Jurnal Teknovasi
Volume 04, Nomor 01, 2017, 10-27
ISSN : 2355-701X

PERANCANGAN MIKROPROSESOR 8 BIT DENGAN MENGGUNAKAN BAHASA VHDL PADA FPGA XILINX SPARTAN 3

Friendly[1]*
[1]Program Studi Teknik Komputer dan Informatika
Politeknik Negeri Medan Medan Indonesia
Telp: 081370203112
*Email : friendly@polmed.ac.id

ABSTRAK

Kebutuhan perancangan perangkat mikroprosesor yang saat ini didominasi oleh negara maju disebabkan biaya produksi yang tinggi. Untuk mengatasi hal tersebut, beberapa penyedia peralatan semikonduktor mengembangkan FPGA. FPGA (Field Programmable Gate Array) merupakan salah satu perangkat digital yang dapat diprogram ulang dengan jumlah gerbang antara 50.000 sampai dengan 5.000.000. Pemanfaatan FPGA untuk membuat dan merancang peralatan-peralatan untuk aplikasi yang rumit dimudahkan dengan banyaknya bahasa yang dapat dipakai untuk memrogramnya. VHDL adalah salah satu dari banyak bahasa yang ditawarkan untuk mengkonfigurasikan sebuah FPGA. Pemanfaatan FPGA dalam merancangan mikroprosesor dapat membantu mengurangkan biaya yang diperlukan untuk membuat perancangan awal. Dalam tulisan ini dipaparkan desain dan perancangan mikroprosesor 8 bit yang akan dapat bekerja seperti halnya mikroprosesor 8 bit lainnya.

Kata Kunci : ***FPGA, mikroprosesor, vhdl***

PENDAHULUAN

Mikroprosesor merupakan salah satu perangkat elektronika yang berkembang dengan sangat pesat sehubungan dengan meningkatnya kebutuhan untuk pengolahan data-data berkapasitas besar dan cepat. Hal ini membutuhkan pengembangan mikroprosesor secara cepat yang juga harus disesuaikan dengan kebutuhan. Mikroprosesor dengan spesifikasi khusus telah banyak dipergunakan secara luas pada perangkat-perangkat portable saat ini seperti: mp3 player portable, video player portable, dan pada peralatan yang membutuhkan pemrosesan sinyal digital yang kompleks. Untuk membuat suatu mikroprosesor dengan tujuan khusus diperlukan biaya yang cukup mahal dan peralatan yang cukup banyak serta diperlukan keahlian dalam membuat rancangan VLSI yang membentuk gerbang-gerbang logika, membuat rancangan logika untuk membentuk suatu blok, dan menyusun blok-blok tersebut berdasarkan diagram keadaan agar setiap instruksi yang diberikan pada memori. Program dapat dijemput, diterjemahkan dan dijalankan dengan baik sesuai fungsi dari mikroprosesor tersebut. Dari beberapa hal diatas tampak bahwa tidak memungkinkan untuk dapat merancang suatu mikroprosesor dengan kemampuan khusus secara cepat dan tepat. Setelah mikroprosesor tersebut selesai dan bila terjadi kesalahan sehingga mikroprosesor tersebut tidak dapat berjalan sesuai dengan fungsinya, maka perancangan mikroprosesor tersebut harus diulang dari awal dan dengan penelusuran yang amat panjang dan rumit.

FPGA (*Field Programmable Gate Array*) merupakan salah satu perangkat digital yang dapat diprogram ulang dengan jumlah gerbang antara 50.000 sampai dengan 5.000.000. Dengan

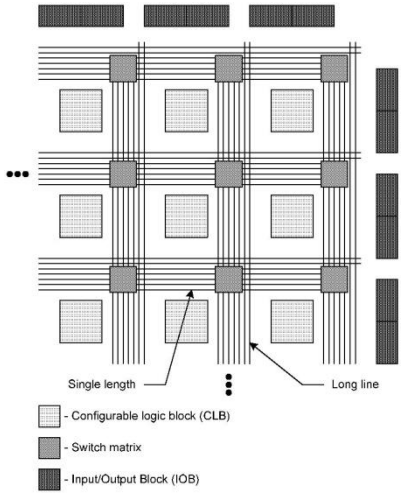
Jurnal Teknovasi
Volume 04, Nomor 01, 2017, 10-27
ISSN : 2355-701X

memanfaatkan FPGA ini, perancangan mikroprosesor dapat dilakukan dalam waktu yang singkat dan cepat. Konfigurasi pada FPGA menggunakan bahasa tingkat tinggi VHDL (*VHSIC Hardware Description Langiage*) yang memiliki kemiripan dengan bahasa tingkat tinggi lainnya. Selain dengan VHDL, konfigurasi pada FPGA juga dapat menggunakan bahasa Verilog, diagram skematik dan menggunakan diagram pewaktuan. Dalam perancangan ini akan dibuat sebuah mikroprosesor sederhana dengan memanfaatkan FPGA.

METODE PENELITIAN

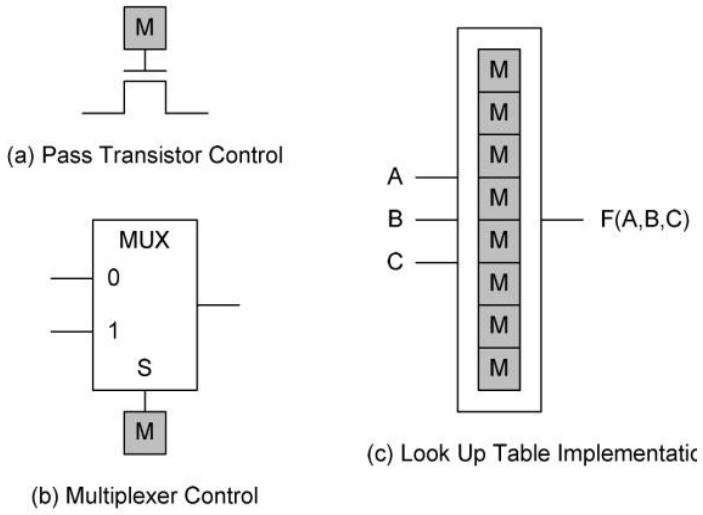
FPGA (*Field-Programmable Gate Array*)

FPGA memiliki struktur yang menyerupai suatu susunan gerbang ASIC (*Application-Specific Integrated Circuit*). FPGA terdiri dari tiga komponen utama, yaitu *Configurable Logic Block* (CLB), *Switch Matrix*, dan *Input/Output Block* (IOB) (Gambar 1).



