MODUL 6
IMPLEMENTASI RANGKAIAN DIGITAL PENGANCING, FLIP-FLOP DAN REGISTER DI FPGA MENGGUNAKAN VERILOG-HDL

1. Tujuan Praktikum Modul 6

Setelah mempraktekkan Topik ini, mahasiswa diharapkan dapat :

1. Mengimplementasikan Rangkaian Digital Pengancing di FPGA DE10-Lite
2. Mengimplementasikan Rangkaian Digital Flip-Flop di FPGA DE10-Lite
3. Mengimplementasikan Rangkaian Digital Register di papan FPGA DE10-Lite

2. Alat dan Bahan

a. Laptop yang telah terinstal *software Quartus 18*

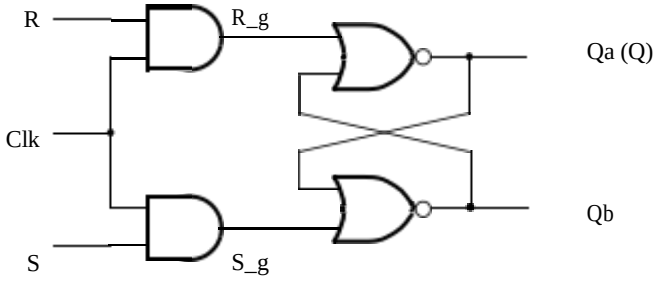
b. Papan Pengembangan FPGA DE10-Lite

3. Dasar Teori Praktikum Modul 6

3.1 Rangkaian Digital Pengancing (*Latches*)

Intel® FPGA memiliki flip-flop yang disediakan untuk mengimplementasikan suatu rangkaian pengguna. Kita akan tunjukkan bagaimana kegunaan flip-flop ini di Percobaan IV. Namun pertama-tama kita akan tunjukkan bagaimana elemen penyimpanan dapat dibuat pada FPGA tanpa menggunakan flip-flop khusus.

Gambar 1 menunjukkan sebuah rangkaian pengancing RS dengan gerbang. Dua gaya kode Verilog yang dapat digunakan untuk menggambarkan rangkaian ini diberikan pada Gambar 2 dan 3. Gambar 2 menunjukkan pengancing yang diberi masukan dari gerbang logika, dan Gambar 3 menggunakan pernyataan logika untuk membuat rangkaian yang sama. Jika pengancing ini diimplementasikan di sebuah FPGA yang memiliki 4-masukan tabel *lookup* (LUT), maka hanya satu tabel *lookup* saja yang diperlukan, seperti ditunjukkan pada Gambar 4a.



Gambar 1: Rangkaian pengancing RS dengan masukan gerbang.

```
// A gated RS latch
module part1 (Clk, R, S, Q);
input Clk, R, S;
output Q;
```

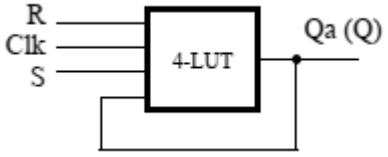
```
wire R_g, S_g, Qa, Qb /* synthesis keep */ ;

and (R_g, R, Clk);
and (S_g, S, Clk);
nor (Qa, R_g, Qb);
nor (Qb, S_g, Qa);

assign Q = Qa;

endmodule
```

Gambar 2: Rangkaian pengancing RS dengan pernyataan gerbang logika.

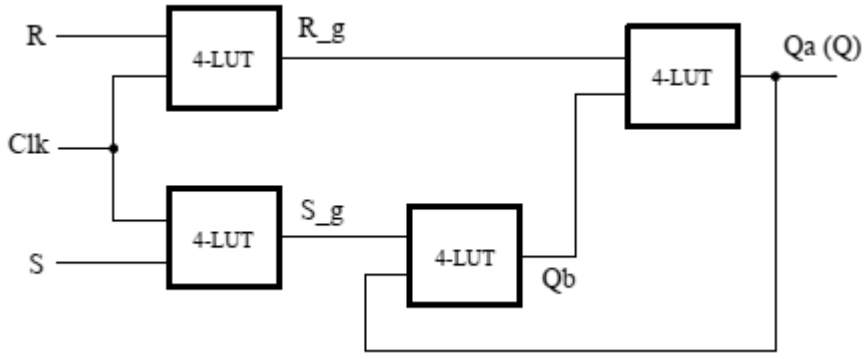
```
// A gated RS latch
module part1 (Clk, R, S, Q);
input Clk, R, S;
output Q;

wire R_g, S_g, Qa, Qb /* synthesis keep */ ;

assign R_g = R & Clk;
assign S_g = S & Clk;
assign Qa = ~(R_g | Qb);
assign Qb = ~(S_g | Qa);
assign Q = Qa;

endmodul
```

Gambar 3: Rangkaian Pengancing RS dengan menggunakan ekspresi Boolean.

