2010. 4. 10
디지털 시스템 설계 및 실험 출제: 채수익
중간고사 1

1. (Design of a 2-to-1 mux and a 8-to-1 mux)

A. Write a module for a 2-to-1 mux. (5 pts)

```
module mux_2to1(x, a, b, select);
parameter n = 8;
input select;
input [n-1:0] a, b;
output [n-1:0] x;
...
endmodule
```

B. Write a module for a 8-to-1 mux by using the 2-to-1 mux. (5 pts)

```
module mux_8to1(x, a, b, c, d, e, f, g, h, select);
parameter n = 8;
input [2:0] select;
input [n-1:0] a, b, c, d, e, f, g, h;
output [n-1:0] x;
...
endmodule
```

2. (Design of a 16-bit carry lookahead adder)

A. Write a module for the full adder without using logic gate primitives. (3 pts)

```
module full_adder(sum, c_out, a, b, c_in);
input a, b, c_in;
output sum, c_out;
...
endmodule
```

B. Write a module for a n-bit carry lookahead logic. (3 pts)

```
module carry_lookahead(c_out, g, p, c_in);
parameter n = 4;
output [n-1:0] c_out;
input [n-1:0] g;
input [n-1:0] p;
input c_in;
...
endmodule
```

C. Write a module for an n-bit carry lookahead adder using the **full_adder** and the **carry_lookahead** circuit. (5 pts)

```
module CLA_adder(S, c_out, A, B, c_in);
parameter n = 4;
input [n-1:0] A, B;
input c_in;
output [n-1:0] S;
output c_out;
...
endmodule
```

D. Write a module for a 20-bit CLA by using four 5-bit CLA adders. (5pts)

```
module CLA_20bit_adder(S, c_out, A, B, c_in);
input [19:0] A, B;
input c_in;
output [19:0] S;
output c_out;
...
endmodule
```

3. Explain the behavior of the following module by drawing the waveform for each register. (15 pts)

```
module Block_nonblock;
reg a, b, c, d, e, f;
  initial begin
      a = #10 1;
      b = #2  0;
      c = #4  1;
  end

  initial begin
     d <= #10 1;
     e <= #2  0;
     f <= #4  1;
  end
endmodule
```

4. Explain the difference between the following modules and draw their corresponding circuits. (5 pts + 5 pts)

```
module circuit3_A(enable, a, q, reset_n);
input enable, a, reset_n;
output reg q;
always @(a or enable or reset_n)
    if ((!reset_n) q <=1b'0;
    else if (enable) q <= a;
endmodule

module circuit3_B(enable, a, q, clock, reset_n);
input enable, a, clock, reset_n;
output reg q;
always @(posedge clock or negedge reset_n);
    if (!reset_n) q <=1b'0;
    else if (enable) q <= a;
endmodule
```

5. Answer to the following question regarding to the verilog simulation and its stratified event scheduling.

A. Define the following terminologies. (3 pts)
simulation cycle:
inactive event:

B. Reorder the following events according to their scheduling priority in simulation. (3 pts)

non-blocking assignment event, inactive event, active event, monitor event

C. Write the pseudo code for the verilog simulation reference model. (5 pts)

6. Write a module for a BCD counter. (10 pts)

```
module BCD_counter(clk, start, qout);
parameter n= 8;
input clk, start;
output reg [n-1:0] qout;
...
endmodule
```

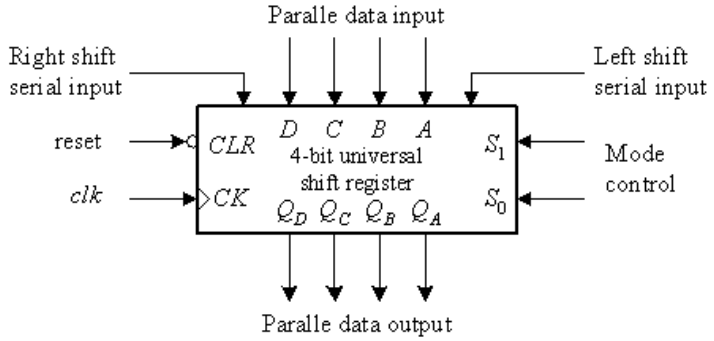
7. Write a module for a ring counter. (10 pts)

```
module Ring_counter(clk, start, qout);
parameter n= 8;
input clk, start;
output reg [n-1:0] qout;
...
endmodule
```

8. Write a module for a priority encoder using a casex statement. (10 pts)

```
module priority_en(y, valid, in);
parameter n = 8;
parameter log2n = 3;
input [n-1:0] in;
input valid;
output reg [log2n-1:0] y;
...
endmodule
```

9. Write a module for a universal shift register. [10 pts]



$s1$	$s0$	Function
0	0	No change
0	1	Right shift
1	0	Left shift
1	1	Load data

```
module universal(clk, reset_n, s0, s1,rsi, lsi, din, qout);
parameter n = 4;
input clk, reset_n;
input s0, s1; // control
input lsi, rsi; // left shift input, right shift input
input [n-1:0] din;
output reg [n-1:0] qout;
...
endmodule
```

10. Assume that x is computed such that x= a * (b - c), based on the sampled values of a, b, c, at the rising edge of clock.

A. Write a module so that the execution is performed in a single cycle and its latency is also 1 clock cycle. (10 pts)

```
module single_cycle(clock, a, b, c, x);
input [7: 0] a, b, c;
input clock;
output [7:0] x;
...
endmodule
```

B. Write a pipelined module so that its latency is also 3 clock cycles. (10 pts)

```
module pipeline_3cycle(clock, a, b, c, x);
input [7: 0] a, b, c;
input clock;
output [7:0] x;
...
endmodule
```

11. Write a module that decide parity error based on 8-bit data, one parity bit, and a control bit that determines even or odd parity. (10 pts)

```
module parity(data, parity, even_odd, error);
input [7: 0] data;
input parity;
input even_odd; // 0 for even parity, 1 for odd parity
output error; // 1 for parity error
...
endmodule
```

12. Write a module for a Moore FSM that checks a parity error based on a sequential string of 8 bits followed by a parity bit. It also has a control bit that determines even or odd parity. (10 pts)

```
module parity(clock, data, even_odd, error);
input data, clock;
input even_odd; // 0 for even parity, 1 for odd parity
output reg error; // 1 for parity error
...
endmodule
```

13. Write a mealy FSM that detects a 0101 sequence by using three always statements. (10 pts)

```
module sequence0101(clock, data, z, reset_n);
input data, clock, reset_n;
output reg z; // 1 if detected
...
endmodule
```