Gambar 1. Tiga Komponen Utama FPGA

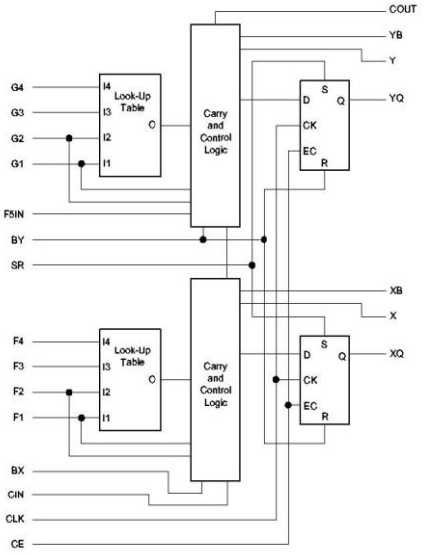
Informasi program untuk mengontrol elemen logika yang dapat dikonfigurasi dan interkoneksi antara sumber-sumber, disimpan menggunakan teknologi SRAM (*Static Random Access Memory*). Implementasi logika dari pengontrolan oleh bit-bit SRAM menggunakan tiga teknik, yaitu *Pass Transistor Control* (Gambar 2.a), *Multiplexer Control* (Gambar 2.b), dan *LookUp Table* (LUT) (Gambar 2.c).



Gambar 2. Implementasi Pengontrolan Logika pada FPGA

CLB merupakan *array* dari blok-blok atau elemen untuk mengkonfigurasi atau membangun logika. Blok-blok dasar yang membangun sebuah CLB disebut *logic cell* (LC). Setiap LC mengandung *4-input function generator , carry logic*, dan *storage element*. Setiap CLB terdiri dari empat buah LC yang dikelompokan dalam dua *slice* yang serupa (Gambar 3).

Bagian 4-*input function generator* diimplementasikan sebagai 4-*input look-up table* (LUT). Setiap LUT dapat menyediakan 16x1-*bit synchronous RAM*. Sehingga dua LUT dalam sebuah *slice* dapat disatukan untuk membentuk 16x2-*bit* atau 32x1-*bit synchronous RAM* atau 16x1-*bit dual-port synchronous RAM*. LUT juga dapat menyediakan 16-*bit shift register* yang cocok untuk menangkap data kecepatan tinggi atau *burst-mode* (Bhasker, 1999).



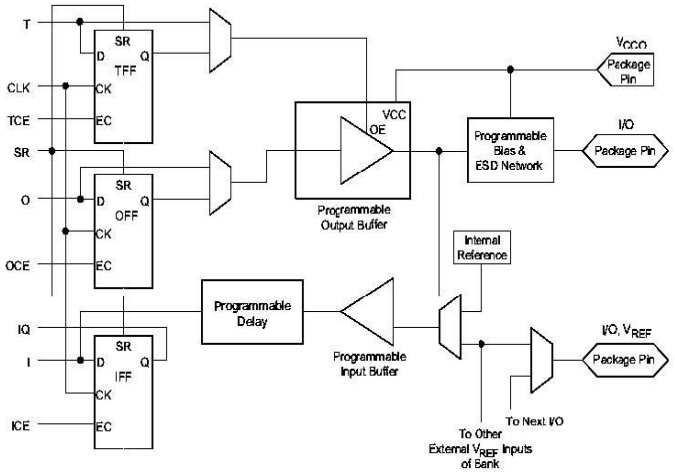
Gambar 3. Skematik Sebuah *Slice*

Storage element pada *slice* dapat dikonfigurasi sebagai *edge-triggered D-type flip-flop* atau sebagai *level-sensitive latch*. Masukkan pada D dapat melalui *function generator* di dalam *slice*

maupun langsung dari masukan *slice* (tidak melalui *function generator*) (Bhasker, 1999).

Switch Matrix menghubungkan ke dan dari CLB dan IOB yang dapat diprogram, yaitu dengan *Pass Transistor Control*. *Long-line* merupakan jenis persambungan jarak jauh, sedangkan *Single-length* menghubungkan antara CLB atau IOB yang bersebelahan.

IOB menyediakan hubungan antara *pin* dari paket(I/O *pin*) dengan logika internal. Rangkaian IOB ditunjukkan pada gambar 4.



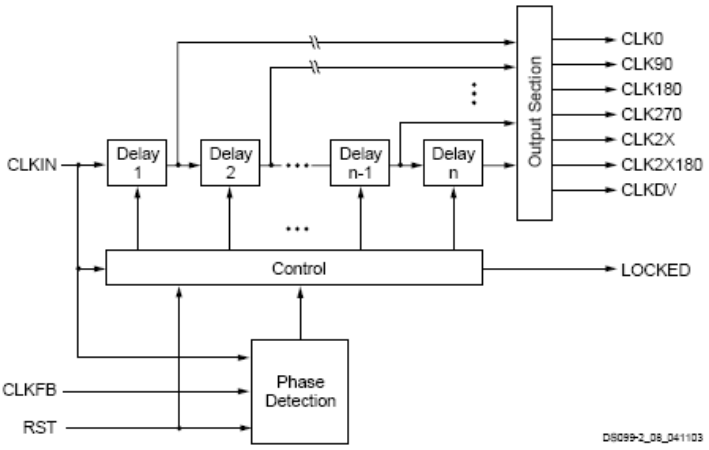
Gambar 4. Skematik IOB

Keistimewaannya, *input* dan *output* dapat mendukung banyak standar sinyalisasi. Standar sinyal untuk *low-voltage* bergantung pada VREF sedangkan untuk *high-voltage* bergantung pada VCCO (Bhasker, 1999).

Rangkaian *weak-keeper* terhubung dengan setiap *output*. Tegangan pada VREF harus menyediakan standar sinyal yang diinginkan agar rangkaian ini dapat memeriksa tegangan pada *pad* dan mengedalikan *pin High* atau *Low* sesuai dengan sinyal *input* yang didapat melalui IOB *input buffer* (Bhasker, 1999).

IOB terdiri dari *register* dan *three-state buffer*. *Register* pada IOB berfungsi sebagai *edge-triggered D-type flip-flop* atau sebagai *level-sensitive latch* (Gambar 3). *Three-state buffer* memungkinkan I/O *pin* digunakan sebagai masukan, keluaran, atau masukan/keluaran (Bhasker, 1999).

