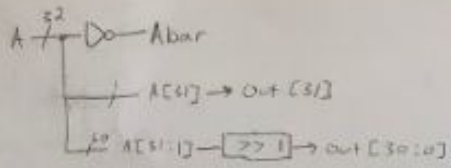
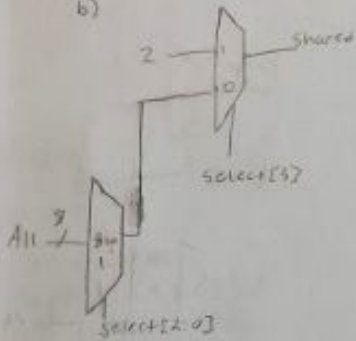


1) a)

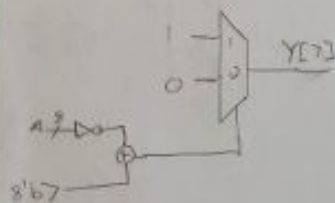
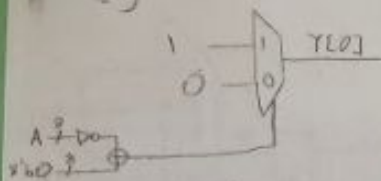


b)



A			Y							
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

c) contd.)



Always 0 (0)

d) Case $x \in \{A, B, C\}$

$x = A: H = F \oplus G;$

$x = B: H = F \oplus G;$

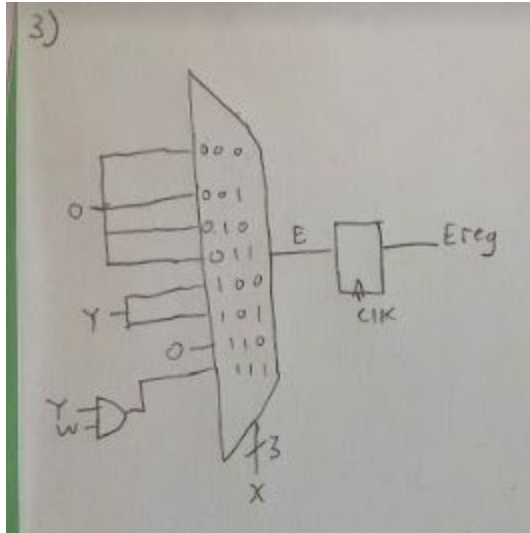
$x = C: H = F \oplus G;$

Default: $H = 0$

```

2) reg[7:0] A;
   reg[2:0] L;
   reg[2:0] A_accum;
   reg Parity;
   Always@ (posedge clk)
   begin
       A_accum <= 3'b0;
       For(L=0; L<7; L=L+1)
       begin
           if A[L] == 1'b1
           begin
               A_accum = A_accum + 1'b1;
           end
           else begin
               A_accum = A_accum + 3'b0;
           end
       end
       if (A_accum == 3'b0) || (A_accum % 2 == 0)
       begin
           Parity <= 1'b1;
       end
       else begin
           Parity <= 1'b0;
       end
   end

```



4)

Synopsys Errors:

```

djmock@dhaltsim:~/Documents/ECE_564/hw_4
File Edit View Search Terminal Tabs Help
djmock@dhaltsim:~/Documents/ECE_564/hw_4 x B v
This software and the associated documentation are proprietary to Synopsys,
Inc. This software may only be used in accordance with the terms and conditions
of a written license agreement with Synopsys, Inc. All other use, reproduction,
or distribution of this software is strictly prohibited.
Initializing...
design_vision> design_vision>
design_vision>
design_vision> source setup.tcl
10
design_vision> source read.tcl
Loading db file '/afs/eos.ncsu.edu/lockers/research/ece/wdavis/tech/nangate/NangateOpenCellLibrary_PDKv1_2_v2008_10/liberty/520/NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm.db'
Loading db file '/afs/eos.ncsu.edu/software/synopsys2015/syn/libraries/syn/dw_foundation.sldb'
Loading db file '/afs/eos.ncsu.edu/software/synopsys2015/syn/libraries/syn/gtech.db'
Loading db file '/afs/eos.ncsu.edu/software/synopsys2015/syn/libraries/syn/standard.sldb'
Loading link library 'NangateOpenCellLibrary_PDKv1_2_v2008_10_slow_nldm'
Loading link library 'gtech'
Loading verilog file '/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'
Detecting input file type automatically (-rtl or -netlist).
Reading with Presto HDL Compiler (equivalent to -rtl option).
Running PRESTO HDLC
Compiling source file /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:21: The construct 'non-ANSI-style int
erface port declaration' is not supported in this language. (VER-720)
*** Presto compilation terminated with 1 errors. ***
Error: Can't read 'verilog' file '/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'. (UID-59)
No designs were read

```

```

djmock@dhsim:~/Documents/ECE_564/hw_4
File Edit View Search Terminal Tabs Help
djmock@dhsim:~/Documents/ECE_564/hw_4 djmock@dhsim:~/Documents/ECE_564/hw_4
Inferred memory devices in process
in routine fsm line 71 in file
'/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'.
=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| outcell_reg | Flip-flop | 9 | Y | N | N | N | N | N | N |
=====

Inferred memory devices in process
in routine fsm line 74 in file
'/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'.
=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| outcell_reg2 | Flip-flop | 9 | Y | N | N | N | N | N | N |
=====

Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[8]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[7]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[6]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[5]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[4]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[3]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[2]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[1]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
Error: /afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v:74: Net 'outcell[0]' or a directly connected net is driven by more
than one source, and not all drivers are three-state. (ELAB-366)
*** Presto compilation terminated with 9 errors. ***
Error: Can't read 'verilog' file '/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'. (UID-59)
No designs were read
design_vision>

