ECE-399: Selected Topics in ECE: Design with FPGAs
Report 3: Sequential Logic Design, Decoders and ROM

Prof. Shlayan
Nithilam Subbaian

May 15, 2020

1 Objective of Report 3:

For this report, the goal is to familiarize myself with Decoders and ROM in general. Shown at the end are select exercises to learn these concepts.

2 Materials

1. Zedboard [6]
2. 4GB SD card (purchased independently of Zedboard Kit)
3. 12V Power supply
4. Micro USB cable
5. USB Adapter: Male Micro-B to Female Standard-A
6. Xilinx Vivado Design Suite Version 2018.3 [9]

3 Lecture 6 Notes

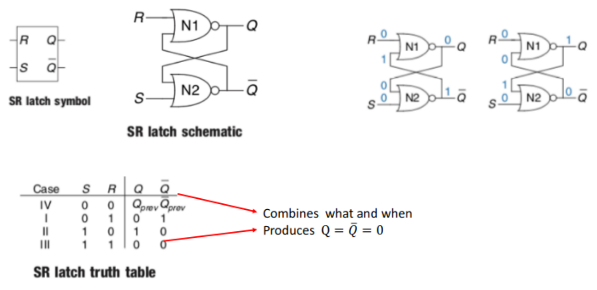
This notes are based of Lecture 6 [7].

3.1 Sequential Logic

The outputs of sequential logic depend on both current and prior input values, therefore sequential logic has memory. Certain previous inputs are explicitly remembered or distilled into the state of the system, a set of bits called state variables that contain all the information about the past necessary to explain the future behavior of the circuit. Examples of simple sequential circuits are Latches and Flip-Flops. There also exist, synchronous sequential circuits, finite state machines in which speed and parallelism are important features to analyze.

Fundamental building block of memory is a bistable element, an element with two stable states. An example of this is a cross-coupled inverter pair, it is cyclic: Q depends on Q not, and vice versa. An element with N stable states conveys log_2N bits of information, so a bistable element stores one bit Other bistable elements, such as latches and flip-flops, provide inputs to control the value of the state variable .

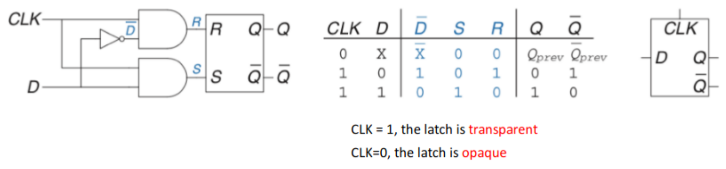
1. **SR Latch (figure 1):** Its state can be controlled through the Sand R inputs, which set and reset the output Q.



Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

Figure 1: SR Latch

2. **D Latch (figure 2):** Data input, D, controls what the next state should be. The clock input, CLK, controls when the state should change.In all cases, Q not is the complement of Q. The D latch avoids the case of simultaneously asserted R and S inputs. *The D latch updates its state continuously while CLK = 1.*



CLK	D	$\bar{D}$	S	R	Q	$\bar{Q}$
0	X	$\bar{X}$	0	0	Q_{prev}	$\bar{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

Figure 2: D Latch

3. **D Flip-Flop (figure 3):** Two back-to-back D latches controlled by complementary clocks. When CLK = 0, the mastervlatch is transparent and the slave is opaque, the value at D propagates through to N1. When CLK = 1, the master goes opaque and the slave becomes transparent, the value at N1 propagates through to Q, but N1 is cut off from D. *D flip-flop copies D to Q on the rising edge of the clock, and remembers its state at all other times.*

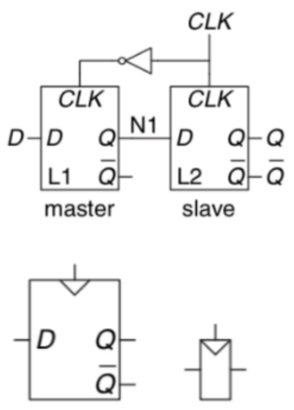


Figure 3: D Flip-Flop

4. **Register (figure 4):** An N-bit register is a bank of N flip-flops that share a common CLK input, so that all bits of the register are updated at the same time.

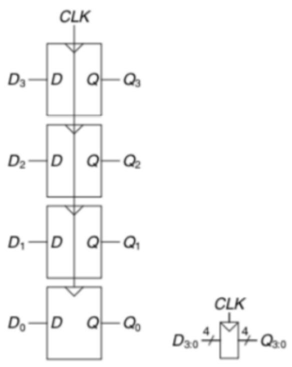


Figure 4: Register

5. **Enabled Flip-Flop (figure 5):** When EN is FALSE, the enabled flip-flop ignores the clock and retains its state.

