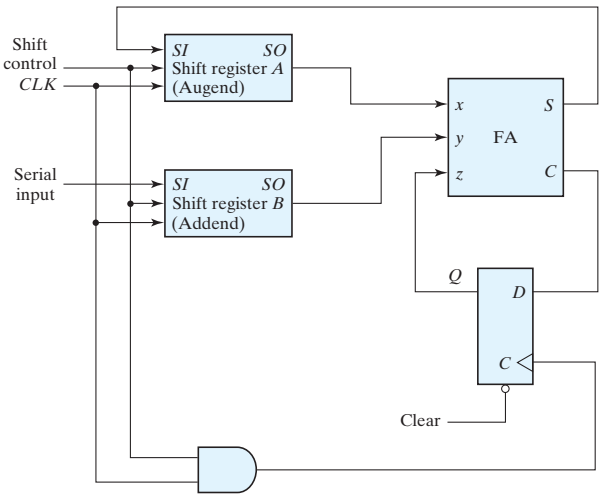
Seung Jun Baek

1. The contents of a four-bit register is initially 0110. The register is shifted six times to the right with the serial input being 1011100. What is the content of the register after each shift?

   Sol: 0110; 0011; 0001; 1000; 1100; 1110; 0111; 1011



2. The serial adder of the figure uses two four-bit registers. Register A holds the binary number 0101 and register B holds 0111. The carry flip-flop is initially reset to 0. List the binary values in register A and the carry flip-flop after each shift.

   Sol: A = 0010, 0001, 1000, 1100. Carry = 1, 1, 1, 0

3. A flip-flops has a 3 ns delay from the time the clock edge occurs to the time the output is complemented. What is the maximum delay in a 10-bit binary ripple counter that uses these flip-flops? What is the maximum frequency at which the counter can operate reliably?

   Sol: 30 ns; 33.3 MHz

4. Using D flip-flops, design a counter with the following repeated binary sequence: 0, 1, 2, 4, 6.

   Sol: We can think of a counter as a state machine. We need 5 states, meaning that we need three bits to represent a state.We have three flip-flops whose outputs are A, B, C. Below is the state table for the counter. Here we have some unused states, so we can simplify the expression of D_A as a function of A, B, C. We can use K-map in Table 2. Note that we can use the "unused bits" expressed in x in the K-map, to further simplify the logic. From K-map we obtain $D_A = A'B + B'A$ or $A \oplus B$. You can perform a similar K-map method to obtain the following:

Table 1: table

Present State ABC	Next State $D_AD_BD_C$
000	001
001	010
010	100
011	xxx
100	110
101	xxx
110	000
111	xxx

Table 2: K-map for D_A

A \ BC	00	01	11	10
0			x	1
1	1	x	x	

$D_A = A \oplus B$

$D_B = AB' + C$

$D_C = A'B'C'$

5. (Verilog) You will be asked to design a "zero-detector" system. Zero-detector takes a bit sequence as input, and will output 1 if two or more consecutive *zeros* are detected in the input sequence. Denote the input signal by x and output by y. Note that the output should become 1 **after** we observe two zeros in the input. Here is the example of signal values sampled over the clock cycles:

$$x : 0110110011000010$$
$$y : 0000000010001100$$

You can see that, when the consecutive zeros appear at the input, the output becomes 1 after one cycle.

(a) Draw a state table.

(b) Based on the state table, derive the logic for the next state. That is, express the next state as a function of present state and x. You should obtain the minimal expression, i.e., use K-map method if applicable.

(c) Derive the output as a function of state variables.

(d) Draw a state diagram.

(e) Implement the verilog code for zero-detector. In the attached file "hw6.v", you can fill in the module "one_detector". Do not change anything else.

(f) Run GTKWave and view waveform. You need to capture waveforms for the following signals from `main` module:

- `CLOCK`
- `p_state[1:0]`
- `t_reset`
- `x_in`

Print out the captured waveform using the following instruction

- When you run GTKWave, go to menu and select "View"→"Use Black and White"
- Go to menu "File"→"Grab to File". This captures the waveform window.
- Save the capture, and print it out. Your captured printout should show the signal waveforms ranging from time 0 to at least time 150 sec.