Walaupun rangkaian pengancing dapat direalisasikan menggunakan satu LUT 4-masukan, implementasikan tidak memungkinkan penggunaan sinyal internalnya, seperti R_g and S_g, agar dapat diamati, karena tidak disediakan sebagai keluaran dari LUT. to be observed, because they are not provided as outputs from the LUT. Agar sinyal internal ini dapat digunakan pada rangkaian yang diimplementasikan, perlu disertakan suatu pengarah kompilasi dalam kodenya. Pada Gambar 2 dan 3 pengarah */* synthesis keep */* disertakan untuk memerintahkan kompiler Quartus® untuk menggunakan elemen logika terpisah bagi setiap sinyal R_g, S_g, Qa, dan Qb. Sehingga dengan mengkompilasi kode tersebut dihasilkan rangkaian dengan empat buah LUT 4-masukan yang ditunjukkan pada Gambar 4b.

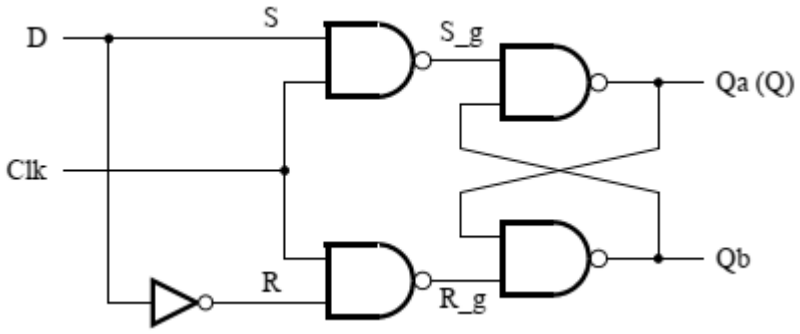


(a) Pengancing RS yng menggunakan sebuah tabel lookup 4-masukan.



(b) Pengancing RS yang menggunakan empat buah table lookup 4-masukan

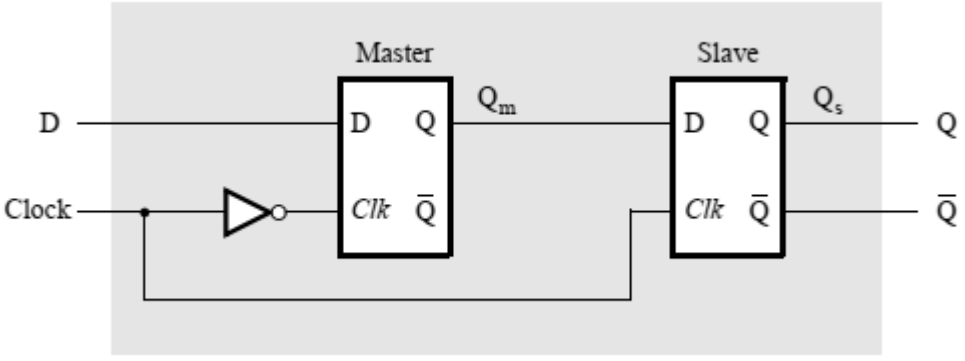
Gambar 4: Implementasi pengancing RS dari Gambar 1.



Gambar 6: Rangkaian pengancing D dengan masukan gerbang

3.2 Rangkaian Digital Flip-Flop

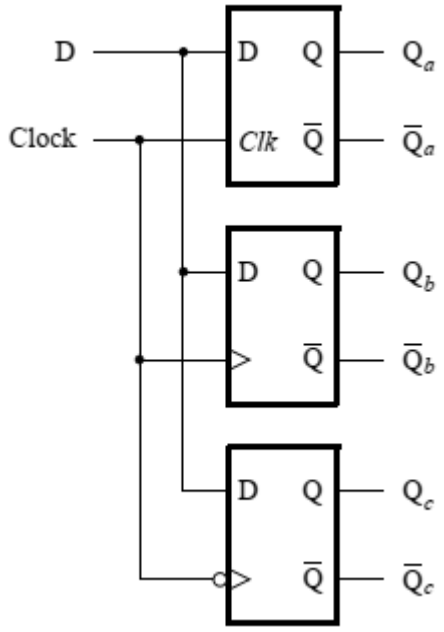
Gambar 7 menunjukkan rangkaian *master-slave D flip-flop.*



