ISSN. 1412-0100 VOL 16, NO 1, APRIL 2015

Perancangan Dan Implementasi Soft Processor Berdasarkan Arsitektur OpenRISC 1200 pada FPGA

Dhita Bagus Ananda Nurman[1], R.Rumani M[2], Denny Darlis[3]
Universitas Telkom, Jl. Telekomunikasi Nomor 1, Tel. 022-7564108, Dayeuhkolot, Bandung 40257
[1,2,3]Fakultas Teknik Elektro, Universitas Telkom
[1]dbagus.ananda@gmail.com, [2]rumani@telkomuniversity.ac.id, [3]dad@ittelkom.ac.id

Abstrak

Soft processor adalah prosesor yang dapat disintesis, diprogram, dan disimulasikan lewat suatu bahasa pemrograman untuk kemudian ditanamkan pada sebuah chip semikonduktor berisi logika yang dapat diprogram ulang seperti teknologi FPGA. Field Programable Gate Array (FPGA) merupakan suatu Integrated Circuit (IC) yang dibuat agar dapat diprogram berulang kali sesuai dengan kebutuhan. OpenRISC 1200 (OR1200) adalah sebuah soft processor yang mengimplementasikan arsitektur RISC 32-bit. Penelitian ini membahas tentang implementasi arsitektur umum openRISC 1200 soft processor dalam bentuk implementasi System-on-Chip (SoC) pada FPGA. Dari hasil penelitian, ternyata sistem dapat berjalan di FPGA Xilinx Spartan-6 LX45 board Atlys™. Hal ini dapat dibuktikan dengan melakukan pengujian, yaitu menjalankan program aplikasi sederhana pencacah bilangan prima. Hasilnya adalah keluaran dari FPGA sama dengan keluaran hasil simulasi dengan or1ksim. Dari hasil simulasi dapat dilihat bahwa prosesor OpenRISC dapat bekerja menjalankan program aplikasi tersebut dengan ISA yang sesuai dengan standar OpenRISC.

Kata kunci— OpenRISC, SoC, FPGA, Spartan-6, Atlys

Abstract

Soft processor is a processor that can be synthesized, programmed, and simulated by programming language so that later it can be planted in a reprogrammable micro semiconductor chip like FPGA technology. Field Programable Gate Array (FPGA) is an Integrated Circuit (IC) that can be reprogrammed as we need. This characteristic of FPGA make it has more benefit from other IC product, which is low cost to reprogrammed in case of bugs. OpenRISC 1200 (OR1200) is a soft processor that implement 32-bit RISC architecture processor. This final project explains about implementing process of general architecture of openRISC 1200 soft processor in System-on-Chip in FPGA. System is running on FPGA Xilinx Spartan-6 LX45 board Atlys™. It can be seen by doing a test, which is to run a simple application program to count prime numbers. And the outputs from FPGA are the same as the outputs from simulation process. And the simulation process itself shows that OpenRISC is running the application by using standard ISA of OpenRISC.

Keywords— *OpenRISC, SoC, FPGA, Spartan-6, Atlys*

1. PENDAHULUAN

Perkembangan sistem mikroprosesor saat ini semakin menekankan pada perancangan arsitektur prosesor yang lebih kecil, fleksibilitas yang lebih tinggi, dan ketidaktergantungan dari jenis *platform.* Kini dengan perkembangan mikro elektronik, yang dapat memuat banyak gerbang logika dalam satu *chip*, menjadikan FPGA menjadi salah satu produk *Integrated Circuit* (IC) yang dapat

dikembangkan.

Prosesor merupakan salah satu komponen perangkat keras utama dari sistem komputer yang mempunyai fungsi untuk pengontrolan dan pengolahan data. Sedangkan *soft processor* adalah prosesor yang dapat diprogram dan disimulasikan lewat suatu bahasa pemrograman untuk kemudian ditanamkan pada sebuah *chip* semikonduktor berisi logika yang dapat diprogram ulang. *OpenRISC* adalah sebuah *soft processor* yang arsitekturnya kini dikembangkan oleh komunitas *Opencores*. Proyek yang dijalankan komunitas ini bertujuan untuk mengembangkan beberapa arsitektur *general purpose* RISC CPU. Salah satunya adalah *soft processor openRISC* 1200. *OpenRISC* 1000 adalah suatu bentuk arsitektur prosesor yang masih belum diimplementasikan pada *chip*. Sedangkan *OpenRISC* 1200 adalah pengimplementasian arsitektur *OpenRISC* 1000 pada *chip* FPGA. Lisensi yang bersifat *open source* sesuai GNU *Lesser General Public License* (LGPL) membuat *OpenRISC* 1200 dapat dimodifikasi dan dikembangkan oleh individu.