Rangkaian *Delay-locked Loop* (DLL) pada Gambar 5 memungkinkan *zero propagation delay*, *low clock skew* dari sinyal *clock* keluaran yang disebarkan kesetiap *device*, dan *clock domain control* yang sangat baik (Bhasker, 1999).



Gambar 5. Skematik DDL

VHDL (VHSIC *Hardware Decription Language*)

VHDL adalah singkatan dari VHSIC *Hardware Description Language*. Sedangkan VHSIC adalah singkatan dari *Very High Speed Integrated Circuit*. VHDL merupakan standar yang dikembangkan oleh IEEE (*Institute of Electrical and Electronics Engineers*). Standaryang digunakan secara luas adalah VHDL 1076-1987. Sedangkan versi revisinya, VHDL 1076-1993 masih dalam proses untuk menggantikan versi yang lama.

VHDL dapat digunakan sebagai dokumentasi, pembuktian, dan sintesa pada perancangan digital berukuran besar. VHDL menggunakan tiga pendekatan untuk mendiskripsikan *hardware*. Ketiga pendekatan itu adalah metode *structural*, *data flow*, dan *behavioral*. [7]

Metode *structural* membagi rancangan kedalam beberapa blok agar mudah dimengerti dan diatur. Blok-blok tersebut kemudian dihubungkan hingga membentuk rancangan yang utuh. Setiap blok pada VHDL dapat disamakan dengan sebuah bagian yang berdiri sendiri yang disebut *entity*. *Entity* juga menggambarkan antarmuka rancangan. *Component* menggambarkan antarmuka dari *entity* yang nantinya akan digunakan sebagai sebuah *instance* (sub blok). *Component instance* adalah salinan lain dari sebuah *component* yang akan dihubungkan kebagian (*part*) dan sinyal lain.

Pada metode *data flow*, jalur digambarkan dengan menyatakan bagaimana *input* dan *output* dalam komponen primitip (seperti gerbang AND) terhubung. Bagian arsitektur mengambarkan operasi internal dari sebuah rancangan dan metode ini menentukan bagaimana aliran data dari *input* hingga *output*.

Pendekatan dengan metode *behavioral* berbeda dengan dua metode sebelumnya, ia tidak benar-benar menggambarkan bagaimana rancangan diimplementasikan. Dasarnya adalah pendekatan kotak hitam (*blackbox*) dalam melakukan pemodelan, tidak perduli apa isi kotak hitam tersebut dan bagaimana cara kerjanya. Penjabaran behavioral didukung oleh *process statement* yang muncul dalam badan *architecture declaration* seperti pada saat menyatakan *signal assignment*. Isi dari *process statement* dapat diurutkan penulisannya seperti pada *sequential statement* yang ditemukan dalam bahasa pemrograman.

Gambar pada skematik dapat langsung menyampaikan struktur rancangan, tetapi karena formatnya yang spesifik menyebabkan skematik tidak *portable*. VHDL lebih *portable* dan mudah dimodifikasi. Banyak *development software* yang hanya mendukung representasi textual dari sebuah rancangan(seperti VHDL, Verilog, Abel, dan HDL yang lain). Ada beberapa tool yang memungkinkan perubahan format dari representasi textual yang satu ke yang lainnya.

Jurnal Teknovasi
Volume 04, Nomor 01, 2017, 10-27
ISSN : 2355-701X

Mikroprosesor

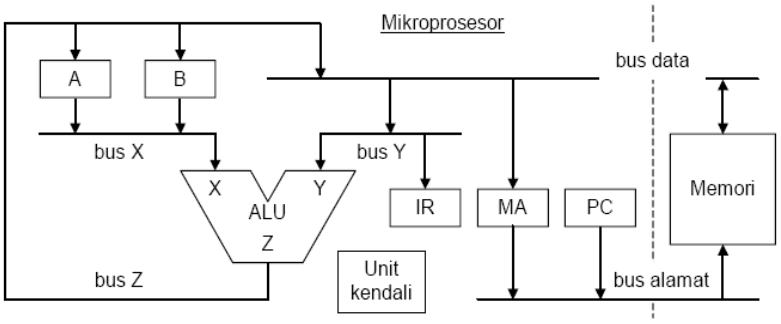
Mikroprosesor merupakan suatu komponen elektronik yang terusun atas transistor-transistor yang sangat kecil pada sebuah rangkaian semikonduktor terintegrasi (IC). Mikroprosesor umumnya dipakai sebagai pengendali utama dari uatu sistem elektronik. Beberapa bagian utama yang terdapat pada sebuah mikroprosesor adalah, control unit, arithmetic logic unit, memory, dan input/output interface.

Control Unit memiliki fungsi untuk mengatur kerja dari bagian lain pada mikroprosesor bila suatu instruksi diberikan. Berdasarkan instruksi yang diberikan, control unit akan mengeluarkan bit bit kendali yang terhubung ke arithmetic logic , buffer, serta mengatur aliran data dari dan ke memory dan register. Control unit terdiri dari : program counter, instruction register, instruction decode dan branch decision . Program counter merupakan suatu register yang menunjukkan alamat instruksi pada program memory pada control unit. Jika suatu instruksi diambil dari program memory, maka nilai dari program counter akan dinaikkan. Data alamat dari program coutner ini berbeda-beda pada setiap mikroprosesor. Data alamat dapat berupa alamat dari instruksi yang sedang dijalankan atau alamat dari instruksi yang akan dijalankan. Maka instruksi umumnya diperoleh secara sequential dari memory. Beberapa instruksi seperti pencabangan dan prosedur dapat dilakukan dengan memasukkan nilai baru ke dalam program counter. Instruction register merupakan suatu register yang menyimpan instruksi yang telah diambil dari memory sebelum instruksi tersebut dijalankan. Instruction decoder terdiri dari gerbang-gerbang logika yang disusun untuk mengartikan kode intruksi yang diambil dari program memory untuk mengaktifkan dan mematikan fungsi-fungsi blok lainnya pada mikroprosesor. Blok tersebut termasuk blok pada control unit. Branch decision merupakan suatu blok yang terdiri dari gerbang-gerbang logika yang diaktifkan bila instruction decoder menerima intruksi untuk mengakses instruksi pada alamat tertentu pada memory program.

Arithmetic logic unit berfungsi untuk melakukan operasi-operasi logika dan aritmatik pada data yang ada. Operasi aritmatika berupa penjumlahan dan pengurangan dan operasi logika berupa OR, AND, EX-OR dan NOT. Operasi penjumlahan dapat dilakukan dengan menggunakan rangkaian half-adder (penjumlah sebagaian) atau full-adder (penjumlah penuh).
Cara pertama adalah dengan mengubah data masukan pegurang (bit pengurang) menjadi data negatif (mengkomplemenkan data masukan pengurang) dan menjumlahkan kedua data tersebut. Cara kedua adalah dengan menggunakan rangkaian pengurang. Rangkaian pengurang dapat menggunakan rangkaian half-subtractor (pengurang sebagian) atau full-subtractor (pengurang penuh).