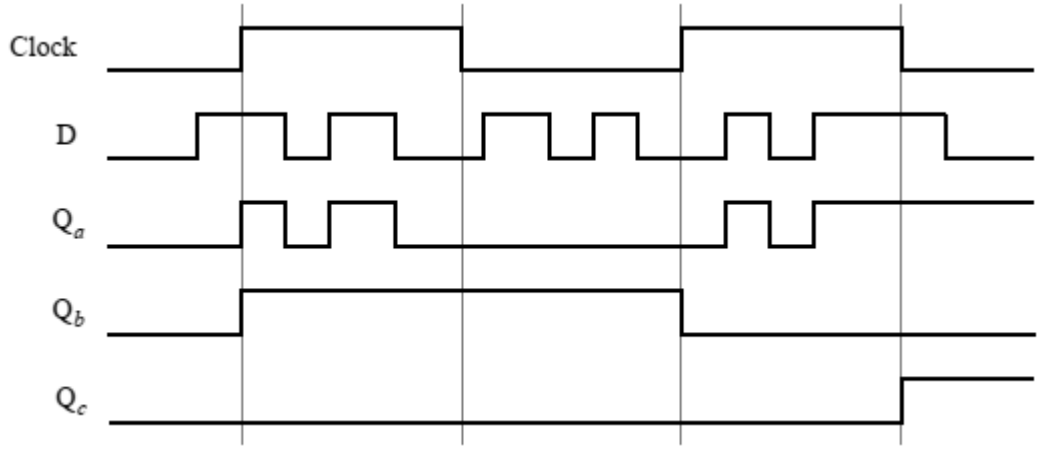
Gambar 7 Rangkaian Master-Slave D Flip-flop

3.3 Rangkaian Digital Register

Gambar 8 menunjukkan sebuah rangkaian dengan tiga elemen penyimpan yang berbeda yaitu: pengancing D dengan gerbang, D flip-flop yang dipicu tepi positif, dan D flip-flop yang dipicu tepi negatif.



(a) rangkaian



(b) Diagram pewaktuan

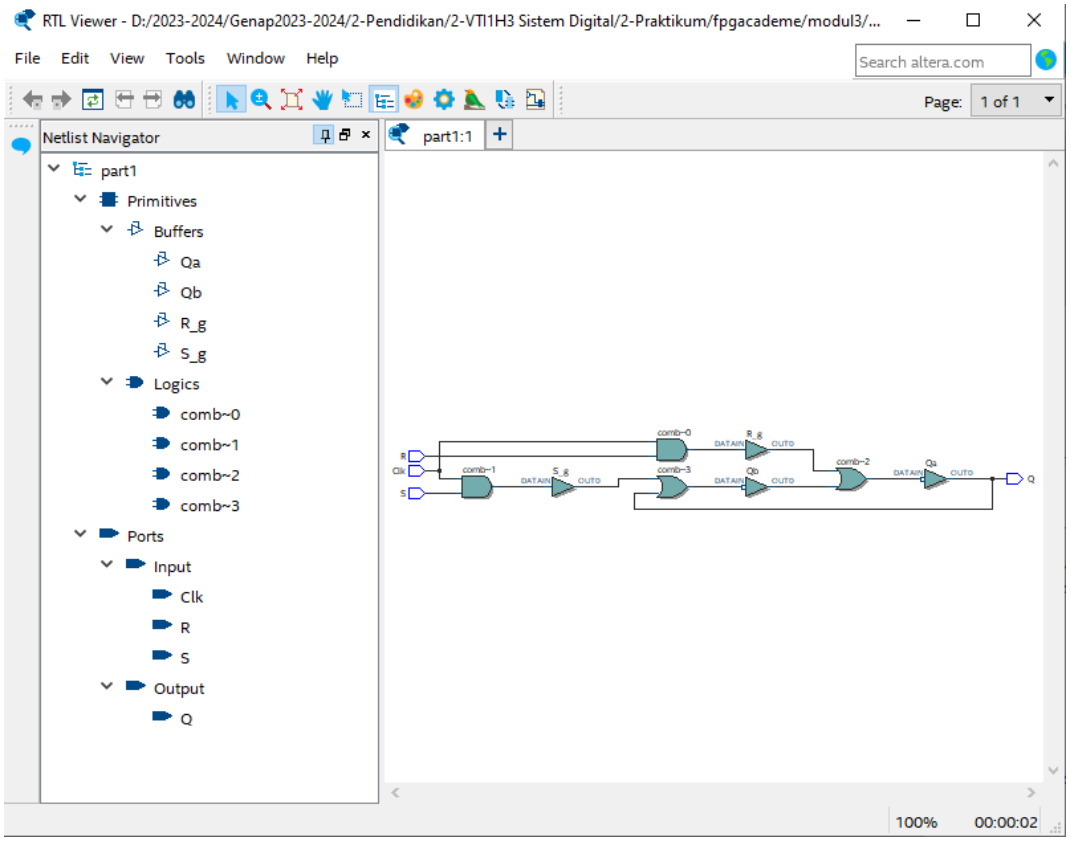
Gambar 8: Rangkaian dan bentuk gelombang untuk Percobaan IV.