```

Synopsys output after errors fixed:

```

djmock@dhsim:~/Documents/ECE_564/hw_4
File Edit View Search Terminal Tabs Help
djmock@dhsim:~/Documents/ECE_564/hw_4 djmock@dhsim:~/Documents/ECE_564/hw_4
=====
| Line | full/ parallel |
=====
| 39 | auto/auto |
=====

Inferred memory devices in process
in routine fsm line 27 in file
'/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'.
=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| current_state_reg | Flip-flop | 1 | N | N | Y | N | N | N | N |
=====

Inferred memory devices in process
in routine fsm line 37 in file
'/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'.
=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| inc_flag_reg | Latch | 1 | N | N | N | N | - | - | - |
| location_reg | Latch | 9 | Y | N | N | N | - | - | - |
=====

Inferred memory devices in process
in routine fsm line 71 in file
'/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/badFSM.v'.
=====
| Register Name | Type | Width | Bus | MB | AR | AS | SR | SS | ST |
=====
| outcell_reg | Flip-flop | 9 | Y | N | N | N | N | N | N |
=====

Presto compilation completed successfully.
Current design is now '/afs/unity.ncsu.edu/users/d/djmock/Documents/ECE_564/hw_4/fsm.db:fsm'
Loaded 1 design.
Current design is 'fsm'.
fsm
Current design is 'fsm'.
design_vision>

```

Problem:

Line 21: erg signal;

(Syntax Error)

Fix:

Replace with: reg signal;

Problem:

Line 37: always @(current_state or start or done_flag)

(Sensitivity list incomplete)

Fix:

Replace with: always @(current_state or start or done_flag or match_address)

Problem:

Lines 40-52:

```
s0:
  begin
    if (start)
      begin
        location = 9'd0;
        next_state = s1;
      end
    else
      begin
        inc_flag = 0;
        next_state = s0;
      end
```

(Incorrect logic of state machine)

Fix:

Replace with:

```
s0:
  begin
```

```

if (start)
  begin
    next_state = s1;
  end
else
  begin
    location = 9'd0;
    inc_flag = 0;
    next_state = s0;
  end

```

Problem:

Lines 54-65

```

s1:
  begin
    if (done_flag)
      begin
        location = match_address;
        next_state = s0;
      end
    else
      begin
        inc_flag = 1;
        next_state = s1;
      end

```

(Incorrect state machine logic)

Fix:

Replace with:

```

s1:
  begin
    location = match_address;
    inc_flag = 1;
    if (done_flag)
      begin
        next_state = s0;
      end
    else
      begin

```

```
next_state = s1;  
end
```

Problem:

Lines 71-75:

```
always@(posedge clock)  
    outcell = location ^ (location << 1);
```

```
always@(posedge clock)  
    outcell = location ^ (location >> 1);
```

(driven by more than 1 source)

Fix:

```
always@(posedge clock)  
    outcell = location ^ (location >> 1);
```

Final Verilog Code:

// Verilog file for the fsm for the pattern matching engine

```
module fsm (clock, reset, start, done_flag, match_address, inc_flag, location,  
outcell);
```

```
input clock;           // 100 Mhz clock  
input reset;           // resets the fsm  
input start;           // starts the search  
input [8:0] match_address; // address for the pattern match
```

```
input done_flag;       // signal from compare module saying it has finished  
                      // its search
```

```
output inc_flag;       // used to increment the address location  
output [8:0] location; // location output for pattern match  
output [8:0] outcell;  // A hash on location
```

```
reg [8:0] location, outcell;  
reg current_state, next_state;  
reg inc_flag;
```

```
reg signal;
```

```
parameter
```

```
    s0 = 0,
```

```
    s1 = 1;
```

```
always @(posedge clock or negedge reset)
```

```
begin
```

```
if (!reset)
```

```
    current_state = s0;
```

```
else
```

```
    begin
```

```
        current_state = next_state;
```

```
    end
```

```
end
```

```
always @(current_state or start or done_flag or match_address)
```

```
begin
```

```
case (current_state)
```

```
s0:
```

```
    begin
```

```
    if (start)
```

```
        begin
```

```
            next_state = s1;
```

```
        end
```

```
    else
```

```
        begin
```

```
            location = 9'd0;
```

```
            inc_flag = 0;
```

```
            next_state = s0;
```

```
        end
```

```
    end
```

```
s1:
```

```
    begin
```

```
        location = match_address;
```

```
        inc_flag = 1;
```

```
        if (done_flag)
```

```
            begin
```

```
                next_state = s0;
```

```
            end
```

```
        else
```

```
            begin
```



```
    next_state = s1;  
    end  
end
```

```
endcase  
end
```

```
always@(posedge clock)  
    outcell = location ^ (location << 1);
```

```
always@(done_flag)  
    signal = done_flag & (^location[4:2]);
```

```
endmodule
```