2. DASAR TEORI

2.1. Prosesor

Prosesor adalah suatu komponen dalam sebuah komputer yang dapat menjalankan beberapa set instruksi dari suatu program komputasi dengan menjalankan perintah dasar aritmatik, logika, dan operasi *input/output* dari suatu sistem komputer tersebut[1]. Bentuk, rancangan blok, dan implementasi dari prosesor mengalami perkembangan yang sangat pesat sampai saat ini. Namun, operasi dasar dan fungsi dari prosesor sendiri relatif tetap sama. Perkembangan di dalamnya termasuk munculnya istilah *System on Chip* (SoC).

System on Chip adalah sebuah *Integrated Circuit* (IC) yang menggabungkan semua komponen dari sebuah komputer atau sistem elektronik dalam sebuah *chip*[2]. Penggunaanya yang luas, karena dapat menerapkan sistem elektronik digital maupun analog, membuat *System on Chip* relatif lebih unggul dan berkembang pesat daripada mikrokontroler yang hanya mampu menerapkan suatu proses tertentu.

Fungsi utama prosesor adalah melakukan operasi aritmatika dan logika terhadap data yang diambil dari memori atau dari informasi yang dimasukkan melalui beberapa perangkat keras[1].
Instruksi-intruksi dijalankan dalam bentuk siklus oleh prosesor. Siklus ini berulang terus-menerus selama program yang dijalankan oleh prosesor tetap berjalan. Siklus ini pada umumnya terdiri dari 4 tahap proses, yaitu *fetch, decode, execute* dan *write back* [2].

Beberapa prosesor memiliki 5 proses dalam memproses instruksi [3]. Selain proses tersebut di atas, ada proses lain yaitu proses *Save instruction*, atau simpan instruksi, proses terakhir ini menyimpan instruksi yang sudah dijalankan pada *cache*. Prosesor yang menggunakan 5 *step* dalam memproses instruksi salah satunya adalah *Reduced Instruction Set Computer* (RISC).

Reduced Instruction Set Computer adalah suatu bentuk rancang prosesor yang menekankan pada pengimplementasian instruksi yang sederhana.Tujuan dari penekanan ini adalah untuk meningkatkan unjuk kerja (performa) prosesor. Dengan instruksi yang sederhana, maka kecepatan eksekusi instruksi menjadi lebih cepat.

2.2. *Soft Processor OpenRISC 1200*

Untuk memahami *Soft Processor OpenRISC* 1200, kita perlu memahami terlebih dahulu mengenai konsep *Soft Processor.*

2.2.1. *Soft Processor*

Soft processor atau *soft core processor* adalah sebuah model prosesor yang arsitektur dan perilaku kerjanya secara spesifik dinyatakan dengan menggunakan suatu *subset* yang dapat disintesis dari suatu *Hardware Description Language* (HDL).*Soft processor* dapat disintesis pada bermacam-macam teknologi *Application-Specific Integrated Circuit* (ASIC) atau *Field Programable Gate Array*

(FPGA). Dengan demikian, *soft processor* memberikan tingkat fleksibilitas yang penting bagi para perancang sistem (s*ystem designer*) [4].

Penggunaan *soft processor* memberikan banyak manfaat untuk para perancang sistem tertanam (*embedded system*), diantaranya sebagai berikut [5]:

a. Pertama, *soft-prosesor* fleksibel dan dapat diatur penggunaannya untuk suatu aplikasi yang spesifik dengan relatif lebih mudah.
b. Kedua, *soft prosesor* memiliki kebebasan penggunaan teknologi dan dapat disintesis pada jenis teknologi AISC atau FPGA.
c. Ketiga, arsitektur dan perilaku *soft processor* dideskripsikan pada tingkatan abstraksi yang lebih tinggi menggunakan suatu HDL.

Salah satu contoh *soft processor* yang kini sedang dikembangkan, adalah *OpenRISC* 1200.