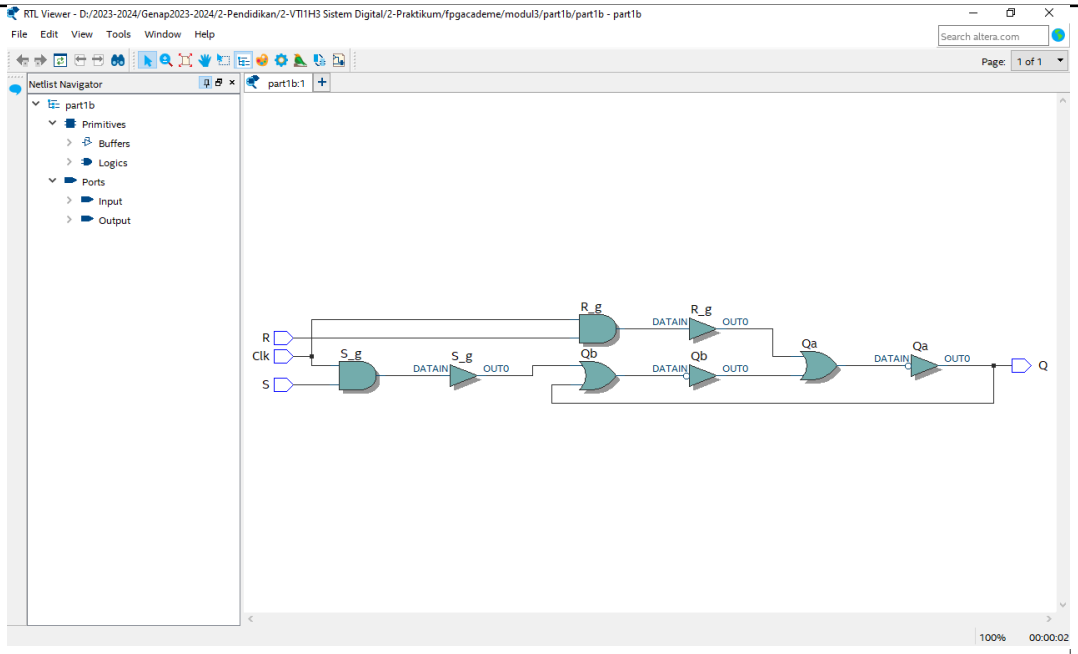
4. Lembar Kegiatan Praktikum Modul 4 :