Pada arithmetic logic unit juga terdapat operasi pergeseran. Fungsi yang ada pada penggeser ini adalah menggeser data masukan ke kiri, ke kanan, dan tanpa pergeseran. Kondisi logika bit rendah akan mengisi data hasil pergeseran yang kosong. Memory pada sebuah mikroprosesor digunakan sebagai tempat penyimpanan data sementara data, hasil operasi, dan bit-bit kendali. Memory ini bersifat volatile. Memory ini dapat diakses secara cepat dan terdiri dari beberapa register.

Input/output interface merupakan suatu rangkaian yang mengatur data keluar dan masuk dari dan ke mikroprosesor. Rangkaian untuk antarmuka ini berbeda untuk tiap tiap mikroprosesor. Bila semua bagian diatas digabungkan maka dapat disusun sebuah mikroprosesor. Susunan arsitektur umum sebuah mikroprosesor dapat dilihat pada Gambar 6.



Gambar 6. Arsitektur umum mikroprosesor

Secara garis besar kerja yang terdapat pada mikroprosesor adalah sebagai berikut: Instruksi yang dijalankan oleh mikroprosesor ada di memori, berupa urutan data-data biner yang merupakan bahasa mesin mikroprosesor. Mikroprosesor mengambil instruksi biner tersebut dari memori yang ditunjuk oleh sebuah register yang bernama Program Counter atau register PC. Mula-mula bus alamat diisi dengan informasi alamat di mana letak instruksi berikutnya yang hendak dijalankan dengan register PC. Lalu mikroprosesor mengambil instruksi tersebut melalui bus data dan menyimpannya di Instruction Register atau register IR. Selanjutnya isi register PC ditambah satu, dengan demikian akan menunjuk ke alamat memori berikutnya di mana instruksi berikutnya akan dijalankan lagi. Secara simbolik kejadian di atas dapat dituliskan sebagai berikut:

$$\mathbf{Mem(PC) \rightarrow IR}$$
$$\mathbf{PC + 1 \rightarrow PC}$$

Apabila instruksi yang sudah terambil belum merupakan instruksi yang utuh (setiap instruksi bisa tersusun atas lebih dari 1 byte) maka kejadian di atas diulang lagi.

Setelah register IR berisi instruksi biner, unit kendali (*control unit*) lalu menerjemahkannya dan mengeksekusinya. Apa yang dilakukan oleh mikroprosesor tergantung dari instruksi yang diberikan tersebut. Misalnya instruksinya adalah operasi menjumlahan isi register B dengan isi suatu memori dan hasilnya disimpan di dalam register B lagi (alamat memori yang hendak ditambahkan merupakan bagian dari instruksi), maka operasi yang akan dijalankan adalah oleh mikroprosesor adalah:

$$\mathbf{Mem(PC) \rightarrow MA}$$
$$\mathbf{PC + 1 \rightarrow PC}$$
$$\mathbf{B + Mem(MA) \rightarrow B}$$

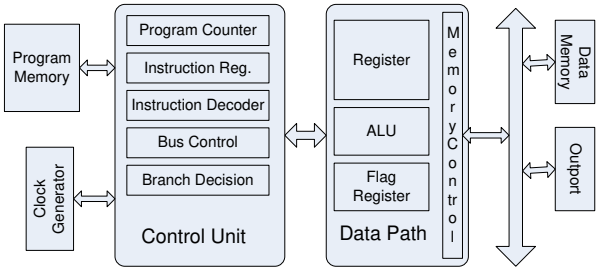
Perancangan Sistem

Arsitektur Prosesor

Prosesor yang akan dibangun memiliki ciri-ciri sebagai berikut:

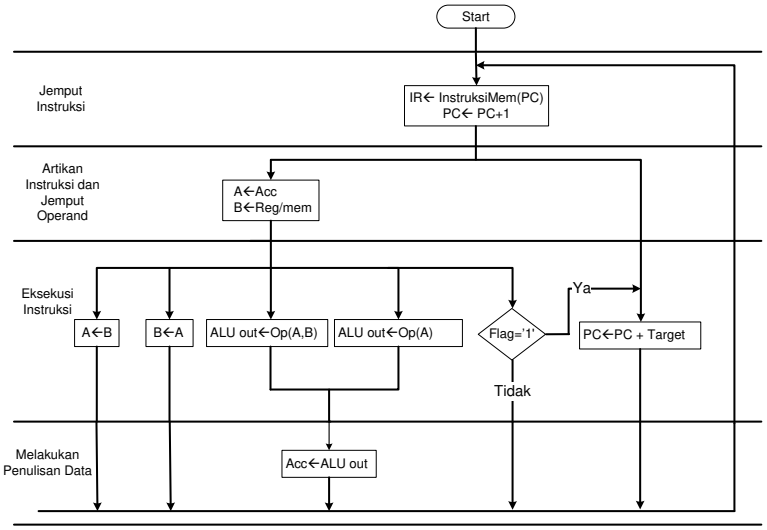
- memiliki 7 *register file* 8 *bit*, 7 *internal Data Memory* 8 *bit*, dengan pengantara 1 terminal keluaran 8 *bit* dan 1 terminal masukan 8 *bit*,
- 16 *bit bus* instruksi, 10 *bit bus* alamat instruksi, 8 *bit* data
- 3 *bit bus* control internal
- maksimum pengalamatan ROM adalah 1 Kbyte.

Prosesor tersebut akan dibangun menggunakan modul FPGA Xilinx Spartan XC3S200 FPGA XC3S200FT256. Semua batasan perangkat keras dari FPGA akan menjadi batasan perangkat yang dirancang. Hal ini termasuk kemampuan respon dari prosesor terhadap perubahan-perubahan masukan dari luar sistem. Arsitektur prosesor yang akan dirancang dapat dilihat pada Gambar 7.



Gambar 7. Arsitektur Prosesor

Pada Gambar 8 menunjukkan diagram alir dari prosesor yang dibuat. Diagram alir ini menunjukkan proses yang terjadi pada prosesor dan langkah-langkah yang akan dilakukan berikutnya untuk setiap intruksi.



