateViewTextMedium(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_nFontPoint == nMEDIUMFONT*10)? 1:0);
```

```
id CFileView::OnUpdateViewTextSmallest(CCmdUI* pCmdUI)

  pCmdUI->SetCheck((m_nFontPoint == nSMALLESTFONT*10)? 1:0);
```

Alamat : Jl. Ngagel Madya Utara 5B

Surabaya, 60284

Penulis dilahirkan pada tanggal 30 Juli 1976 di Malang, Jawa Timur sebagai putra pertama dari tiga bersaudara dari Ayah Soegiarto Prasetyo dan Ibu Marliana Soerachman Cholil.

Riwayat Pendidikan :

- SD Katolik Santa Maria I Malang : Lulus Tahun 1988
- SMP Katolik Santa Maria I Malang : Lulus Tahun 1991
- SMA Katolik Santo Albertus Malang : Lulus Tahun 1994
- Diterima di Jurusan Teknik Elektro FTI-ITS tahun 1994 melalui jalur UMPTN dan masuk bidang studi Elektronika sampai sekarang.

Kegiatan Kemahasiswaan :

- Asisten Praktikum Rangkaian Listrik.
- Asisten Praktikum Elektronika.
- Asisten Praktikum Elektronika Lanjut II.