2.2.2. *OpenRISC* 1200

*Soft processor OpenRISC*1200[5](OR1200) adalah sebuah *soft processor* yang arsitekturnya kini dikembangkan oleh komunitas *Opencores.* Tujuan dari proyek *OpenRISC* adalah membuat komputasi *platform* yang gratis dan *open source*[6]. Desain OR1200 adalah implementasi dari arsitektur *OpenRISC* 1000.

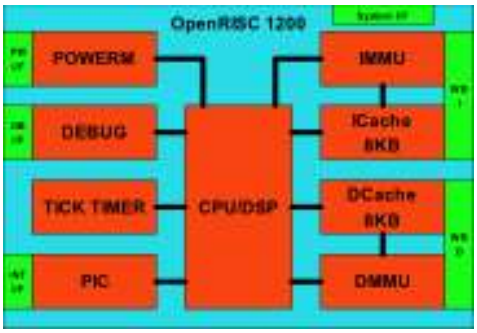
Arsitektur *OpenRISC* 1000 (OR1K) adalah sebuah arsitektur 32/64-bit *load and store* RISC yang dirancang dengan mengandalkan pada performansi, sederhana, kebutuhan daya yang rendah, dan kemampuan skalabilitas. *OpenRISC* 1000 bekerja pada suatu spektrum *chip* dan implementasi sistem dengan beragam harga/performansi untuk suatu rentang aplikasi[6]. *OpenRISC* 1000 adalah arsitektur *OpenRISC* pertama yang di dalamnya mendefinisikan set instruksi, aturan pengalamatan, sistem eksepsi, set register, mekanisme perubahan konteks secara cepat, antarmuka *cache*, antar muka *Memory Management Unit* (MMU), dan *Application Binary Interface* (ABI)[5].

OpenRISC dapat diimplementasikan ke dalam bentuk SoC dan disebut dengan *OpenRISC Reference Platform System on Chip* (ORPSoC). Dengan ORPSoC, pengujian dan pengembangan prosesor *OpenRISC* dan SoC dapat diimplementasikan pada berbagai macam FPGA[2].

OpenRISC 1200 adalah suatu prosesor yang dapat disintesis dan merupakan implementasi *open source* dari arsitektur RISC *OpenRISC* 1000.*OpenRISC* 1200 adalah suatu prosesor RISC 32-bit skalar dengan arsitektur Harvard, 5 tingkat integer *pipeline*, *virtual memori* (MMU) dan kemampuan DSP dasar. Spesifikasi utama *OpenRISC* 1200 adalah sebagai berikut:

a. 32-bit skalar RISC prosesor dengan model arsitektur Harvard.
b. 5 tahap integer *pipeline.*
c. 64-entry dengan sistem *hash-based 1-way direct-mapped vitual memory* (MMU) berbeda untuk data dan instruksi.
d. 8 KB *cache* yang berbeda untuk data dan instruksi masing-masing ukurannya 16-*byte line.*
e. Operasi MAC 32x32 pada DSP unitnya.

ArsitekturOR1200[5] terdiri dari beberapa blok sesuai gambar berikut:



Gambar 1 Arsitetur Standar dari IP *core* OR1200[6]

2.2.3. *Or1ksim*

OpenRISC 1000 *Simulator* (Or1ksim) adalah suatu program simulasi yang bekerja sebagai fungsi untuk menjalankan suatu arsitektur *OpenRISC* pada suatu *platform* tertentu sesuai implementasi HDL [1]. Beberapa fitur yang terdapat pada Or1ksim, antara lain [4]:

a. Gratis, *open source code*, dan dapat dipakai sebagai simulator tunggal atau sebagai *library*
b. Simulasi arsitektur dengan level yang tinggi dan cepat sehingga perancang dapat memberikan analisis dan evaluasi performansi sistem
c. Mudah dikonfigurasi pada bermacam-macam target implementasi

Fungsi programnya dapat dijalankan dengan mudah dengan cara menjalankan perintah "*or32-elf-sim"* pada terminal *platform open source* Linux setelah sebelumnya menambahkan *PATH* yang merujuk pada folder binary tempat instalasi or1ksim.

2.2.4. GNU Toolchain

GNU *Toolchain* adalah istilah untuk suatu set perangkat pemrograman yang dibuat oleh GNU *Project*. GNU *toolchain* sangat penting dalam pengembangan *kernel* Linux, BSD, dan perangkat lunak untuk *embedded system* (sistem tertanam).