Gambar 8. Diagram Alir

Kumpulan Instruksi Mikroprosesor

Jenis-jenis instruksi yang akan dirancang ditunjukkan pada Tabel 1. Instruksi-instruksi ini dikelompokkan berdasarkan jenis operasinya.

Jurnal Teknovasi
Volume 04, Nomor 01, 2017, 10-27
ISSN : 2355-701X

Tabel 1. Instruksi Berdasarkan Operasi

Jenis Operasi	Nama Instruksi
Tidak ada operasi	NOP
Lompatan Bersyarat	Jump if Zero
	Jump if Carry
	Jump if Accumulator Zero
Lompatan Tak Bersyarat	Unconditional Jump
Membaca dan menyimpan	Load Accumulator with immediate data
	Load Register with immediate data
	Load Memory with immediate data
	Load Accumulator with Register data
	Load Register with Accumulator data
	Load Register with Register data
Aritmatika	Addition
	Addition Immediate
	Subtractor
	Subtractor Immediate
	Increment data in Accumulator
	Decrement data in Accumulator
Logika	AND Accumulator and Register
	AND Accumulator and Immediate
	OR Accumulator and Register
	OR Accumulator and Immediate
	XOR Accumulator and Register
	XOR Accumulator and Immediate
	NOT A
Rotasi	Rotate Left Accumulator
	Rotate Left Accumulator with carry
	Rotate Right Accumulator
	Rotate Right Accumulator with carry

Jurnal Teknovasi
Volume 04, Nomor 01, 2017, 10-27
ISSN : 2355-701X

Jenis jenis operasi yang dapat ditangani oleh mikroprosesor yang akan dirancang ini adalah lompatan bersyarat, lompatan tak bersyarat, membaca dan menyimpan data dari instruksi, register, memory dan akumulator ke register, memory dan akumulator, operasi aritmatika penjumlahan dan pengurangan, operasi logika XOR, AND, OR dan NOT serta operasi rotasi data. Untuk operasi logika NOT, dan rotasi data hanya dapat dilakukan pada akumulator.

Format Instruksi

Intruksi akan dirancang dengan panjang yang tetap yaitu 16 bit untuk satu instruksi. Jenis format instruki yang akan digunakan adalah :

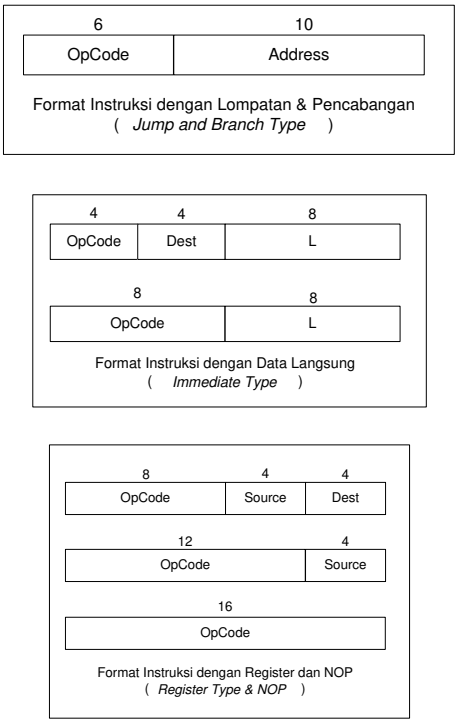
1. *Register type*,
2. *Immediate type*,
3. *Jump / Branch type*,

Panjang masing-masing field ditentukan berdasarkan faktor-faktor sebagai berikut:

- Jangkauan alamat yang digunakan untuk instruksi lompatan bersyarat dan tak bersyarat,
- Jumlah *register* dan *data memory*,
- Menggunakan instruksi dengan 2 operand
- Jumlah instruksi; menggunakan 28 buah instruksi.

Format instruksi yang dirancang akan disusun berdasarkan pembagian *field-field*. Keterangan masing-masing *field* adalah:

OpCode: Kode instruksi, disusun dengan jumlah bit yang bervariasi agar instruksi yang disusun dapat dimuati alamat register sumber maupun target dan data langsung
Address : lokasi alamat tujuan dalam pencabangan atau lompatan (10 *bit*)
L : data langsung (8 *bit*)
Source : sumber data (4 *bit*)
Dest : alamat tujuan (4 *bit*)



Gambar 9. Format Instruksi

Jurnal Teknovasi
Volume 04, Nomor 01, 2017, 10-27
ISSN : 2355-701X

Detail Rancangan

Rancangan prosesor dibagi menjadi dua kelompok besar yaitu rancangan *control unit* dan *datapath* seperti terlihat pada Gambar 9. *Datapath* terdiri dari *register file* dan ALU (*Arithmetic Logic Unit*), sehingga sebuah prosesor memiliki tiga komponen utama yaitu *control unit*, *register file*, dan ALU).

Control Unit

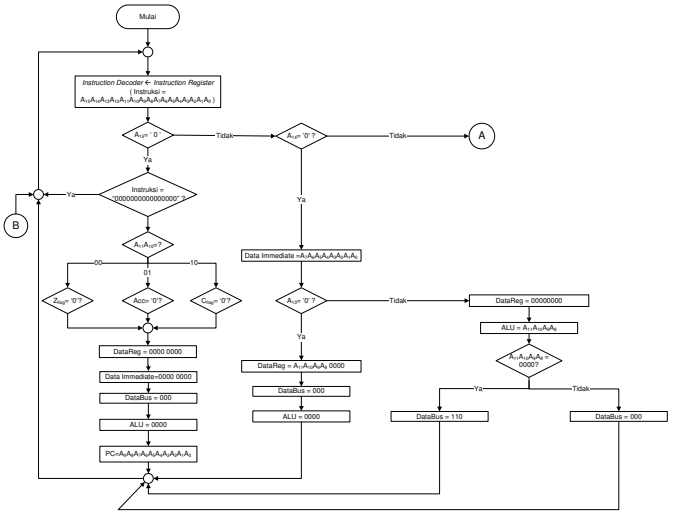
Control unit berfungsi untuk mengatur jalannya operasi pada prosesor dimana memperoleh masukkan berupa kondisi dari datapath melalui sinyal status untuk memutuskan instruksi selanjutnya yang akan diambil. Control unit akan mengeluarkan alamat instruksi ke memori instruksi untuk mengambil instruksi yang akan dikerjakan sebagai masukkannya. Masukkan lain dari control unit berupa instruksi (data program). Kemudian instruksi akan dikodekan dan dikirim melalui sinyal kontrol ke datapath untuk memberitahukan apa yang harus dikerjakan oleh datapath. *control unit* memiliki enam buah masukan dan enam buah keluaran. Dari keenam masukan tiga data masukannya berupa data flag yang berasal dari datapath. Masukan program berfungsi untuk menerima masukan instruksi, clr berfungsi untuk melakukan *reset*, dan clk merupakan masukan pewaktu bagi mikroprosesor. Keluaran dari *control unit* ini sebagian besar menjadi masukan bagi *datapath*. Keluaran Clk_Dp merupakan pewaktu bagi *datapath*, DataAlu merupakan sinyal kendali bagi ALU, DataReg memuat informasi *register* dan lokasi memori data yang dipakai sebagai operand, DataBus mengandung informasi kendali bus data, dan DI merupakan *immediate data* yang akan dibaca oleh *datapath* bila instruksi yang diberikan adalah instruksi *immediate type*. Untuk dapat menyusun pengartian instruksi maka diperlukan susunan format instruksi yang jelas. *Control unit* ini mampu bekerja dan menjalankan instruksi dalam 1 *clock cycle*. Tabel 2 menunjukkan pendekodean data yang dilakukan oleh *instruction decoder* dan beberapa keluaran dari *control unit*.

