MODUL 2 PENGENALAN DESAIN MENGGUNAKAN FPGA

Rifkiansyah Meidian Cahyaatmaja (13511084)

Asisten: Moh. Afandi Mahdiputra

Tanggal Percobaan: 04/10/2012

EL2195-Sistem Digital

Laboratorium Dasar Teknik Elektro - Sekolah Teknik Elektro dan Informatika ITB

Abstrak

FPGA atau Field-programmable Gate Array adalah sejenis Integrated Chip atau IC yang memiliki kemampuan untuk diprogram oleh user menggunakan software tertentu. Pada percobaan ini dilakukan pemrograman FullAdder dan 4-bit Adder pada FPGA dengan pendekatan baik secara skematik maupun menggunakan bahasa VHDL

Kata kunci: FPGA, IC, FullAdder, 4-Bit Adder, Skematik, VHDL.

1. Pendahuluan

FPGA atau Field-Programmable Gate Array adalah IC yang dapat deprogram ulang oleh pengguna dan tidak terikat pada spesifikasi di pabrik. FPGA disukai karena mudah untuk digunakan dan sangat fleksibel dalam penggunaannya..

FULL-ADDER adalah fungsi operasi yang menambahkan seluruh input dan menghasilkan output berupa Carry dan Sum yang diintepretasikan sebagai 1 dan 0 dalam board FPGA. Ada beberapa jenis FULL-ADDER namun tak semuanya dibahas dalam praktikum ini.

VHDL atau VHSIC Hardware Description Language adalah bahasa pemrograman yang digunakan untuk menterjemahkan perintah agar dapat diterima sebagai masukan oleh perangkat keras.

Ada tujuh jenis percobaan dalam modul 2 ini, yaitu Mendesain Full-Adder dengan skematik, Mendesain Full-Adder dengan pendekatan bahasa VHDL, Mendesain 4-bit Ripple Carry Adder dengan VHDL, Mendesain 4-bit Adder dengan skematik, Simulasi sederhana menggunakan ModelSim, Membuat Testbench, dan Melakukan proses tapping sinyal dari sebuah desain.

1. Studi Pustaka

FPGA adalah sebuah devais logika yang dapat deprogram yang menyokong implementasi gerbang logika yang cukup besar. FPGA cukup berbeda dari SPLD dan CPLD dimana FPGA tidak mencakup bidang AND atau OR [1].

VHDL adalah sebuah bahasa deskripsi perangkat keras yang digunakan dalam pengotomatisasian desain elektronik untuk menjelaskan system sinyal digital dan sinyal campuran seperti FPGA dan IC [2].

* 1. Altera Quartus

Adalah piranti lunak yang digunakan untuk membuat rangkaian dan mengevaluasinya baik dalam bentuk simulasi maupun implementasi langsung pada board yang disambungkan.

* 1. Altera Modelsim

Adalah piranti lunak yang digunakan untuk membuat rangkaian dan mensimulasikan sinyal yang dihasilkannya secara digital. Hasil dievaluasi dari sinyal yang terlihat di layar.

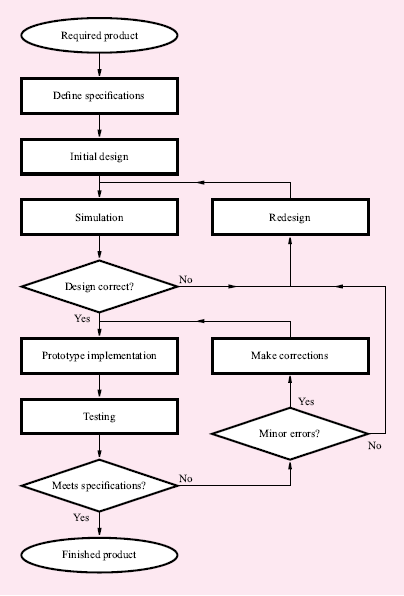
* 1. Altera DE2-70

Merupakan bagian dari perangkat pembelajaran yang digunakan dalam praktikum modul 2. Altera DE2-70 terdiri atas board utama serta kabel penghubungnya.

Ketiga deskripsi diatas diambil dari Datasheet masing-masing.

1. Metodologi

Pada praktikum ini dilakukan percobaan dengan metoda pengembangan desain seperti yang digambarkan pada gambar 3.1 berikut.



Gambar 3‑1 Skema Dasar Konsep Desain

Pada percobaan 2A, dibuat sebuah skema Full-Adder menggunakan program Altera Quartus II 9.0 sp2 Web Edition. Disini digunakan library yang baru dibuat dan menggunakan symbol untuk menampilkan rangkaian yang akan diujicobakan.

