HW 4

Saturday, October 5, 2024 8:46 PM

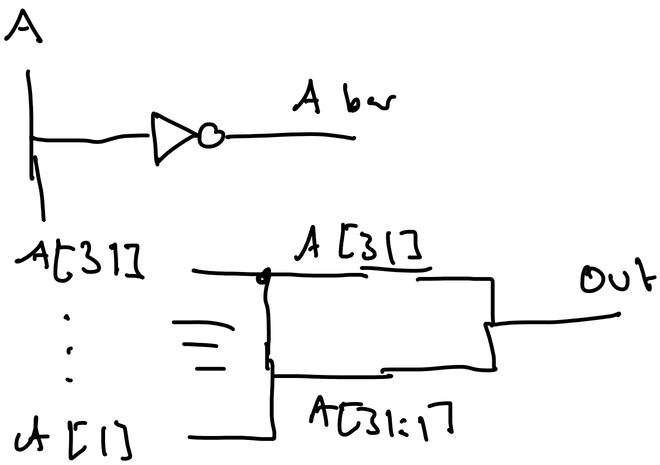
Thiago Gesteira 20043117 3

1

a.

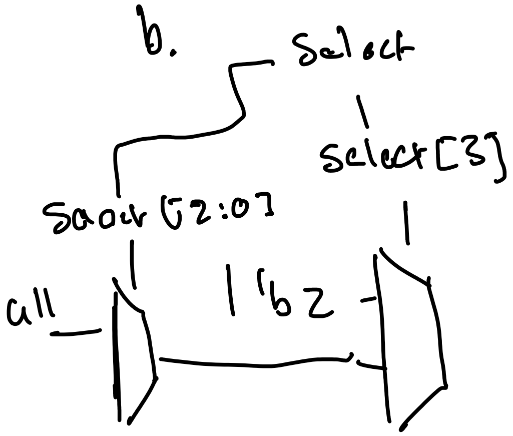
concatenation?

assign out = {A[31], A[31:1]}



b.

all[select[2:0]]

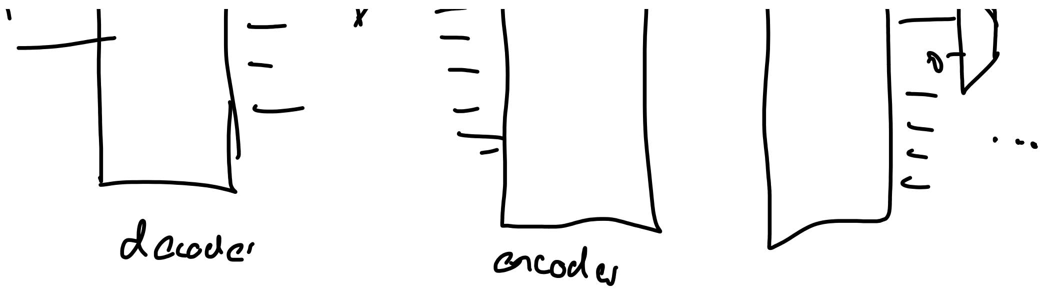


c.

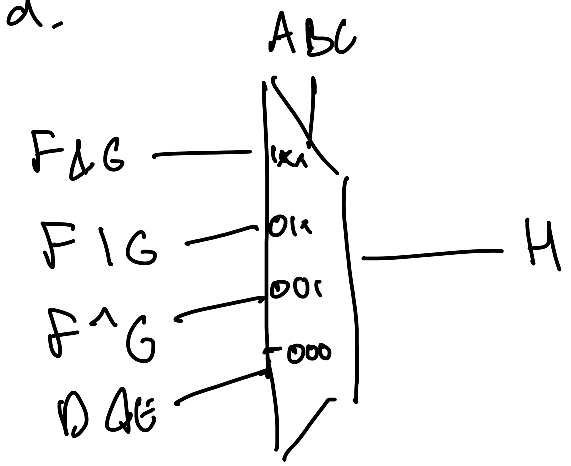
always @ (A) ?

for N

if (A== N) Y[N] = 1



d.

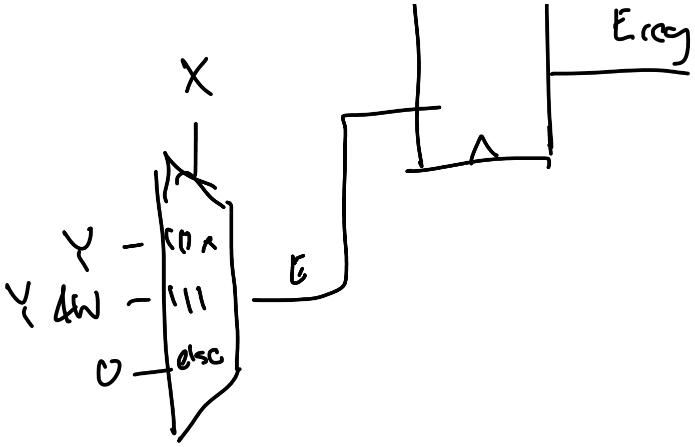


```
always @ (*)
  casex ({A,B,C}) :
    case 1xx: F&G
    case 01x: F|G
    case 001: F^G
    default: D&E
  end case
```

2.

```
reg [7:0] A;
reg parity;
parity = A[0]
always @ (posedge clk
   for(i = 1; i <= 7; i = i+1)
        parity = parity ^ A[i];
   parity = ~parity
```

3.