Fungsi-fungsi yang dipakai misalnya "*or32-elf-gcc*", yaitu perintah untuk melakukan *compile* pada suatu sintaks program; juga "*or32-elf-objdump*", yaitu perintah untuk menampilkan *listing program* dalam bahasa Assembly [8].

2.3. Field Programmable Gate Array (FPGA)

FPGA adalah singkatan dari *Field Programmable Gate Array* [7] yang merupakan suatu IC yang dapat diprogram untuk melakukan fungsi-fungsi logika tertentu. FPGA dapat diprogram oleh pengguna (*end user*), tersusun dari sekumpulan sel-sel yang dihubungkan satu sama lain melalui interkoneksi.

Beberapa kelebihan dari FPGA antara lain [8] :

a. FPGA lebih murah jika dibandingkan dengan PLD dan ASIC.
b. FPGA lebih mudah dan cepat dari sisi implementasi.
c. FPGA memilki kelebihan dari PLD yaitu dapat diprogram berulang-ulang.
d. FPGA bisa digunakan untuk membangun sistem perangkat keras yang spesifik serta memakan cukup banyak *resource* seperti pengolahan sinyal, memori, bahkan mikrokontroller.

Secara umum, arsitektur bagian dalam dari IC FPGA terdiri atas tiga elemen utama yaitu *Input/Output Blok* (IOB), *Configurable Logic Block* (CLB) dan *Programmable Interconnect* [8].

a. *Configurable Logic Blocks*, yang memiliki fungsi :
   1. Memiliki *Look up table* berdasarkan struktur kompleks komponen dalam FPGA.
   2. Mengimplementasikan rangkaian sekuensial
b. *Programmable Interconnect*, yang berfungsi:
   1. Berisi *wire segments* (kabel penyambungan antarsegmen) dan *programmable switches* (*port switch* yang biasa diprogram sebagai *input* rangkaian).
   2. Penghubung antara *Configurable Logic Blocks* yang berbeda
c. *Input/output block*, yang memiliki fungsi :
   Sebagai *interface* antara *external package pin* dari perangkat dan *internal user logic* pada FPGA.

3. PERANCANGAN SISTEM

Dalam bagian ini dibahas mengenai perancangan sistem *soft processor OpenRISC* 1200. Perancangan ini meliputi spesifikasi sistem prosesor, program aplikasi yang diuji, proses simulasi, dan implementasi.

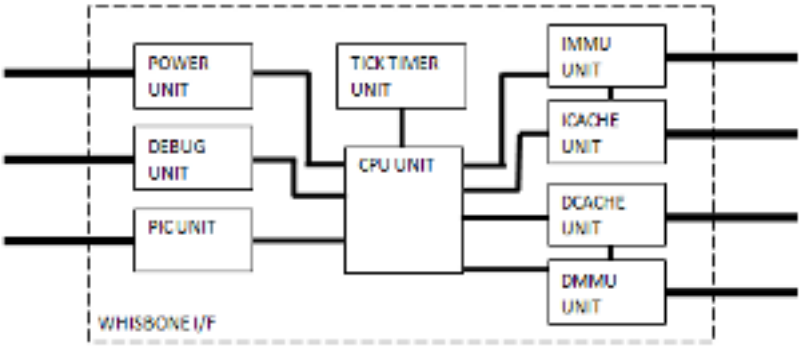
3.1. Spesifikasi Sistem Prosesor

Perancangan sistem difokuskan pada rancangan *core prosesor* yang memiliki ciri dan spesifikasi utama yaitu:

a. 32-bit skalar RISC prosesor dengan model arsitektur Harvard.
b. 5 tahap integer *pipeline.*
c. 64-entry dengan sistem *hash-based 1-way direct-mapped vitual memory* (MMU) berbeda untuk data dan instruksi.
d. 8 KB *cache* yang berbeda untuk data dan instruksi, dengan ukurannya masing-masimg sebesar 16-*byte line.*
e. Operasi MAC 32x32 pada unit DSP-nya.
f. Menghasilkan nilai *Dhrystone* sebesar 2.1 MIPS saat bekerja pada 300 MHz dengan menjalankan 0.18u proses.