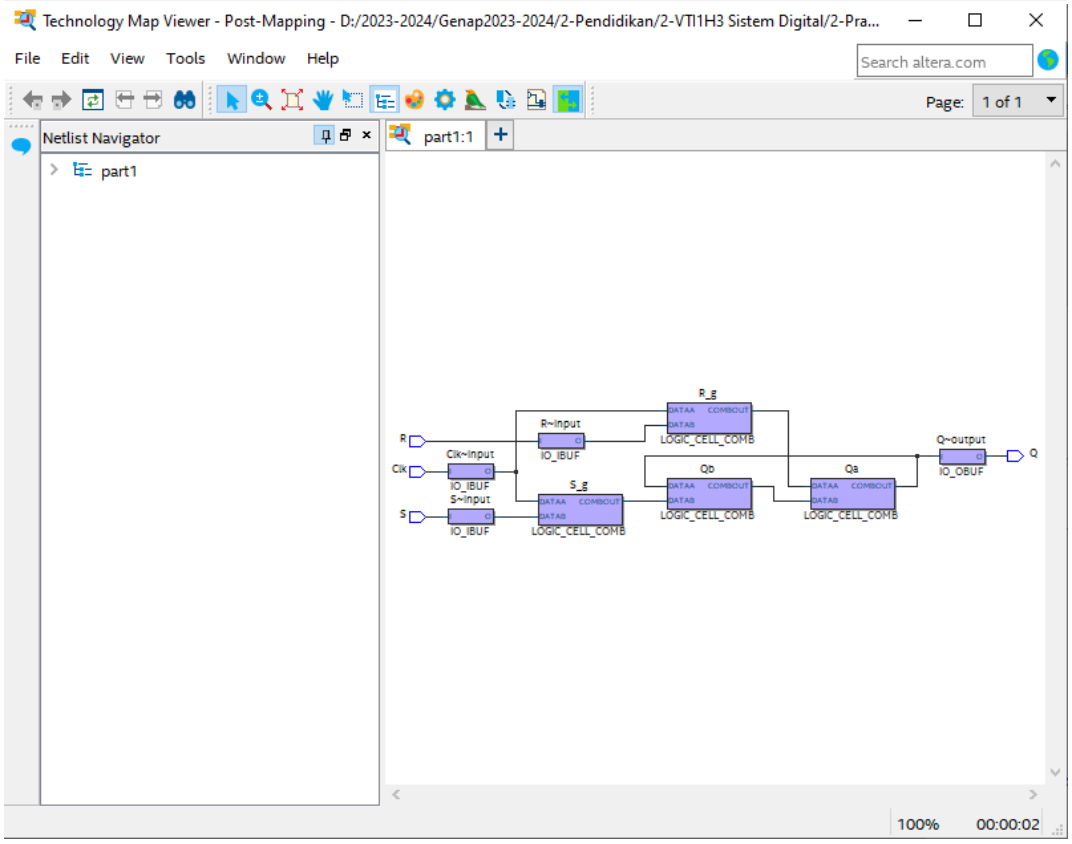
4.1 Percobaan I

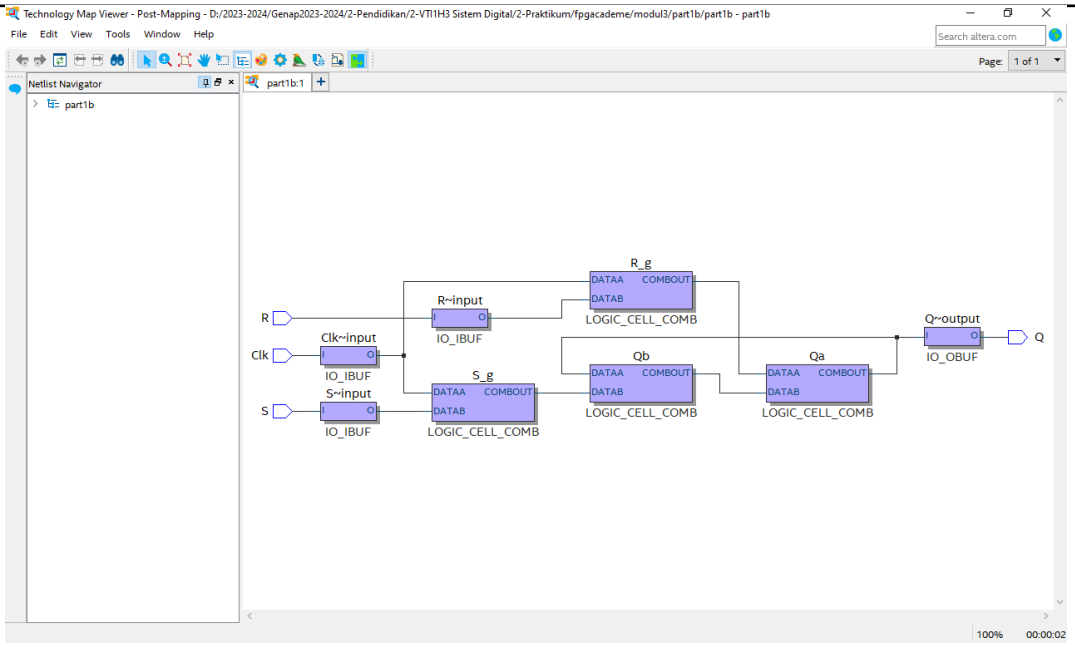
Buat proyek Quartus untuk sirkuit kait RS sebagai berikut:

1. Buat proyek Quartus baru untuk papan DE-series Anda.
2. Buat file Verilog untuk rangkaian pengancing RS. Gunakan kode pada Gambar 2 atau Gambar 3 (kedua versi kode harus menghasilkan sirkuit yang sama) dan sertakan dalam proyek.
3. Kompilasi kode yang ditulis. Gunakan alat Quartus RTL Viewer untuk memeriksa rangkaian tingkat gerbang (*gate-level circuit*) yang dihasilkan dari kode, dan gunakan alat Penampil Peta Teknologi (*Technology Map Viewer*) untuk memverifikasi bahwa rangkaian pengancing diimplementasikan seperti yang ditunjukkan pada Gambar 4b.