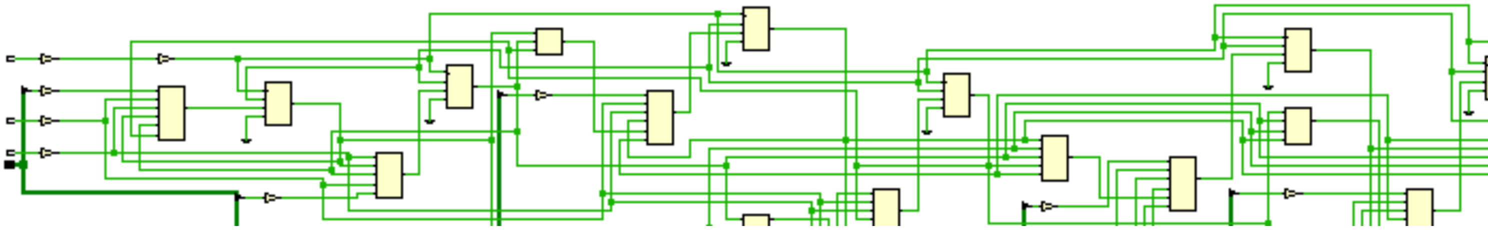
4. ☆ redo

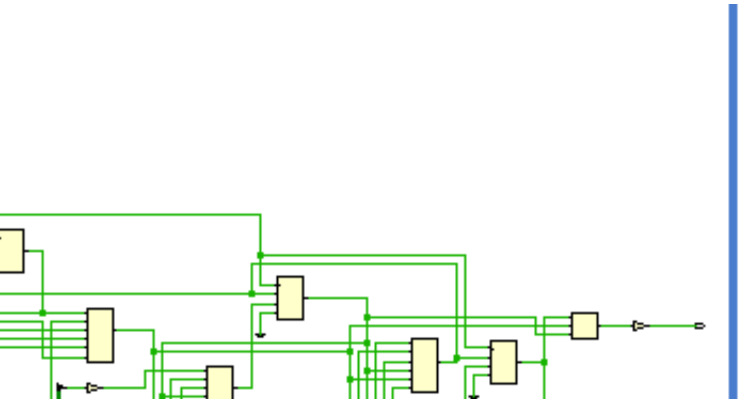
```
// Verilog file for the FSM for the pattern matching engine
module fsm (
   input clock,           // 100 MHz clock
   input reset,           // resets the FSM
   input start,           // starts the search
   input [8:0] match_address, // address for the pattern match
   input done_flag,       // signal from compare module that search is finished

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

// State encoding
parameter s0 = 1'b0;
parameter s1 = 1'b1;

reg current_state, next_state;

// Synchronous reset and state transition
always @(posedge clock or negedge reset) begin
   if (!reset)
      current_state <= s0;
   else
      current_state <= next_state;
```

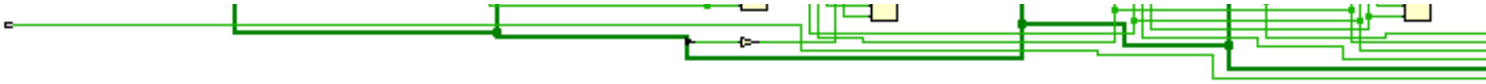
```
end

// Next state logic and output logic
always @(current_state or start or done_flag) begin
   // Default values
   next_state = current_state; // remain in current state unless changed
   inc_flag = 1'b0;           // default to no increment

   case (current_state)
      s0: begin
         if (start) begin
            location = 9'd0;   // reset location
            next_state = s1;   // move to state S1
         end else begin
            inc_flag = 1'b0;   // remain in state S0
         end
      end

      s1: begin
         if (done_flag) begin
            location = match_address; // set location to match address
            next_state = s0;          // move back to state S0
         end else begin
            inc_flag = 1'b1;    // continue incrementing location
            next_state = s1;    // stay in state S1
         end
      end
   endcase
end

// Hash calculation on location
always @(posedge clock) begin
   outcell <= location ^ (location << 1); // example hash, could adjust
end

endmodule
```

Problem	Fix
State register size not explicitly defined: The state register current_state and next_state were not properly declared, potentially causing issues in state encoding.	Changed the state register declaration to reg current_state, next_state; to reg [1:0] current_state, next_state;. This ensures proper state encoding.
Synchronous logic using blocking assignment: The state transitions were using blocking assignments (=) instead of non-blocking (<=). This can cause issues with timing in FPGA/ASIC designs.	Changed the assignment inside the always block for state transitions to non-blocking assignments (<=).
Location and inc_flag not initialized properly on reset: The reset logic did not ensure proper initialization of key variables like location and inc_flag.	Added initialization for location and inc_flag in the reset block to ensure proper reset behavior.
Duplicate outcell assignment in separate always blocks: Two separate always blocks were assigning values to outcell, which is not allowed in Verilog and would cause synthesis issues.	Merged the two outcell assignments into a single always block. The hash function is computed based on the updated value of location.
Blocking assignment in combinational logic: Combinational logic block that determines the next state and output logic was using blocking assignments (=).	Converted the assignments inside the always block for combinational logic to use non-blocking assignments where appropriate.

5. Xillinx







2.4 ns