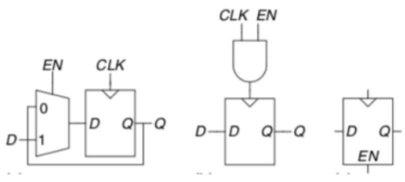


Figure 5: Enabled Flip-Flop

6. **Resettable Flip-Flop (figure 6):** When RESET is FALSE: the flip-flop behaves like an ordinary D flip-flop. When RESET is TRUE: the flip-flop ignores D and resets the output to 0. May be synchronously (reset only on the rising edge of CLK) or asynchronously (Reset as soon as RESET becomes TRUE, independent of CLK) resettable.

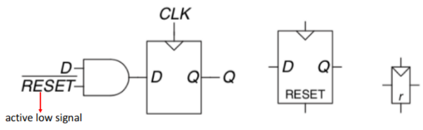


Figure 6: Resettable Flip-Flop

3.2 Synchronous Logic Design

Sequential circuits with cyclic paths can have undesirable races or unstable behavior. To avoid these problems:Break the cyclic paths by inserting registers in the feedback path which contain the state of the system that are synchronized to the clock.

Synchronous sequential circuits consists of interconnected circuit elements such that: Every circuit element is either a register or a combinational circuit, at least one circuit element is a register, all registers receive the same clock signal, every cyclic path contains at least one register.

3.3 Finite State Machines

Finite State Machines provide a systematic way to design synchronous sequential circuits given a functional specification. A circuit with k registers can be in one of a finite number (2K) of unique states. Example can be seen in figure 7.

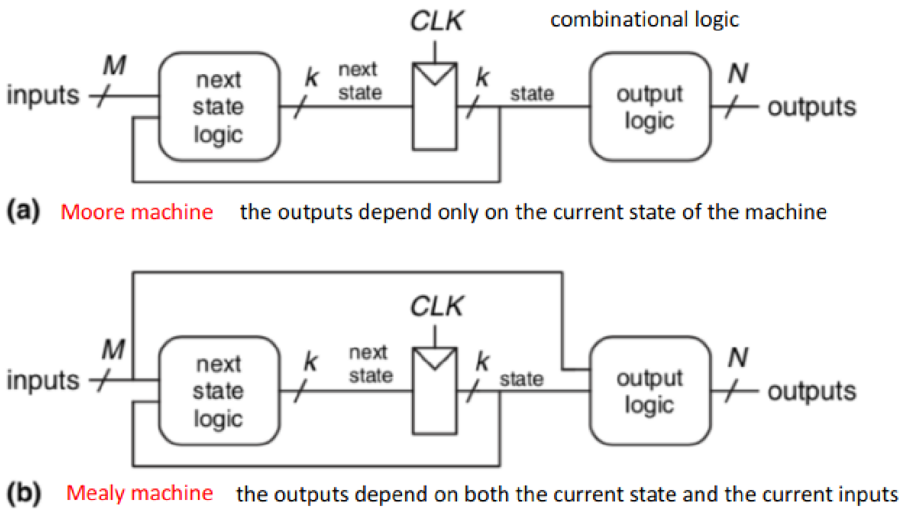

CLK
combinational logic
inputs
M
next state logic
k
next state
k
state
output logic
N
outputs
(a) Moore machine the outputs depend only on the current state of the machine
CLK
inputs
M
next state logic
k
next state
k
state
output logic
N
outputs
(b) Mealy machine the outputs depend on both the current state and the current inputs

Figure 7: Resettable Flip-Flop

3.4 State Transition Diagram

In state transition diagrams, circles represent states and arcs represent transitions between states. The transitions take place on the rising edge of the clock.

State transition table indicates, for each current state and input, what the next state, S, should be. To build a circuit the states and outputs must be assigned binary encodings.

To determine the encoding that produces the circuit with the fewest logic gates or the shortest propagation delay, use inspection! Choice between binary encoding and one-hot encoding (one bit is "hot" or TRUE at any time) i.e. one-hot encoded FSM with three states would have state encodings of 001, 010, and 100Each bit of state is stored in a flip-flop, so one-hot encoding requires more flip-flops than binary encoding. However, with one-hot encoding, the next-state and output logic is often simpler, so fewer gates are required.

3.5 Deriving an FSM from a Schematic

1. Examine circuit, stating inputs, outputs, and state bits.
2. Write next state and output equations.
3. Create next state and output tables.
4. Reduce the next state table to eliminate unreachable states.
5. Assign each valid state bit combination a name.
6. Rewrite next state and output tables with state names.
7. Draw state transition diagram.
8. State in words what the FSM does.

3.6 Timing

A sequential element has an aperture time around the clock edge, defined by a setup time and a hold time, before and after the clock edge, respectively. During which the input must be stable for the flip-flop to produce a well-defined output. Static discipline limited us to using logic levels outside the forbidden zone, the dynamic discipline limits us to using signals that change outside the aperture time. The clock period has to be long enough for all signals to settle. The skew, due to variation in arrival of the clock does to all flip-flops, further increases the clock period.Using synchronizers when interfacing with asynchronous inputs.