Tabel 2. Pendekodean instruksi

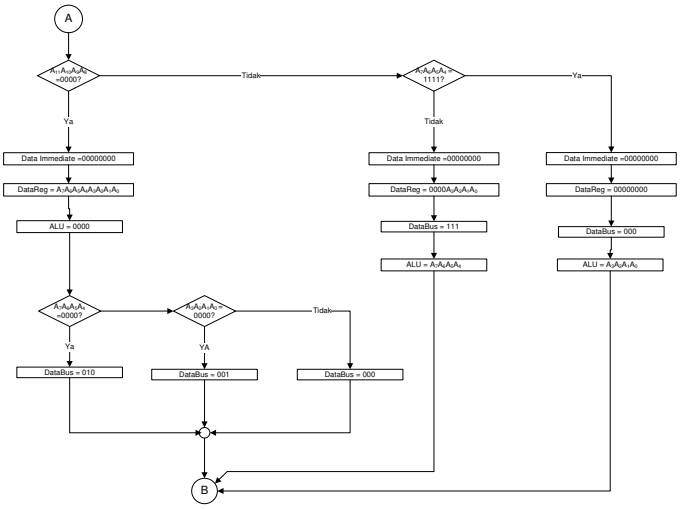
Mnemonic	Instruksi	DataBus	ALU	DataReg
NOP	0000 0000 0000 0000	000	0000	00000000
JZ	0000 00AA AAAA AAAA	000	0000	00000000
JIFAZ	0000 01AA AAAA AAAA	000	0000	00000000
JC	0000 10AA AAAA AAAA	000	0000	00000000
JMP	0000 11AA AAAA AAAA	000	0000	00000000
MOV R,L	1000 RRRR LLLL LLLL	100	0000	RRRR0000
MOV A,L	1011 0000 LLLL LLLL	110	0000	00000000
AND A,L	1011 0001 LLLL LLLL	000	0001	00000000
OR A,L	1011 0010 LLLL LLLL	000	0010	00000000
XOR A,L	1011 0011 LLLL LLLL	000	0011	00000000
ADD A,L	1011 0100 LLLL LLLL	000	0100	00000000
SUB A,L	1011 0101 LLLL LLLL	000	0101	00000000
MOV A,R	1111 0000 0000 RRRR	010	0000	0000RRRR
MOV R,A	1111 0000 RRRR 0000	001	0000	RRRR0000
MOV R,R	1111 0000 RRRR RRRR	000	0000	RRRRRRRR
AND A,R	1111 1111 0001 RRRR	111	0001	0000RRRR
OR A,R	1111 1111 0010 RRRR	111	0010	0000RRRR
XOR A,R	1111 1111 0011 RRRR	111	0011	0000RRRR
ADD A,R	1111 1111 0100 RRRR	111	0100	0000RRRR
SUB A,R	1111 1111 0101 RRRR	111	0101	0000RRRR
NOT A	1111 1111 1111 1001	000	1001	00000000
INC A	1111 1111 1111 1010	000	1010	00000000

DEC A	1111 1111 1111 1011	000	1011	00000000
ROL A	1111 1111 1111 1100	000	1100	00000000
RLC A	1111 1111 1111 1101	000	1101	00000000
ROR A	1111 1111 1111 1110	000	1110	00000000
RRC A	1111 1111 1111 1111	000	1111	00000000

Keterangan dari tabel diatas:
LLLL LLLL : 8 *bit* data langsung;
AA AAAA AAAA : 10 *bit* alamat
RRRR : alamat register (untuk 0RRR) / memory (untuk 1RRR)
Diagram alir penguraian instruksi oleh blok *instruction decoder* dapat dilihat pada Gambar 10 dan Gambar 11.



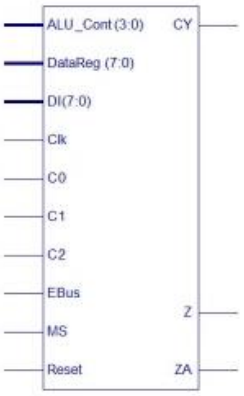
Gambar 10. Diagram Alir Penguraian Instruksi oleh Blok *Instruction Decoder*



Gambar 11. Diagram Alir Penguraian Instruksi oleh Blok *Instruction Decoder*

DataPath

Datapath memiliki keluaran berupa alamat atau data ke memori data dan memiliki masukkan berupa data dari memori data. Blok komponen ALU dan *register file* beserta komponen *memory control* dan *data bus*, dan *flag register* disusun pada datapath. *Datapath* memiliki beberapa masukan berupa alamat *register* yang akan ditulis dan dibaca, *immediate data*, data kendali ALU, *bus control data*, *pin clock*, dan *pin reset*. Keluaran dari *datapath* ini adalah data flag Z(*zero flag*) yang akan belogika '1' bila keluaran ALU adalah 00000000 dan berlogika '0' bila keluaran ALU tidak 00000000, ZA(*zero accumulator flag*) berlogika '1' bila ALU bernilai 00000000 dan berlogika '0' bila ALU tidak bernilai 00000000, dan CY(*carry flag*) yang berlogika '1' bila hasil aritmatika pada ALU menghasilkan *carry*. Pada Gambar3.6 ditunjukkan simbol blok dari *datapath*.