3.2. Diagram Blok Sistem Prosesor

Sistem *soft prosesor* yang dibuat berdasarkan spesifikasi standar dari *OpenRISC* 1200. Berikut adalah blok diagram dari *OpenRISC* 1200 pada FPGA *chip*.



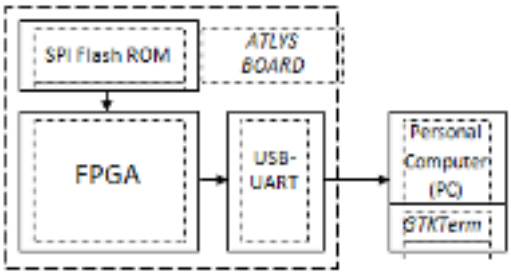
Gambar 2 Diagram Blok *OpenRISC* 1200

3.2.1. Central Processing Unit (CPU)

CPU adalah bagian utama dari sistem prosesor. Fungsi utamanya adalah menerima data dan instruksi serta *flag* dan memprosesnya sesuai dengan kode program. CPU *OpenRISC* 1200 hanya dapat memproses data yang panjangnya 32-bit.

3.3. Spesifikasi FPGA

Blok diagram *hardware* yang dilakukan pada penelitian ini digambarkan sebagai berikut:



Gambar 3 Blok diagram *Hardware*

Pembahasan terhadap hasil penelitian dan pengujian yang diperoleh disajikan dalam bentuk uraian teoritik, baik secara kualitatif maupun kuantitatif.Hasil percobaan sebaiknya ditampilkan dalam berupa grafik atau pun tabel. Untuk grafik dapat mengikuti format untuk diagram dan gambar.

4. Pengujian Sistem Dan Analisis Sistem Prosesor Pada FPGA

4.1. Sintesis Sistem

Proses sintesis umumnya dilakukan dengan cara menyatukan *code Verilog* HDL dalam proyek *Xilinx* ISE *Project Navigator*. Namun, pengembang *OpenRISC* sudah menulis dan menyertakan suatu *script* yang dapat menjalankan proses sintesis ini tanpa perlu menggabungkan *code* dalam satu *folder* dan menjalankan program ISE. *Script* ini dijalankan dalam *bash shell* dengan perintah: *make all*.

Proses selanjutnya adalah proses *Place And Route* (PAR). Proses ini dilakukan untuk mengetahui kebutuhan *resource* dari FPGA dan melakukan proses *mapping* sinyal-sinyal antara blok logika. Proses ini juga mudah dijalankan pada *bash shell* dengan perintah*: make orpsoc.ncd*. Dari proses ini dapat dilihat penggunaan *resource* FPGA yang dipakai untuk mengimplementasikan sistem. Hasil proses PAR tersimpan pada *log file: orpsoc.par.*

Laporan penggunaan *resource* FPGA dapat disimpulkan menjadi tabel berikut.

Tabel 1 Penggunaan *Resource* pada FPGA

Resources FPGA	Terpakai	Dari total	Persentase
Slices	4379	6822	64 %
Slices Flip-Flop	6721	54576	12 %
Slices 6 input LUTs	12534	27288	45 %

Tabel di atas menunjukkan bahwa kebutuhan minimal dari FPGA untuk dapat diimplementasikan sistem prosesor *OpenRISC* 1200 adalah sebanyak 4379 *slices. Slice* sendiri adalah satuan jumlah *Configurable Logic Block* (CLB) yang di dalamnya terdiri dari 4 *Look Up Table* (LUT) 6-*input* dan 8 *flip-flop (sequential unit).*

4.2. Simulasi Program Aplikasi

4.2.1. Membuat *File Library Board Atlys*

Untuk melakukan proses simulasi, spesifikasi *library board* perlu dibuat terlebih dahulu karena *library* untuk *board* Atlys belum ada. Pada penelitian ini, dibuat dua *file* atlys.S dengan parameter *_board_clk_freq* yang berbeda, yaitu 50MHz (atlys.S) dan 100MHz (atlys2.s).

Setelah *file* atlys.S disimpan, satu persatu *file board* di-*compile* dengan perintah berikut: *or32-elf-gcc –c atlys.S*. Proses selanjutnya adalah mengubahnya menjadi *file library* dengan perintah: *or32-elf-ar–rs libboard.a atlys.o*. Hasilnya adalah *file library, libboard.a*, yang kemudian disimpan dalam program simulasi *toolchain* Or1ksim.