Pada percobaan 2B, dibuat sebuah FullCarry-Adder dengan pendekatan bahasa VHDL. Dalam hal ini, VHDL di-input untuk kemudian digunakan sebagai program dalam board.

Pada percobaan 2C, dibuat sebuah skema Ripple-Carry-Adder dengan menggunakan bahasa VHDL. Bahasa VHDL yang sebelumnya di-input kedalam Quartus II diubah dari perintah sebelumnya dengan mengubah kode perintah.

Pada percobaan 2D, dibuat sebuah 4-bit Adder dengan skematik menggunakan sebuah data Full-Adder yang telah dibuat sebelumnya. Hasil dari input diamati dalam bentuk LED.

Pada percobaan 2E, dibuat sebuah simulasi sederhana menggunakan software ModelSIm. Modelsim ini sebenarnya adalah bawaan dari Altera Quartus. Dalam percobaan ini yang dilihat adalah sinyal keluaran input dan hasilnya di output.

Pada percobaan 2F, rangkaian digital disimulasi inputnya satu-persatu. Dibuatlah sebuah file bernama testbench dan Device Under Test untuk menguji dasar dari program tersebut

Pada percobaan 2G, dilakukan tapping sinyal dari desain, yaitu menggunakan hanya satu jenis sinyal yang terlihat agar dapat diketahui dengan lebih mudah bila ada sinyal yang error.

1. Hasil dan Analisis
   1. Mendesain Full Adder Dengan Skematik

Dalam percobaan skematik Altera Quartus ini, hasil diamati dengan indikator lampu LED merah dan hijau yang ada pada board.

Dari pengamatan pada board Altera DE2-70 didapat hasil dibawah ini:

Tabel ‑ Tabel Kebenaran Percobaan 2A

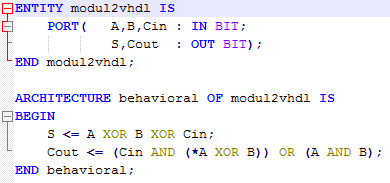
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | Sum | Carry |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Tabel diatas sesuai dengan referensi dari modul. Artinya, percobaan yang dilakukan telah sesuai dengan yang diinginkan. Skematik yang digambarkan telah sesuai.

* 1. Mendesain FULL-ADDER Dengan Pendekatan Bahasa VHDL

Dalam percobaan VHDL Altera Quartus ini, hasil diamati dengan indikator lampu LED.

Kode yang digunakan dalam percobaan ini adalah sebagai berikut:



Gambar 4‑1 Script percobaan 4B

Dari pengamatan pada board Altera DE2-70 didapat tabel seperti berikut:

Tabel 4‑2 Tabel Kebenaran Percobaan 2B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | Sum | Carry |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

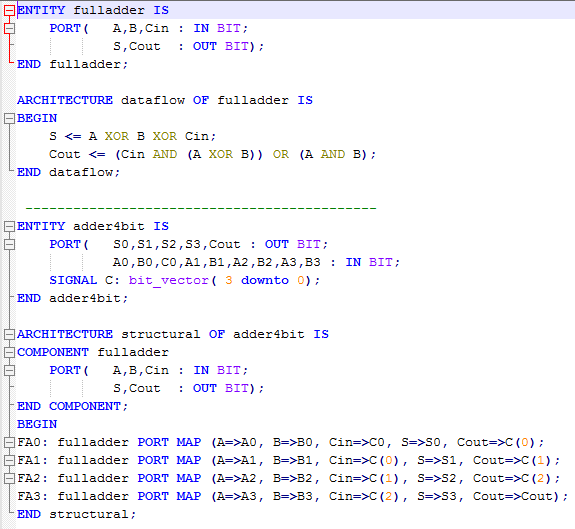
Tabel diatas sesuai dengan referensi dari modul dan sama dengan tabel 4-1. Artinya, penggunaan bahasa VHDL untuk memprogram board sudah benar dan penggunaannya sama dengan skematik dari percobaan sebelumnya.

