CME341 Assignment 5

1. Intel provides extensive documentation about Quartus and its FPGAs, both online and through the “Help” menu within Quartus. To give you practice in using these resources, please try to find the answers to the following questions. *Hint: You may start from the “Help Topics” entry in the Help menu or else just conduct an online search using your favorite search engine.*

   (a) Does the Intel primitive called DFFE have a “Q_BAR” output (search for DFFE primitive)?

   (b) Does the DFF primitive have asynchronous “set” and “clear” inputs? If so are they active high or active low?

   (c) Define the Intel TFF primitive.

   (d) List the primitives that are not part of the verilog standard, but are available in Intel Quartus II (search for Design Compiler Technology Libraries). Intel distinguishes their primitives from standard verilog primitives by capitalizing the names.

   (e) A latch has two inputs and one output. In class one of the inputs is referred to as the “gate”. What is the name used for the gate input in the Intel primitive called LATCH?

2. Draw schematic diagrams for each of the verilog HDL descriptions given below.

   (a) Circuit 1:

```
module circuit_1 (clk, data_in, data_out);
input clk;
input [1:0] data_in;
output [2:0] data_out;
reg [2:0]  data_out;

always @ (posedge clk)
begin
   data_out[0] = data_in[0]^data_out[0]^data_out[1];
   data_out[1] = data_out[0] & data_out[2];
   data_out[2] = data_out[1] | data_in[1];
end
endmodule
```

(b) Circuit 2:

```
module circuit_2 (clk, data_in, data_out);
input clk;
input [1:0] data_in;
output [1:0] data_out;
reg [1:0]  data_out;

always @ (posedge clk)
begin
   case (data_in[1])
1'b0 : data_out = data_in;
1'b1 : data_out = data_out;
   endcase
end
endmodule
```

(c) Circuit 3:

```
module circuit_3 (clk, data_in, data_out);
input clk;
input [1:0] data_in;
output [1:0] data_out;
reg [1:0]  data_out;

reg [1:0] internal_signal;

always @ (posedge clk)
begin
  data_out[0] = internal_signal[1];
  data_out[1] = data_out[0] | internal_signal[0];
end
```

```
always @ (posedge clk)
begin
  internal_signal[0] = data_in[0] & data_out[0];
  internal_signal[1] = ~(data_in[1] & data_out[0]);
end
endmodule
```

3. Generate another set of schematics for the circuits that result from changing all the blocking assignments (i.e. =) to nonblocking assignments (i.e. <=) in the Verilog HDLs above.

4. Consider the Verilog HDL given below.

```
module circuit_4 (clk, data_in, data_out);
input clk;
input [1:0] data_in;
output [2:0] data_out;

wire [3:0] internal_signal;

assign internal_signal[0] = & data_in;
assign internal_signal[1] = internal_signal[0] ^ data_out[2];
assign internal_signal[2] = internal_signal[1] ^ data_out[0];
assign internal_signal[3] = internal_signal[2] ^ data_out[1];

/* instantiate d flip/flops */
dff dff_1(.clk(clk),.d(internal_signal[1]),.q(data_out[0]));
dff dff_2(.clk(clk),.d(internal_signal[2]),.q(data_out[1]));
dff dff_3(.clk(clk),.d(internal_signal[3]),.q(data_out[2]));

endmodule
```

   (a) Draw a schematic for circuit described above.

   (b) Provide a Verilog behavioral description of the circuit above. In your description use only blocking assignments, i.e. =.

(c) Provide a second Verilog behavioral description of the circuit above, but this time use only nonblocking assignments, i.e. <= in your description.

5. Verify your answers to Question 4 by simulating both the behavioral and structural descriptions and comparing the results.