Gambar 12. *DataPath*

PENGUJIAN SISTEM

Pengujian Software

Keseluruhan sistem yang telah dirancang dan dibuat dengan menggunakan VHDL, kemudian digabungkan menjadi mikroprosesor. Hasil penggabungan ini diuji untuk mengetahui bahwa sistem dapat bekerja dengan baik. Pengujian dengan simulasi ini menggunakan software simulasi yang telah terintegrasi dengan software Xilinx ISE 8.1i. Simulasi dilakukan dengan melakukan double click pada "Simulate Behavioral Model" pada jendela Process.

Pengujian Hardware

Pengujian dilakukan dengan menerapkan program pada mikroprosesor yang dirancang melalui sebuah ROM yang dirancang dan diisi dengan program. Pengukuran pada pin FPGA dilakukan dengan menggunakan osiloskop dan led pada papan FPGA untuk mengetahui apakah mikroprosesor yang dirancang dapat bekerja sesuai dengan yang diharapkan.

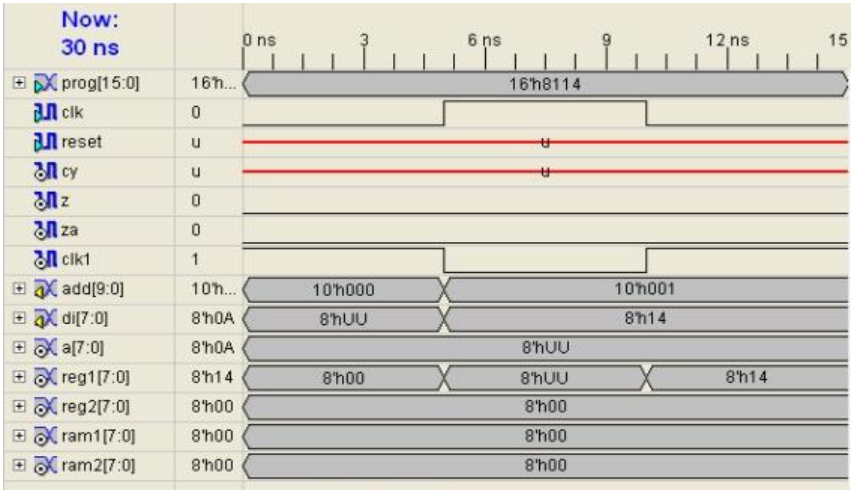
HASIL DAN PEMBAHASAN

Pengujian Software

Pada pengujian *software*, pengujian dilakukan secara simulasi dengan menggunakan tampilan bentuk gelombang yang ditampilkan pada jendela *simulate behavioral model* .

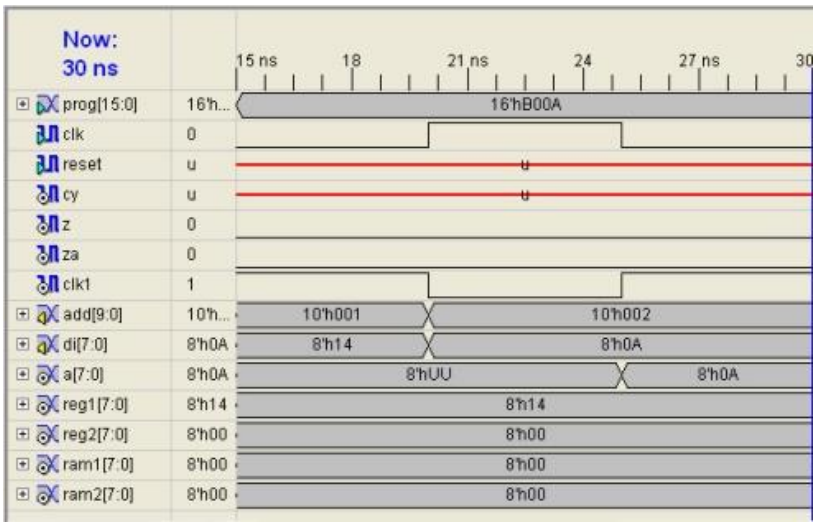
Pengujian yang dilakukan adalah sbb:

1. *Pengetesan perintah MOV Register, #dataimmediate*
   Register : Reg 1
   Data Langsung : 14_{16} =00010100_2
   Instruksi : 1000 0001 00010100
   Perintah ini akan memasukkan data langsung yang terdapat pada instruksi ke register Reg 1 .
   Waveform diagram dapat dilihat pada Gambar 13.



Gambar 13. Waveform Diagram 1

2. *Pengetesan perintah MOV Accumulator, #dataimmediate*
   Data Langsung : $0A_{16}$ = 00001010_2
   Instruksi : 1011 0000 00001010
   Perintah ini akan memasukkan data langsung yang terdapat pada instruksi ke akumulator .
   Waveform diagram dapat dilihat pada Gambar 14.



Gambar 14. *Waveform Diagram 2*

Pengujian Hardware
Persiapan Pengujian

ROM yang dirancang memiliki fungsi untuk menyimpan data program penguji yang akan dieksekusi pada saat melakukan evaluasi. Jumlah maksimum alamat yang dapat disimpan oleh ROM ini 1 Kbyte. Terdapat 4 program penguji yang akan dibuat dengan tujuan untuk menunjukkan apakah prosesor yang dirancang berjalan sesuai dengan apa yang diharapkan. Sebelum melakukan pengujian terhadap perangkat keras, pengujian terlebih dahulu disimulasikan menggunakan software Xilinx ISE 8.1i. Setelah program yang disimulasikan dapat berjalan sesuai yang diinginkan, program tersebut diujikan pada perangkat keras.

Pelaksanaan Evaluasi

Pengujian dilakukan terhadap penggunaan instruksi baca tulis I/O, pengalamatan PC (Program Counter), operasi logika, operasi aritmatika dan lompatan bersyarat. Gambaran umum ke-2 program pengujian tersebut adalah sebagai berikut:

1. Test : Pengujian Terhadap Pengalamatan Pada PC

Pengujian ini dilakukan dengan mengisikan instruksi ke semua lokasi memori program dan instruksi yang diberikan hanya melakukan lompatan bila telah mencapai alamat terakhir dari PC. Pada program penguji ini akan dilkukan penulisan data ke port keluaran yang datanya berubah-ubah untuk mengetahui bahwa program tersebut bekerja.