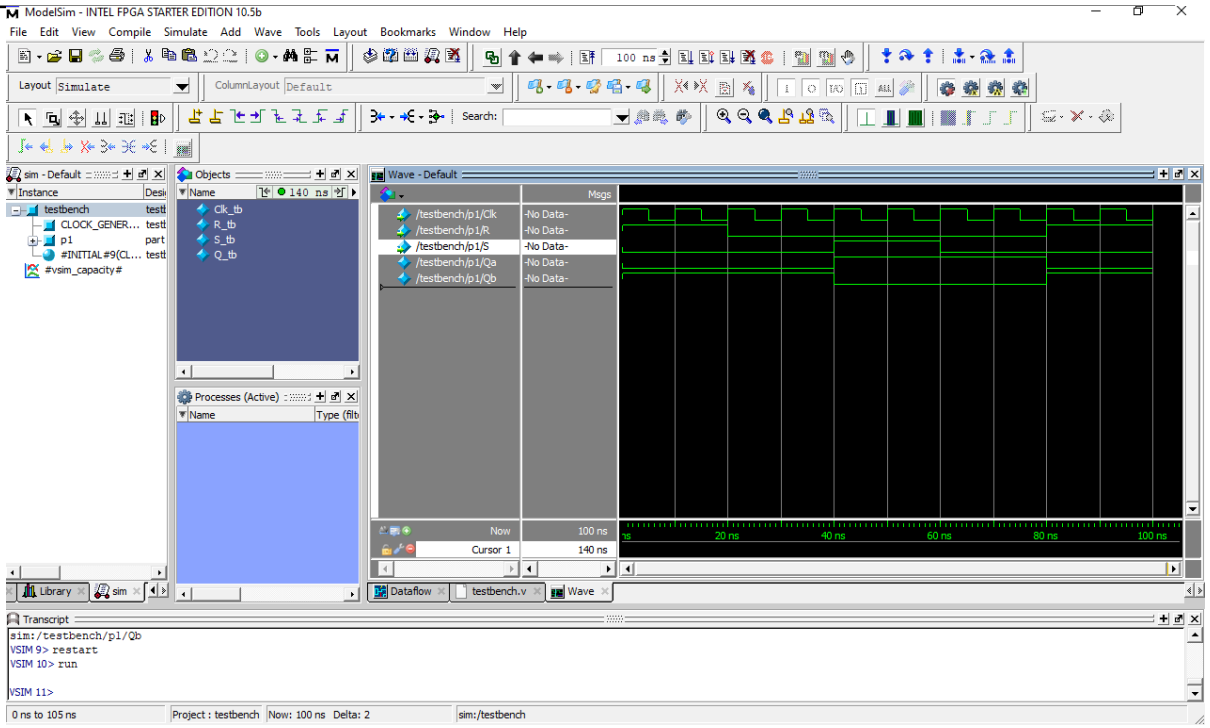

RTL Viewer - D:/2023-2024/Genap2023-2024/2-Pendidikan/2-VTI1H3 Sistem Digital/2-Praktikum/fpgacademe/modul3/part1b/part1b - part1b
File Edit View Tools Window Help
Search altera.com
Page: 1 of 1
Netlist Navigator
part1b
Primitives
Buffers
Logics
Ports
Input
Output
part1b:1
R
Clk
S
S_g
DATAIN
OUT0
R_g
Qb
Qa
Q
100%
00:00:02

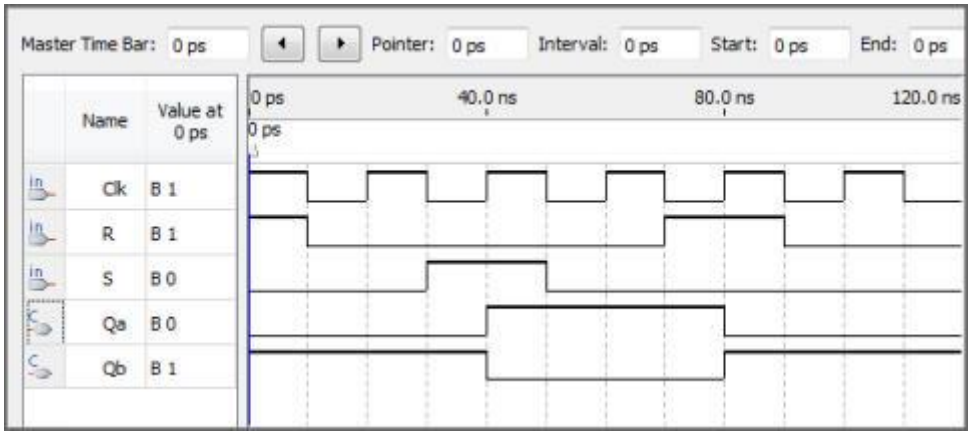

Technology Map Viewer - Post-Mapping - D:/2023-2024/Genap2023-2024/2-Pendidikan/2-VTI1H3 Sistem Digital/2-Pra...
File Edit View Tools Window Help
Search altera.com
Page: 1 of 1
Netlist Navigator
part1
part1:1
R
Clk
S
R~input
Clk~input
S~input
IO_IBUF
R_g
S_g
Qb
Qa
LOGIC_CELL_COMB
Q~output
IO_OBUF
Q
100%
00:00:02