3.7 Read-Only Memories (ROM)

ROM consists of interconnected arrays to store an array of binary information. m address input pins and n information output pins to store 2m words information.

1. Once the binary information is stored it can be read any time but cannot be altered
2. it can be used to implement combinatorial circuits
3. It uses a decoder
4. An address is provided at the input and content of the corresponding word is read at the output pins.

In order for the memory to be read only memory should only be read and not written into; memory should somehow be initialized with the desired content.

3.8 Virtual Input/output (VIO) Core

The LogiCORETMIP Virtual Input/Output(VIO) core is a customizable core that can both monitor and drive internal FPGA signals in real time.

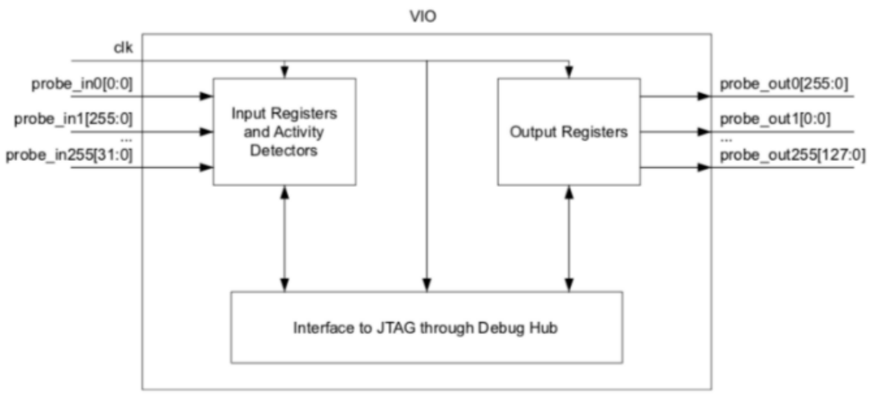


Figure 8: Resettable Flip-Flop

4 Vivado and Zedboard Work

Overall good reviews that helped me with these exercises are [3], [5], and [2].

4.1 Design a 3-to-8 line decoder. Assign the input to SW2-SW0 and output to LED7-LED0. Use dataflow modeling constructs.

```verilog

module to8decoder( in, out );

input wire [2:0] in;
output wire [7:0] out;

assign out[0] = (~in[2] & ~in[1] & ~in[0]);
assign out[1] = (~in[2] & ~in[1] & in[0]);
assign out[2] = (~in[2] & in[1] & ~in[0]);
assign out[3] = (~in[2] & in[1] & in[0]);
assign out[4] = (in[2] & ~in[1] & ~in[0]);
assign out[5] = (in[2] & ~in[1] & in[0]);
assign out[6] = (in[2] & in[1] & ~in[0]);
assign out[7] = (in[2] & in[1] & in[0]);

endmodule
```

