Part B Programs

4 bit adder

- Open terminal in the given folder where script.tcl, constraints.sdc are present.

```
csh
source /home/install/cshrc
```

- To open NCLaunch, run the following command:

```
nc launch
```

- While set design directory, select the folder where all the files are present add cds file.
- Now, make library of project do not include IEEE library (explicitly for four bit adder program)
- Type out all the relivant full_adder, four_bit_adder, and adder_test_bench files using gedit.
- full_adder.v

```
module full_adder(A,B,C_in,Sum,C_out);
input A,B,C_in;
output Sum,C_out;
assign Sum = A ^ B ^ C_in;
assign C_out = (A & B) | (C_in & (A ^ B));
endmodule
```

- four_bit_adder.v

```
module four_bit_adder(A,B,C0,S,C4);
input[3:0]A;
input[3:0]B;
input C0;
output[3:0]S;
output C4;
wire C1,C2,C3;
full_adder fa0 (A[0],B[0],C0,S[0],C1);
full_adder fa1 (A[1],B[1],C1,S[1],C2);
full_adder fa2 (A[2],B[2],C2,S[2],C3);
full_adder fa3 (A[3],B[3],C3,S[3],C4);
endmodule
```

- four_bit_adder_test.v

```
module adder_test;
reg[3:0]A;
reg[3:0]B;
reg C0;
wire[3:0]S;
wire C4;
four_bit_adder dut(A,B,C0,S,C4);
initial
begin
A=4'b0011;B=4'b0011;C0=1'b0;
#10 A=4'b1011;B=4'b0111;C0=1'b1;
#10 A=4'b1111;B=4'b1111;C0=1'b1;
end
endmodule
```

- Compile (2nd Icon) all the files in the left window
- Elaborate (3rd Icon) all files in worklib in right window
- Launch Simulator (4th Icon) for the sanpshot/test module in right window
- Right click on the adder in left panel and Send to target > Waveform window
- Assure (constraints.sdc and script.tcl) presence in the working directory.
- Run the command in Cadance Tools :
- To open cadance tools, run the following command:

```
csh
source /home/install/cshrc
```

- Furthur run:
  `genus -f script.tcl`

D Flip Flop

- Open terminal and type out the following command to open Cadance tools:

```
csh
source /home/install/cshrc
```

- To open NCLaunch run the following command:

```
nc launch
```

- Now, make library of project and include IEEE library

- Type out all the relivant verilog code dff.v and dff_test.v using gedit.
- dff.v

```
module dff(clk,d,reset,q,qbar);
input clk,d,reset;
output reg q;
output qbar;
always @(posedge clk)
begin
if(reset==1'b0)
q<=1'b0;
else
q<=d;
end
assign qbar=~q;
endmodule
```

- dff_test.v

```
module dff_test;
reg clk,d,reset;
wire q, qbar;
dff dut(.clk(clk),.d(d),.reset(reset),.q(q),.qbar(qbar));
initial
begin
clk=0;
forever #10clk=~clk;
end
initial
begin
reset=1;
d<=1'b0;
#10 reset=1;
#50 d<=1'b1;
#50 d<=1'b0;
#50 d<=1'b1;
end
initial
#100 $finish;
endmodule
```

4-bit Counter

- counter.v

```
`timescale 1ns/1ps
```

```
module counter(clk,m,rst,count);

input clk,m,rst;
output reg[3:0] count;

always@(posedge clk or negedge rst)
begin
if(!rst)
count=0;
else if(m)
count=count+1;
else
count=count-1;
end
endmodule
```

- counter_test.v

```
`timescale 1ns/1ps
module counter_test;
reg clk,rst,m;
wire[3:0] count;
initial
begin
clk=0;
rst=0;#100;
rst=1;
end
initial
begin
m=0;
#600 m=1;
#500 m=0;
end
counter counter1(clk,m,rst,count);
always #5 clk=~clk;
initial $monitor("Time=%t rst=%b clk=%b count=%b",$time,rst,clk,count);
initial
#1400 $finish;
endmodule
```

- counter_32.v

```
`timescale 1ns/1ps
module counter_32(clk,m,rst,count);
input clk,m,rst;
output reg[31:0] count;
always@(posedge clk or negedge rst)
begin
```

```
if(!rst)
count=0;
else if(m)
count=count+1;
else
count=count-1;
end
endmodule
```

- counter_32_test.v

```
`timescale 1ns/1ps
module counter_32_test;
reg clk,rst,m;
wire[31:0] count;
initial
begin
clk=0;
rst=0;#100;
rst=1;
end
initial
begin
m=0;
#600 m=1;
#500 m=0;
end
counter_32 counter1(clk,m,rst,count);
always #5 clk=~clk;
initial $monitor("Time=%t rst=%b clk=%b count=%b",$time,rst,clk,count);
initial
#1400 $finish;
endmodule
```

ALU

- alu.v

```
module alu_32bit_if(y,a,b,f);
input [31:0]a,b;
input [2:0]f;
output reg[31:0]y;
always@(*)
begin

if(f==3'b000)
y=a&b;

else if(f==3'b001)
```

```
y=a|b;
else if(f==3'b010)
y=a^b;
else if(f==3'b011)
y=~(a&b);
else if(f==3'b100)
y=a+b;
else if(f==3'b101)
y=a-b;
else if(f==3'b110)
y=a*b;
else if(f==3'b111)
y=a/b;
else
y=32'bx;
end
endmodule
```

- alu_test.v

```
module alu_test;
reg[31:0]a;
reg[31:0]b;
reg[2:0]f;
wire[31:0]y;
alu_32bit_if g1(y,a,b,f);
initial
begin
a=32'b00000000;
b=32'b01010101;
f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b011;
#10 f=3'b100;
#10 f=3'b101;
#10 f=3'b110;
#10 f=3'b111;
#10 f=3'bxxx;
end
initial
begin
#100 $finish;
end
endmodule
```