Sol:

(a) We need three states, which means that we need 2 bits to express present state. See table 3

Table 3: table

	Next State	Next State	Output
Present State	x =0	x =1	
00	01	00	0
01	10	00	0
10	10	00	1

(b) Let A and B denote the bit 1 and 0 of Present state variable. Then, for the bit 1 of the next state we have K-map given by Table 4. Note that, since we don't use the state 11, we can replace the entries of K-map with don't care bits x. As a result we obtain a simple expression $D_A = (A + B)x'$.

Table 4: K-map

x \ AB	00	01	11	10
0		1	x	1
1			x	

For the bit 0 of the next state we have Table 5. We can't really simplify things here, so we have $D_B = A'B'x'$.

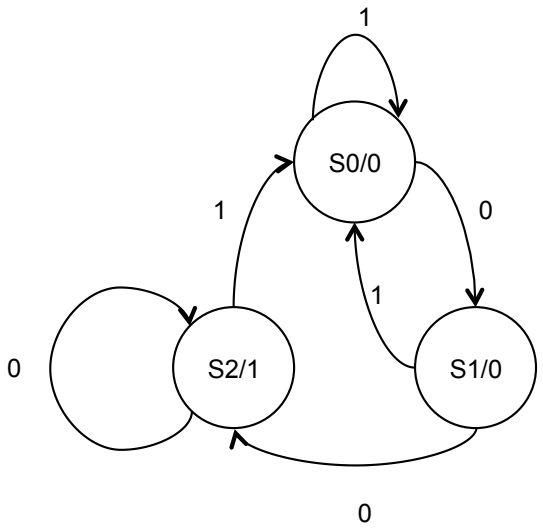
(c) $y = AB'$

(d) Here we set state parameters $S0 = 00$, $S1 = 01$ and $S2 = 10$.

(e)
```
module zero_detector (
output [1:0] present_state,
output y,
input x,
input clock,
```

Table 5: K-map

x \ AB	00	01	11	10
0	1		x	
1			x	




```
input reset );
reg [1:0] state;
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;
always @ (posedge clock, negedge reset)
   if (reset == 0) state <= S0; // Initialize to state S0
   else case (state)
      S0:  if (x) state <= S0; else state <= S1;
      S1:  if (x) state <= S0; else state <= S2;
      S2:  if (x) state <= S0; else state <= S2;
   endcase
assign present_state = state; // Output of flip-flops
assign y = state[1]& ~state[0] ;
endmodule
```

Or, we can implement in alternative way, using the next state logic we derived above:

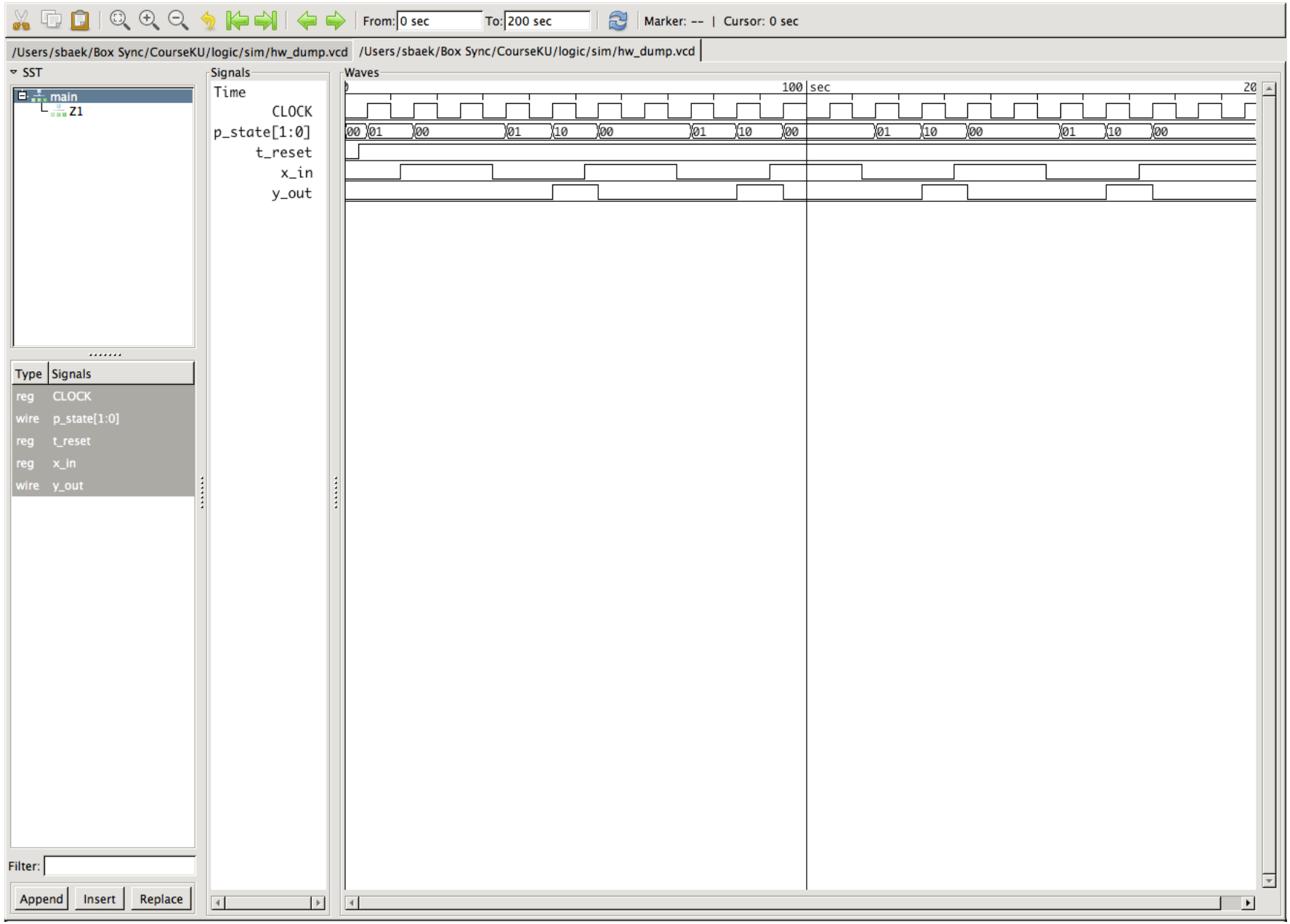
```
module zero_detector (
```

```
output [1:0] present_state,
output y,
input x,
input clock,
input reset );
reg [1:0] state;
wire [1:0] nxt_state;
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;
always @ (posedge clock, negedge reset)
   if (reset == 0) state <= S0; // Initialize to state S0
   else
      state <= nxt_state;

/* We directly implement the next state */
assign nxt_state[1]=(state[1]|state[0])&(~x); /* This is D_A = (A+B)x' */
assign nxt_state[0]=~state[1]&~state[0]&~x; /* This is D_B = A'B'x' */

assign present_state = state; // Output of flip-flops
assign y = state[1]& ~state[0] ;
endmodule
```

(f)


From: 0 sec
To: 200 sec
Marker: -- | Cursor: 0 sec
/Users/sbaek/Box Sync/CourseKU/logic/sim/hw_dump.vcd
/Users/sbaek/Box Sync/CourseKU/logic/sim/hw_dump.vcd
SST
main
Z1
Type
Signals
reg CLOCK
wire p_state[1:0]
reg t_reset
reg x_in
wire y_out
Filter:
Append
Insert
Replace
Signals
Time
CLOCK
p_state[1:0]
t_reset
x_in
y_out
Waves
100 sec
20
00 01 00 01 10 00 01 10 00 01 10 00 01 10 00