4. Simulasikan perilaku kode Verilog Anda dengan menggunakan fitur simulasi yang disediakan di perangkat lunak Modelsim. Gunakan testbench yang disediakan untuk menggerakkan sinyal untuk simulasi Anda. Contoh file bentuk gelombang vektor ditampilkan pada Gambar 5. Bentuk gelombang pada gambar dimulai dengan menetapkan *Clk = 1* dan R = 1, yang memungkinkan alat simulasi untuk menginisialisasi semua sinyal di dalam pengancing ke nilai yang diketahui.

```
`timescale 1ns / 1ns

module testbench ();
	reg Clk_tb;
	reg R_tb;
	reg S_tb;
	wire Q_tb;

	initial
		begin: CLOCK_GENERATOR
			Clk_tb = 1;
			forever
				begin
					#5 Clk_tb = ~Clk_tb;
				end
		end

	initial
		begin
			R_tb <= 1; S_tb <= 0;
			#20 R_tb <= 0;
			#20 S_tb <= 1;
			#20 S_tb <= 0;
			#20 R_tb <= 1;
		end

	part1 p1 (Clk_tb, R_tb, S_tb, Q_tb);
endmodule
```





Gambar 5: Bentuk gelombang simulasi untuk pengancing RS.

5. Buatlah kesimpulan dari percobaan ini

4.2 Percobaan II

Lakukan langkah-langkah berikut:

1. Buatlah proyek Quartus baru. Buat sebuah file Verilog menggunakan gaya kode pada Gambar 3 untuk pengancing D dengan masukan gerbang. Gunakan pengarah /* *synthesis keep* */ untuk memastikan elemen logika terpisah digunakan untuk mengimplementasikan sinyal-sinyal R, S_g, R_g, Qa, dan Qb.
2. Kompilasi proyek anda dan kemudian gunakan *Technology Map Viewer* untuk memeriksa rangkaian yang diimplementasikan.
3. Verifikasi bahwa rangkaian pengancing telah bekerja sesuai seluruh kondisi masukan menggunakan simulasi fungsional. Periksa karakteristik pewatuan rangkaian menggunakan simulasi pewaktuan,
4. Buatlah sebuah proyek Quartus baru yang akan digunakan untuk implementasi pengancing

D dengan gerbang di papan DE10-lite. Proyek ini harus terdiri dari sebuah *top-level module* yang berisi port-port (pin) masukan dan keluaran yang sesuai papan DE10-Lite. Ujilah keluaran rangkaian pengancing anda. Gunakan saklar SW0 untuk menggerakkan masukan pengancing D, dan gunakan SW1 sebagai masukan Clk. Hubungkan keluaran Q ke LEDR0.

5. Sertakan *pin assignments* yang diperlukan dan kemudian kompilasi proyek anda serta unduh rangkaian yang telah dikompilasi ke papan DE10-Lite anda.
6. Ujilah fungsionalitas rangkaian anda dengan cara mengubah saklar D dan Clk serta catat perubahan keluaran Q berdasarkan perubahan masukannya.
7. Buatlah kesimpulan dari percobaan anda

4.3 Percobaan III

Lakukan langkah-langkah berikut:

1. Buatlah proyek Quartus baru. Buat sebuah file Verilog yang menggambarkan dua buah pengancing D yang anda buat di percobaan II sebelumnya untuk mengimplementasikan *master-slave flip-flop*.
2. Sertakan pada proyek anda port masukan dan keluaran yang sesuai dengan papan DE10-Lite. Gunakan saklar SW0 untuk menggerakkan masukan D flip-flop, dan gunakan SW1 sebagai masukan Clock. Hubungkan keluaran Q ke LEDR0.
3. Sertakan *pin assignments* yang diperlukan dan kompilasi proyek anda.
4. Gunakan *Technology Map Viewer* untuk memeriksa rangkaian D flip-flop, dan gunakan simulasi untuk memverifikasi operasi yang benar.
5. Unduh rangkaian tersebut ke papan DE10-Lite dan uji fungsionalitasnya dengan mengubah-ubah saklar D dan Clock serta catat perubahan keluaran Q berdasarkan perubahan masukannya.
6. Buatlah kesimpulan dari percobaan anda

4.4 Percobaan IV

Implementasikan dan simulasikan rangkaian ini menggunakan software Quartus seperti berikut:

1. Buatlah proyek Quartus baru.
2. Tulislah sebuah file Verilog yang akan menjalankan tiga buah elemen penyimpan. Untuk bagian ini anda tidak perlu lagi menggunakan pengarah */* synthesis keep */* dari Percobaan I hingga III. Gambar 9 memberikan gaya perilaku kode Verilog yang menentukan pengancing D dengan gerbang di Gambar 6. Pengancing ini dapat diimplementasikan dalam satu tabel *lookup* 4-masukan. Gunakan gaya kode yang sama untuk menentukan flip-flop pada Gambar 8.
3. Kompilasi kode anda dan gunakan *Technology Map Viewer* untuk memeriksa rangkaian yang diimplementasikan. Verifikasi bahwa pengancing menggunakan satu tabel *lookup* dan bahwa flip-flop yang diimplementasikan menggunakan fli-flop yang disediakan oleh papan FPGA DE10-Lite.
4. Gunakan Modelsim untuk mensimulasikan rangkaian yang anda buat. Gunakan file *testbench* yang disertakan berikut ini untuk menentukan masukan D dan Clock seperti ditunjukkan pada Gambar 8.

```
`timescale 1ns / 1ns

module testbench ();
    reg Clk_tb;
    reg D_tb;
    wire Qa_tb;
```

```
        wire Qb_tb;
        wire Qc_tb;

        initial
                begin: CLOCK_GENERATOR
                        Clk_tb = 0;
                        forever
                                begin
                                        #30 Clk_tb = ~Clk_tb;
                                end
                end

        initial
                begin
                        D_tb <= 0;
                        #20 D_tb <= 1;
                        #20 D_tb <= 0;
                        #5 D_tb <= 1;
                        #10 D_tb <= 0;
                        #10 D_tb <= 1;
                        #10 D_tb <= 0;
                        #5 D_tb <= 1;
                        #5 D_tb <= 0;
                        #10 D_tb <= 1;
                        #5 D_tb <= 0;
                        #5 D_tb <= 1;
                        #20 D_tb <= 0;
                end

        part4 p4 (Clk_tb, D_tb, Qa_tb, Qb_tb, Qc_tb);
endmodule
```

5. Pastikan *testbench* memberikan masukan pada modul yang dibuat dengan benar dan jalankan simulasi untuk mengamati perilaku ketiga elemen penyimpan.

```
module D_latch (D, Clk, Q);
  input D, Clk;
  output reg Q;

  always @ (D, Clk)
   if (Clk)
      Q = D;
endmodule
```

Gambar 9: Sebuah kode Verilog gaya perilaku yang menentukan pengancing D dengan gerbang.

4.5 Percobaan V

Selanjutnya kita ingin menampilkan nilai heksadesimal sebuah bilangan 8-bit A pada dua buah tampilan 7-segmen HEX3 − 2. Kita juga ingin menampilkan nila hex dari sebuah

bilangan 8-bit B pada dua tampilan 7-segment HEX1 − 0. Nilai A dan B merupakan masukan ke rangkaian yang disediakan oleh saklar SW7−0. Untuk memasukkan nilai A dan B, pertama atur saklar ke nilai A yang diinginkan, simpan nilai saklar ini dalam sebuah register, dan kemudian ubah saklar ke nilai B yang diinginkan. Akhirnya, gunakan sebuah penjumlah untuk membuat operasi penjumlahan sum S = A + B, dan tampilkan hasil penjumlahannya di tampilan 7-segmen HEX5 − 4. Tunjukkan simpanan yang dihasilkan proses penjumlahan di LEDR[0].

1. Buat proyek Quartus baru yang akan digunakan untuk mengimplementasikan rangkaian yang diinginkan di papan DE10-Lite
2. Tulis sebuah file Verilog yang menyediakan fungsionalitas yang diperlukan. Gunakan KEY0 sebagai tobol reset *active-low asynchronous* , dan KEY1 sebagai masukan clock.
3. Sertakan *pin assignments* yang diperlukan untuk saklar tekan dan tampilan 7-segmen, kemudian kompilasi rangkaiannya.
4. Unduh rangkaian ke papan DE10-Lite dan uji fungsionalitasnya dengan menekan saklar dan mengamati keluaran tampilan.
5. Buatlah kesimpulan dari percobaan tersebut.

Tabel *pin assignment* yang digunakan pada modul 6

Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTL

Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR0	PIN_A8	LED [0]	3.3-V LVTTL

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY0	PIN_B8	Push-button[0]	3.3 V SCHMITT TRIGGER"
KEY1	PIN_A7	Push-button[1]	3.3 V SCHMITT TRIGGER"

Signal Name	FPGA Pin No.	Description	I/O Standard
HEX00	PIN_C14	Seven Segment Digit 0[0]	3.3-V LVTTL
HEX01	PIN_E15	Seven Segment Digit 0[1]	3.3-V LVTTL
HEX02	PIN_C15	Seven Segment Digit 0[2]	3.3-V LVTTL
HEX03	PIN_C16	Seven Segment Digit 0[3]	3.3-V LVTTL
HEX04	PIN_E16	Seven Segment Digit 0[4]	3.3-V LVTTL
HEX05	PIN_D17	Seven Segment Digit 0[5]	3.3-V LVTTL

Selamat menikmati praktikum Modul 6!!!