* 1. Mendesain 4-Bit Ripple Carry Adder dengan VHDL

4-Bit Ripple Carry Adder adalah program untuk menjumlahkan 2 buah integer 4-bit dan satu buah carry in untuk menghasilkan 1 buah sum integer 4-Bit dan 1 buah carry out. Hasil penjumlahan kelima bilangan tersebut disimpan dalam bentuk 5-Bit dimana 4 bit adalah Most Significant Byte,dan 1 bit sisanya adalah Carry Out

4-Bit Carry Adder memanfaatkan 4 buah Full-Adder dengan Carry Out ke-I dihubungkan ke Carry In ke-I+1. Dari penjumlahan yang dilakukan bit A pertama dengan bit B pertama, sisa dari penjumlahan itu masuk kedalam carry untuk digunakan di penjumlahan selanjutnya, yaitu penjumlahan bit A kedua dengan bit B kedua. Hal ini berlanjut hingga penjumlahan bit A keempat dan bit B keempat, dimana carry pada akhirnya akan disimpan dan tidak dimasukkan dalam perhitungan 4-Bit yang kemudian dikonversi kedalam bentuk angka.

Kode yang digunakan sebagai berikut:



Gambar 4‑2 Waveform percobaan 4C

Tabel hasil eksperimen ini adalah sebagai berikut:

Tabel 4‑2 Tabel Kebenaran Percobaan 2C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | 1 | 2 | 3 | 4 |
| A0 | 0 | 0 | 1 | 1 |
| A1 | 0 | 0 | 1 | 0 |
| A2 | 0 | 1 | 1 | 0 |
| A3 | 1 | 1 | 1 | 1 |
| B0 | 0 | 1 | 1 | 1 |
| B1 | 0 | 1 | 1 | 0 |
| B2 | 0 | 0 | 1 | 0 |
| B3 | 1 | 0 | 1 | 1 |
| COut | 1 | 0 | 1 | 1 |
| S0 | 0 | 1 | 0 | 0 |
| S1 | 0 | 1 | 1 | 1 |
| S2 | 0 | 1 | 1 | 0 |
| S3 | 0 | 1 | 1 | 0 |

Tabel diatas dibaca dari atas kebawah, dan keempat Sum maupun Carry Out berfungsi dengan baik sesuai dengan teori yang telah dipelajari sebelumnya.

* 1. Mendesain 4-Bit Carry Adder dengan Skematik

Dalam percobaan ini digunakan skematik Full-Adder yang telah disediakan oleh file pendukung sebelumnya.

Tabel 4‑2 Tabel Kebenaran Percobaan 2D

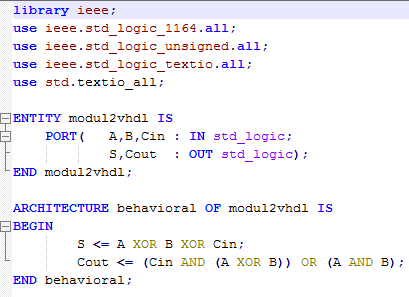
|  |  |  |  |
| --- | --- | --- | --- |
| No. | 1 | 2 | 3 |
| A0 | 1 | 1 | 1 |
| B0 | 0 | 0 | 1 |
| C0 | 0 | 0 | 0 |
| A1 | 1 | 1 | 0 |
| B1 | 1 | 0 | 0 |
| A2 | 0 | 1 | 0 |
| B2 | 1 | 0 | 0 |
| A3 | 0 | 1 | 1 |
| B3 | 0 | 0 | 1 |
| S0 | 1 | 1 | 0 |
| S1 | 0 | 1 | 1 |
| S2 | 0 | 1 | 0 |
| S3 | 1 | 1 | 0 |
| COut | 0 | 0 | 0 |

Tabel diatas memiliki hasil yang sama dengan tabel 4-2 yang dapat dikatakan sesuai dengan teori Full-Adder. Artinya, desain secara skematik juga dapat digunakan dalam pembuatan Full-Adder menggunakan piranti lunak Altera Quartus II, walaupun menggunakan data pendukung yang telah dibuat terlebih dahulu sebelumnya. Hal ini menunjukkan bahwa baik secara VHDL maupun Skematik, keduanya bisa sama-sama mendapat hasil yang sama.

* 1. Simulasi Sederhana Menggunakan ModelSim

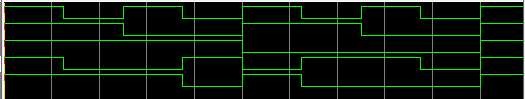
ModelSim, yang juga adalah sebuah piranti lunak dari Altera, menggunakan cara yang lebih sederhana dalam pembacaan hasil analisisnya, yaitu dengan cara membaca Waveform yang dihasilkan dari masukan.