4.2.2. Program Aplikasi

Program aplikasi pada penelitian ini dibuat sebagai program yang dijalankan pada sistem prosesor untuk menjadi program pengujian. Program aplikasi yang dirancang adalah program aplikasi sederhana yang merepresentasikan banyak jenis set instruksi dalam suatu program. Program aplikasi yang disimulasikan dalam penelitian ini, adalah program aplikasi pencacah jumlah bilangan prima dari 1 sampai 2000.

4.2.3. Proses Compile Program Aplikasi

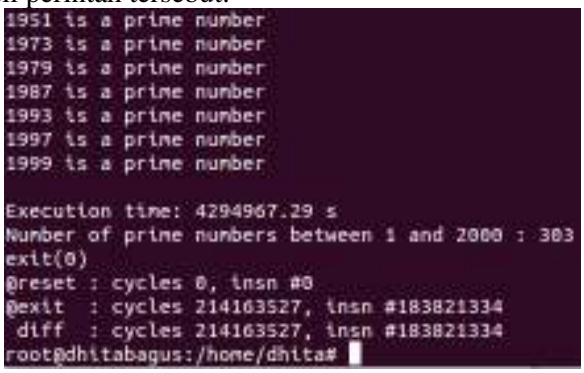
Program aplikasi yang dibuat dengan bahasa C kemudian di-*compile* dengan perintah: *or32-elf-gcc –mnewlib –mboard=atlys PrimeNumbers.c –o PrimeNumber.* Setelah di-*compile, file library board Atlys* perlu diubah dengan *file libboard.a* hasil *compile* dari *file board atlys2.S.* Kemudian *compile* ulang program aplikasi bahasa C namun *output file*nya berbeda, dengan *or32-elf-gcc –mnewlib –mboard=atlys PrimeNumbers.c –o PrimeNumber2.*

Untuk melihat hasil *compile* program aplikasi dalam bahasa Assembly, digunakan perintah: *or32-elf-objdump –d PrimeNumbers.*

4.2.4. Simulasi Program Aplikasi dengan Or1ksim

Untuk melakukan simulasi program aplikasi dengan Or1ksim dapat digunakan perintah: *or32-elf-sim –f $ATLYS_HOME/sim/bin/atlys-or1ksim.cfg PrimeNumbers.*

Berikut adalah hasil perintah tersebut.



Gambar 4 Hasil simulasi program *default* pada or1ksim

4.3. Analisis Hasil Simulasi

4.3.1. Analisis Set Instruksi

OR1200 menerapkan *Instruction Set Architecture* (ISA) sesuai spesifikasi utama arsitektur *OpenRISC* 1000. Tujuan dari analisis ini adalah menunjukkan bahwa ISA sesuai spesifikasi OR 1200 sudah diimplementasikan dengan benar dengan cara melakukan translasi manual dari suatu nilai *opcode* ke dalam instruksi. Dari *tracing opcode* diambil sampel instruksi dan dibandingkan nilainya dengan ISA standar OR1200 sebagai berikut.

Dari hasil *tracing opcode* diambil sampel instruksi dan dibandingkan nilainya dengan ISA standar OR1200. Terlihat adanya perbedaan antara hasil simulasi *tracing* program aplikasi dengan *opcode* asli pada set instruksi. Hal ini terjadi karena adanya proses translasi alamat efektif yang dilakukan oleh IMMU. Untuk menunjukkan bahwa nilai *opcode* dengan ISA sama, perlu dilakukan proses translasi *opcode.*

Setelah proses translasi, maka *opcode* dapat diubah dalam format sesuai ISA sebagai berikut.

Tabel 2 Proses Translasi Opcode

opcode awal	biner	LSB *High*	*Low*	MSB *High*	translasi	*opcode*	*Opcode ISA*
0x18	00011000	00	0110	00	00000110	0x06	0x6
0xa8	10101000	10	1010	00	00101010	0x2a	0x2a
0x44	01000100	01	0001	00	00010001	0x11	0x11
0x15	00010101	*tidak perlu translasi*			00010101	0x15	0x15
0x84	10000100	10	0001	00	00100001	0x21	0x21
0xe0	11100000	11	1000	00	00111000	0x38	0x38
0xe0	11100000	11	1000	00	00111000	0x38	0x38