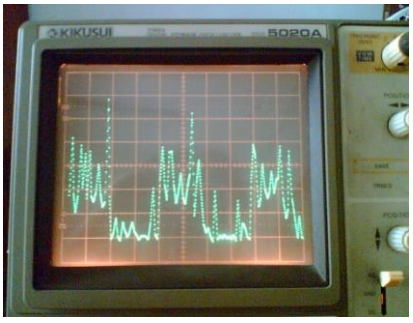
Tabel 3. Urutan Instruksi Program Penguji 1

Alamat	Instruksi	Mnemonic
0000 0000 0000 0000	0000 0000 0000 0000	NOP
0000 0000 0000 0001	0000 0000 0000 0000	NOP
0000 0000 0000 0010	10110000000 01111	MOV A,#FH

Jurnal Teknovasi
Volume 04, Nomor 01, 2017, 10-27
ISSN : 2355-701X

0000 0000 0000 0011	11111111111 11001	NOT A
0000 0000 0000 0100	11110000111 10000	MOV P1,A
0000 0000 0000 0101 s/d 1111 1111 1111 1110	0000 0000 0000 0000	NOP
1111 1111 1111 1111	01110000000 00011	JMP #3H

Frekwensi kerja mikroprosesor ini adalah 50 Mhz yang diambil dari pembangkit pulsa yang terdapat pada modul FPGA. Urutan instruksi yang dipakai dapat dilihat pada Tabel 3. Sinyal keluaran dari mikroprosesor yang dirancang ini diamati dengan menggunakan osiloskop.



Gambar 15. Sinyal Pada Kaki Keluaran P1.0

2. Test : Pengujian Terhadap Operasi Logika dan Aritmatik.

Pengujian ini dilakukan dengan membaca data dari port masukan, menjumlahkan 4 *bit* awal (*bit* 3, *bit* 2, *bit* 1 dan *bit* 0) dengan 4 bit terakhir (*bit* 7, *bit* 6, *bit* 5 dan *bit* 4) dan menampilkan data hasil penjumlahan ke port keluaran. *Flowchart* dari program ini dapat dilihat pada Gambar.. Frekwensi kerja mikroprosesor ini adalah 50 Mhz yang diambil dari pembangkit pulsa yang terdapat pada modul FPGA. Urutan instruksi yang dipakai dapat dilihat pada Tabel 4 dan hasil pengujian progam ini dapat dilihat pada Tabel 4.

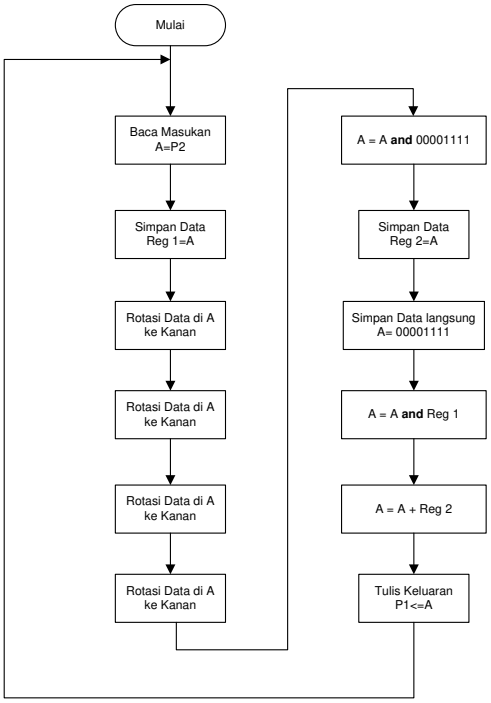
Tabel 4. Urutan Instruksi Program Penguji 2

Alamat	Instruksi	Mnemonic
"0000000000"	"0000000000000000"	NOP
"0000000001"	"0000000000000000"	NOP
"0000000010"	"1111000000001111"	MOV A,P1
"0000000011"	"1111000000010000"	MOV REG1,A

"0000000100"	"1111111111111110"	ROR A
"0000000101"	"1111111111111110"	ROR A
"0000000110"	"1111111111111110"	ROR A
"0000000111"	"1111111111111110"	ROR A
"0000001000"	"1011000100001111"	AND A,#FH
"0000001001"	"1111000000100000"	MOV REG2,A
"0000001010"	"1011000000001111"	MOV A,#FH
"0000001011"	"1111111100010001"	AND A,REG1
"0000001100"	"1111111101000010"	ADD A,REG2
"0000001101"	"1111000011110000"	MOV P1,A
"0000001110"	"0111000000000001"	JMP #01H

Tabel 5. Pembacaan Port Masukan dan Keluaran

P1 (Out)		P2 (In)	
Bit	Bit	Bit	Bit (3,2,1,0)
0001	0111	1001	1110
0000	0111	0010	0101
0001	0010	0011	1111
0000	1011	0110	0101
0001	0000	1111	0001



Gambar 15. *Flowchart* Program Penguji 2

KESIMPULAN

Berdasarkan hasil pengujian dan analisa penelitian ini diperoleh bahwa proses mikroprosesor 8 *bit* pada FPGA Xilinx Spartan 3 XC3S200FT256 dengan menggunakan VHDL dapat berjalan dengan baik dan cukup memuaskan, dimana berdasarkan pengujian secara software dan hardware yang dilakukan, pin pada mikroprosesor menghasilkan grafik sinyal yang sesuai dengan yang diinginkan Simulasi program VHDL menggunakan *software* Xilinx ISE 8.1i dan evaluasi perangkat keras dengan program pengujian dapat dilaksanakan untuk membuktikan bahwa mikroprosesor dapat bekerja dengan baik. Pengujian yang dilakukan pada penelitian ini dilakukan lebih dari yang dipaparkan disini. Untuk pengembangan pengujian selanjutnya, dapat mengembangkan pengujian dengan mengintegrasikan memori serta manajemen port sehingga dapat merubah fungsi mikroprosesor menjadi mikrokontroller, menambahkan fungsi pengolahan citra, serta menerapkan fungsi pembacaan sinyal analog.

DAFTAR PUSTAKA

Anonim, (2006), http://www.gmvhdl.com/ VHDL.html
Bhasker, J. (1999). *A VHDL Primer*, Prentice Hall, New Jersey.
Mano, M., M. (2002). *Digital Design* 3rd *Edition*, Prentice-Hall.
Maxfield, C. (2004). *Design Warriors Guide to FPGA*, Newnes, Mentor Graphics Corporation and Xilinx, USA.
Parnell, Karen & Mehta, N. (2004). *Introduction to Programmable Logic*, Xilinx, San Jose.
Tanenbaum, A., S. (1990). *Structured Computer Organization*, Prentice-Hall.
Zargham, M., R. (1996). *Computer Architecture, Single and Parallel Systems*, Prentice-Hall International.