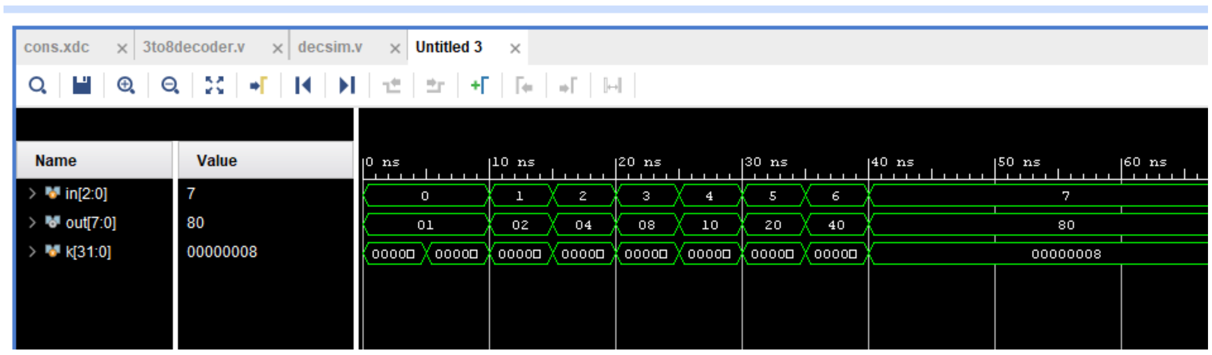


Figure 9: 3-to-8 line decoder

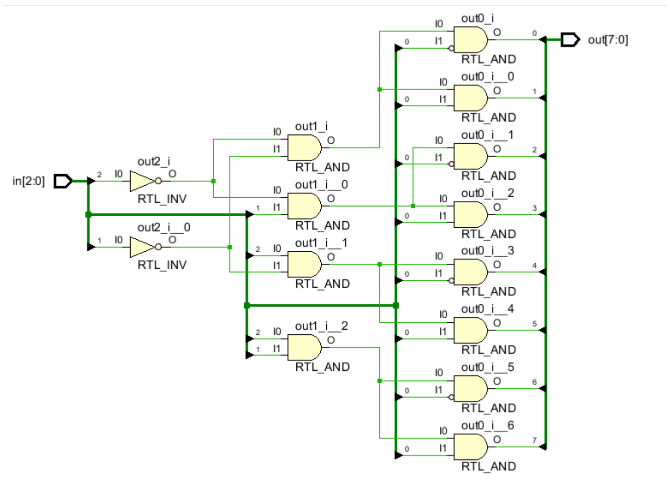


Figure 10: 3-to-8 line decoder

4.2 Implement a 2-bit by 2-bit multiplier using a ROM and $readmemb system task Use LEDs to display the output in binary.

I used [8] to understand what $readmemb meant as well as these resources for this exercise: [4], [1].

```
//  Implement a 2-bit by 2-bit multiplier
// using a ROM and $readmemb system task
// Use LEDs to display the output in binary.
module rom ( ROM_data,  ROM_addr );

output [3:0] ROM_data;
input  [3:0]  ROM_addr;
reg [3:0] ROM [15:0];

assign ROM_data = ROM[ROM_addr];
initial $readmemb("C:/Users/vishm/project_4/romcontent.txt", ROM, 0, 15);

endmodule
```

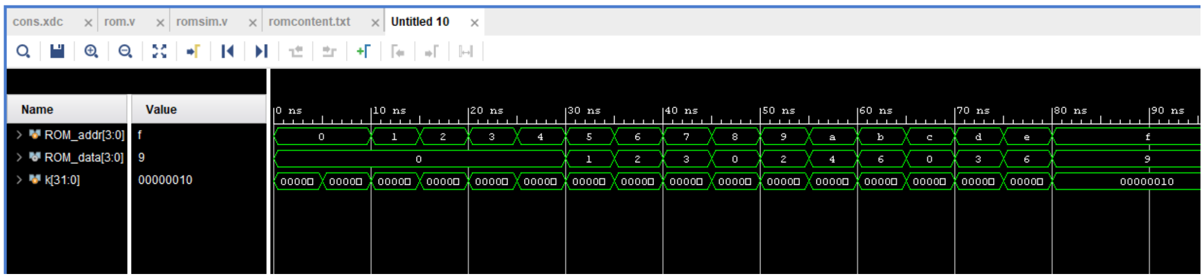


Figure 11: 3-to-8 line decoder

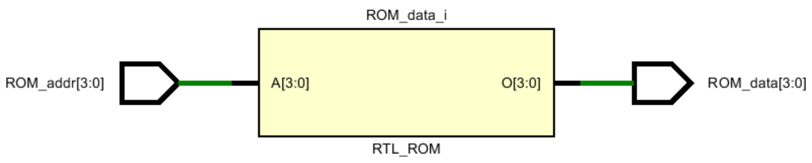


Figure 12: 3-to-8 line decoder

References

[1] Columbia CS. *2-bit multiplier.* URL: http://www.cs.columbia.edu/~martha/courses/3827/sp11/slides/2bit_multiplier_soln.pdf.

[2] UC Davis. *VERILOG 6: DECODER DESIGN EXAMPLES.* URL: https://www.ece.ucdavis.edu/~bbaas/281/notes/Handout.verilog6.pdf.

[3] EngMicroLectures. *Electronics Review.* URL: https://www.youtube.com/channel/UCv5e49EfXTD94Q3dcR5485Q.

[4] Koray Koca. *Multiplier Design Example using ROM, DECODER and MULTIPLEXER.* URL: https://www.youtube.com/watch?v=Squion4z0kE.

[5] Tutorials Point (India) Ltd. *Decoders.* URL: https://www.youtube.com/watch?time_continue=4&v=AFBoQ0zSdTI&feature=emb_logo.

[6] Avnet Electronics Marketing. *Zynq^{TM}Evaluation and Development Hardware User's Guide.* URL: https://reference.digilentinc.com/_media/zedboard:zedboard_ug.pdf.

[7] Naveen Shlayan. *Sequential Logic Design.* URL: https://moodle.cooper.edu/moodle/pluginfile.php/82181/mod_resource/content/1/Lecture6.pdf.

[8] Xilinx. *Read-Only Memories.* URL: https://www.xilinx.com/support/documentation/university/ISE-Teaching/HDL-Design/14x/Nexys3/Verilog/docs-pdf/lab3.pdf.

[9] xlinx. *Vivado Design Suite - HLx Editions.* URL: https://www.xilinx.com/products/design-tools/vivado.html.