4.3.2. Analisis Kecepatan Instruksi

Kecepatan instruksi yang diuji disini berdasarkan satuan *Million Instructions Per Second* (MIPS). MIPS sendiri dapat dipakai sebagai pembanding dengan syarat-syarat:

a. Mesin diuji dengan program aplikasi yang sama
b. Set instruksi yang digunakan kedua mesin harus sama

Dengan syarat di atas, parameter yang dibedakan dari dua mesin yang diuji adalah *clock rate*. Mesin pertama memiliki *clock rate* sebesar 50MHz, sedangkan mesin kedua memiliki *clock rate* sebesar 100MHz. Dari hasil pengujian program aplikasi pada gambar 4. didapat parameter perhitungan MIPS sebagai berikut.

Tabel 3 Hasil Simulasi Masing-Masing *Board*

Parameter	Board atlys.S	Board atlys2.S
clock rate	50.000.000 Hz	100.000.000 Hz
waktu eksekusi program	4.294.967,29 ns	4.294.967,29 ns
cycle saat *state reset*	0	0
jumlah instruksi saat *state reset*	0	0
cycle saat *state exit*	214.163.527	243.759.855
jumlah instruksi saat *state exit*	183.821.324	257.812.144
total *cycle dari reset sampai exit*	214.163.527	243.759.855
total instruksi *dari reset sampai exit*	183.821.324	257.812.144

Dengan menggunakan rumus di bawah dapat dicari nilai MIPS untuk masing-masing mesin, sebagai berikut : [6]

$$\text{CPU time} = \text{jumlah cycle} \times \frac{1}{\text{clock rate}} \quad (1)$$

$$\text{Cycles per Instruction (CPI)} = \frac{\text{jumlah cycle}}{\text{jumlah instruksi}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\text{MIPS} = \frac{\text{clock rate}}{\text{CPI} \times 10^6} \quad (3)$$

Sehingga dapat dihasilkan tabel sebagai berikut.

Tabel 4 Nilai MIPS Masing-Masing *Board*

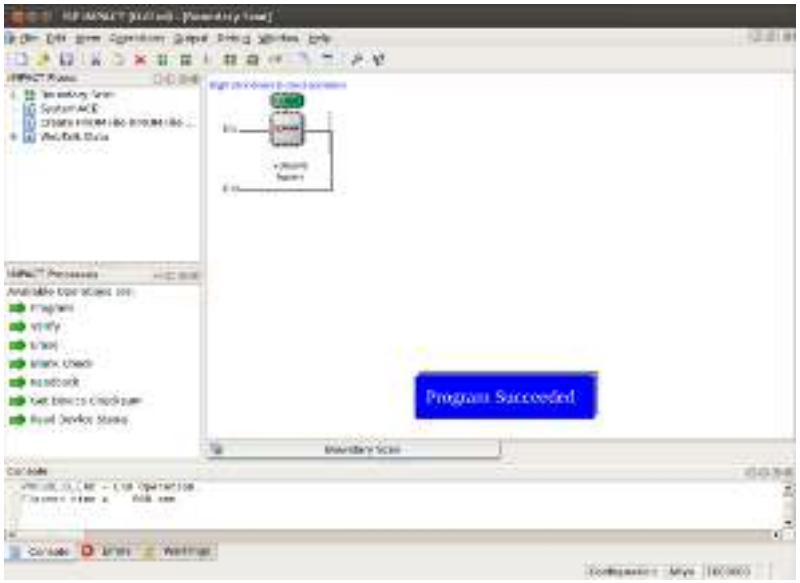
Parameter	Board atlys.S	Board atlys2.S
CPU time	4,28327054	2,43759855
CPI	1,165063501	0,945494076
MIPS	42,916115694	105,764808621

Diketahui bahwa *board atlys.S* memiliki *clock rate* 50Hz, sedangkan *atlys2.S* memiliki *clock rate* 100MHz. Sehingga terlihat jelas pengaruh *clock rate* pada suatu sistem prosesor. Semakin tingggi *clock rate*, maka semakin tinggi pula nilai MIPS.

4.4. Implementasi Sistem pada FPGA

Pemrograman FPGA dapat dilakukan dengan beberapa cara. Salah satunya dengan memprogram *Flash* ROM *built-in* pada *board* FPGA. Dengan cara ini, FPGA akan menjalankan program yang sudah tertanam di dalam *Flash* ROM. Sehingga setiap kali *power-up*, arsitektur prosesor yang dijalankan oleh FPGA adalah *OpenRISC* dengan program aplikasi pengujian yang telah dijelaskan sebelumnya.