Kode yang digunakan sebagai berikut:



Gambar 4‑3 Script percobaan 4E

Waveform yang dihasilkan dari percobaan ini adalah sebagai berikut:



Gambar 4‑4 Waveform percobaan 4E

Gambar tersebut dari atas ke bawah adalah:

A

B

Cin

S

COut

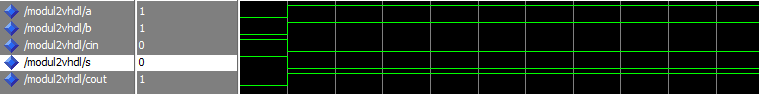
Dengan sinyal tinggi (HIGH) menunjukkan logika 1 dan sinyal rendah (LOW) menunjukkan logika 2, dapat dibuat tabel sebagai berikut:

Tabel 4‑2 Tabel Kebenaran Percobaan 2E

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | CIn | S | COut |
| 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |

Tabel diatas menunjukkan hasil yang sama dengan percobaan 2A dan 2B, yang artinya hasilnya sesuai dengan teori Adder yang telah dipelajari sebelumnya.

Setelah dilakukan pengamatan biasa, kemudian dilakukan modifikasi No Force pada sinyal, yang menghasilkan wave sebagai berikut:



**Gambar 4‑5 Waveform percobaan 4E No Force**

Gambar diatas menunjukkan kalau setelah dilakukan No Force, sinyal tidak naik maupun turun.

Kemudian, dilakukan Force dan hasil yang didapat sama. Artinya, hasil yang didapat dari No Force terbawa ke Force dan tidak berubah.

* 1. Membuat Testbench

Ada cara lain untuk mensimulasikan sistem digital selain memasukkan input satu-persatu, yaitu dengan membuat file VHDL untuk memasukkan input pada rangkaian yang akan diuji (Device Under Test).

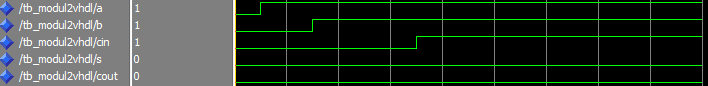
Waveform awal adalah sebagai berikut:



**Gambar 4‑6 Waveform percobaan 4F Initial**

Dalam waveform ini telah diikuti kode yang dituliskan di modul, namun sum dan carry out tidak berjalan sebagaimana mestinya. Hal ini dikarenakan file DUT yaitu modul2vhdl yang tidak dimasukkan terlebih dahulu.

Setelah dimodifikasi, waveform menjadi seperti berikut:



**Gambar 4‑7 Waveform percobaan 4E Post-Modifikasi**

Dapat dilihat perubahan pada input dimana input hanya berubah satu kali dan kemudian tak berubah lagi untuk seterusnya.

* 1. Melakukan Proses Tapping Sinyal dari sebuah Desain

Tapping adalah mengambil nilai sinyal yang sebenarnya bukan output sistem/ rangkaian digital yang akan diuji agar lebih mudah diketahui dimana error terjadi. Dalam hal ini, sinyal yang akan diambil adalah ketiga Input A,B,C, dan jugaTemp.

Waveformnya seperti berikut:



**Gambar 4‑8 Waveform percobaan 4G Tapping**

Seperti dapat dilihat disini, waveform ini memiliki bentuk lurus yang sama dengan 4.6 karena kesalahan pada penggunaan file DUT sebelumnya.

Dari sinyal Temp seharusnya dapat diketahui bentuk yang dimiliki oleh S dan Cout apakah sudah benar atau salah.

1. Kesimpulan

Tabel Kebenaran gerbang logika adalah standar untuk semua jenis gerbang logika baik yang field-programmable maupun yang sudah terprogram dari produsen.

Rangkaian Full Adder dapat dibuat baik dengan pendekatan skematik maupun VHDL. Pada pendekatan skematik, bagaimanapun, diperlukan tambahan skema dari file pendukung untuk mendapat hasil yang diinginkan.

Rangkaian 4-bit Ripple Carry Adder dapat dibuat baik dengan pendekatan skematik maupun VHDL. Rangkaian 4-bit Carry Look-Ahead Adder dapat dibuat dengan pendekatan VHDL.

Modelsim dan Quartus sama-sama dapat digunakan untuk analisa sinyal, walaupun untuk implementasi kedalam FPGA hanya Quartus yang dapat digunakan.

Testbench adalah cara yang sederhana untuk menguji rangkaian karena kita tidak perlu harus selalu memasukkan input kedalam rangkaian, hanya tinggal menggunakan kode yang sudah ada untuk mengetes prototip desain / ARCHITECTURE yang telah dibuat sebelumnya.

Daftar Pustaka

1. Brown, Stephen., *Fundamentals of Digital Logic with VHDL*, hal 105, McGraw-Hill, San Fransisco, 2009.
2. http://en.wikipedia.org/wiki/VHDL , 07 Oktober 2012, 23.15.