Berikut adalah hasil proses pemrograman FPGA dengan aplikasi *impact*.



Gambar 5 Proses Pemrograman Flash FPGA berhasil

Dari gambar di atas dapat dilihat bahwa proses pemrograman sudah sukses dan arsitektur *OpenRISC* dapat dijalankan di dalam FPGA.

4. KESIMPULAN

Kesimpulan yang dapat diambil berdasarkan hasil implementasi yang dilakukan, adalah sebagai berikut:

1. Sistem prosesor yang dirancang dapat diimplementasikan pada FPGA dengan arsitektur sesuai spesifikasi OR1200. Hal ini dapat dilihat dari hasil keluaran serial dari FPGA sama dengan hasil keluaran pada program simulasi *OpenRISC* (or1ksim). Pada simulasi dapat ditunjukkan

bahwa simulasi program dapat berjalan dengan menggunakan arsitektur prosesor *OpenRISC* dengan mengambil *sample opcode* ISA pada program aplikasi pendeteksi bilangan prima sebanyak 7 *opcode* instruksi dari total 52 instruksi di *OpenRISC* sebagai berikut: l.movhi, l.ori, l.jr, l.nop, l.lwz, l.add, l.or.

2. Berdasarkan hasil sintesis *System On Chip prosesor OpenRISC1200 board* FPGA Atlys Spartan-6 LX45 dan menjalankan program aplikasi pencacah bilangan prima dari 1 sampai 2000, didapatkan jumlah *resource* yang dibutuhkan oleh FPGA adalah sebagai berikut.

Resources FPGA	Terpakai	Dari total	Persentase
Slices	4379	6822	64 %
Slices Flip-Flop	6721	54576	12 %
Slices 6 input LUTs	12534	27288	45 %

3. Prosesor dapat memberikan kecepatan instruksi sebesar 42,92 MIPS pada *clock rate* sebesar 50Mhz dan 105,77 MIPS pada *clock rate* sebesar 100MHz.

5. SARAN

Beberapa saran yang dapat yang dapat diberikan untuk pengembangan sistem adalah:

1. Penggunaan *interface* (antar muka) yang lebih banyak, sehingga performansi prosesor *OpenRISC* dapat lebih diketahui.
2. Penanaman *kernel* dan *benchmarking,* sehingga perhitungan terhadap performansi prosesor *OpenRISC* dapat lebih akurat.
3. Penggunaan FPGA *board* dengan seri yang lain yang lebih efisien dalam penggunaannya, serta melakukan proses *clock constraining*, sehingga persentase penggunaan *resource* pada FPGA dapat tepat mencapai 100% (optimal).

DAFTAR PUSTAKA

[1] KC Chang. 2005. *Digital Systems Design with VHDL and Synthesis: An Integrated Approach*. Wiley-IEEE Computer Society, England.

[2] Gray, Jan. 2000. *Designing a Simple FPGA-Optimized RISC CPU and System-on-a-Chip*. Gray Research LLC. Bellevue, WA.

[3] Tong, Jason G., Ian D. L. Anderson and Mohammed A. S. Khalid. 2006. *Soft-Core Processors for Embedded Systems.* Department of Electrical and Computer Engineering University of Windsor. Windsor, Ontario, Canada.

[4] Baxter, Julius. 2001. *Open Source Hardware Development and the OpenRISC Project*. KTH Computer Science and Communication.

[5] Mattsson, Daniel and Marcus Christensson. 2004. *Evaluation of Synthesizable CPU Cores.* Department of Computer Engineering Chalmers University of Engineering. Gothenburg.

[6] Lampret, Damjan. 2007. *OpenRISC 1200 IP Core Specification*. OpenCores.

[7] Brown, Stephen and Zvonko Vranesic. 2005. *Fundamental of Digital Logic with VHDL*. (2nd Ed). San Francisco: Mc Graw Hill.

[8] Rahadi, Hendry Putra. 2013. *Perancangan dan Implementasi Programmable Logic Controller (PLC) Sederhana Berbasis Field Programmable Gate Array (FPGA).* Institut Teknologi Bandung.