

ECE 274 - Digital Logic

Lecture 14

- Lecture 14 – RTL Design
 - Four Step RTL Design Method

1

RTL Design Method

5.2

Step	Description
1. Capture a high-level state machine	Describe the system's desired behavior as a high-level state machine. The state machine consists of states and transitions. The state machine is "high-level" because the transition conditions and the state actions are more than just boolean operations on bit inputs and outputs.
2. Create a datapath	Create a datapath to carry out the data operations of the high-level state machine.
3. Connect the datapath to a controller block. Connect external boolean inputs and outputs to the controller block.	Connect the datapath to a controller block. Connect external boolean inputs and outputs to the controller block.
4. Derive the controller's FSM	Convert the high-level state machine to a finite state machine (FSM) for the controller, by replacing data operations with setting and reading of control signals to and from the datapath.

2

Step 1: Create a High-Level State Machine

- Let's consider each step of the RTL design process in more detail
- Step 1
 - Soda dispenser example
 - Not an FSM because:
 - Multi-bit (data) inputs a and s
 - Local register tot
 - Data operations $tot=0$, $tot<s$, $tot=tot+a$.
 - Useful high-level state machine:
 - Data types beyond just bits
 - Local registers
 - Arithmetic equations/expressions

Inputs: c (bit), a (8 bits) s (8 bits)
Outputs: d (bit)
Local registers: tot (8 bits)

3

Step 1 Example: Laser-Based Distance Measurer

- Example of how to create a high-level state machine to describe desired processor behavior
- Laser-based distance measurement – pulse laser, measure time T to sense reflection
 - Laser light travels at speed of light, $3 \cdot 10^8$ m/sec
 - Distance is thus $D = T \text{ sec} \cdot 3 \cdot 10^8 \text{ m/sec} / 2$

4

Step 1 Example: Laser-Based Distance Measurer

- Laser-based Distance Measurer
 - Used in adaptive cruise control systems for automobiles
 - Maintain particular "safe" distance between car in front
 - May need to decrease cruise speed
 - Need to measure the distance to the car in front

5

Step 1 Example: Laser-Based Distance Measurer

- Inputs/outputs
 - B : bit input, from button to begin measurement
 - L : bit output, activates laser
 - S : bit input, senses laser reflection
 - D : 16-bit output, displays computed distance

6

Step 1 Example: Laser-Based Distance Measurer

Inputs: B, S (1 bit each)
Outputs: L (bit), D (16 bits)

State Machine:

```

    graph LR
      S0((S0)) --> Q[?]
      S0 -- L=0 (laser off)  
D=0 (distance=0)
  
```

- Step 1: Create high-level state machine
- Begin by declaring inputs and outputs
- Create initial state, name it S0
 - Initialize laser to off (L=0)
 - Initialize displayed distance to 0 (D=0)

7

Step 1 Example: Laser-Based Distance Measurer

Inputs: B, S (1 bit each)
Outputs: L (bit), D (16 bits)

State Machine:

```

    graph LR
      S0((S0)) --> S1((S1))
      S1 --> S1
      S1 --> Q[?]
      S0 -- L=0  
D=0
      S1 -- B (button not pressed)
      S1 -- B (button pressed)
  
```

- Add another state, call S1, that waits for a button press
 - B' - stay in S1, keep waiting
 - B - go to a new state S2

Q: What should S2 do? A: Turn on the laser

8

Step 1 Example: Laser-Based Distance Measurer

Inputs: B, S (1 bit each)
Outputs: L (bit), D (16 bits)

State Machine:

```

    graph LR
      S0((S0)) --> S1((S1))
      S1 --> S1
      S1 --> S2((S2))
      S2 --> S3((S3))
      S0 -- L=0  
D=0
      S1 -- B
      S2 -- L=1 (laser on)
      S3 -- L=0 (laser off)
  
```

- Add a state S2 that turns on the laser (L=1)
- Then turn off laser (L=0) in a state S3

Q: What do next? A: Start timer, wait to sense reflection

9

Step 1 Example: Laser-Based Distance Measurer

Inputs: B, S (1 bit each)
Outputs: L (bit), D (16 bits)
Local Registers: Dctr (16 bits)

State Machine:

```

    graph LR
      S0((S0)) --> S1((S1))
      S1 --> S1
      S1 --> S2((S2))
      S2 --> S3((S3))
      S3 --> S3
      S3 --> S1
      S0 -- L=0  
D=0
      S1 -- Dctr=0 (reset/cycle count)
      S2 -- L=1
      S3 -- L=0  
Dctr=Dctr+1 (count cycles)
  
```

- Stay in S3 until sense reflection (S)
- To measure time, count cycles for which we are in S3
 - To count, declare local register Dctr
 - Increment Dctr each cycle in S3
 - Initialize Dctr to 0 in S1. S2 would have been O.K. too

10

Step 1 Example: Laser-Based Distance Measurer

Inputs: B, S (1 bit each)
Outputs: L (bit), D (16 bits)
Local Registers: Dctr (16 bits)

State Machine:

```

    graph LR
      S0((S0)) --> S1((S1))
      S1 --> S1
      S1 --> S2((S2))
      S2 --> S3((S3))
      S3 --> S3
      S3 --> S4((S4))
      S4 --> S1
      S0 -- L=0  
D=0
      S1 -- Dctr=0
      S2 -- L=1
      S3 -- L=0  
Dctr=Dctr+1
      S4 -- D=Dctr/2 (calculate D)
  
```

- Once reflection detected (S), go to new state S4
 - Calculate distance
 - Assuming clock frequency is 3×10^8 , Dctr holds number of meters, so $D = Dctr/2$
- After S4, go back to S1 to wait for button again

11

Step 2: Create a Datapath

- Datapath must
 - Implement data storage
 - Implement data computations
- Look at high-level state machine, do three substeps
 - Make data inputs/outputs be datapath inputs/outputs
 - Instantiate declared registers into the datapath (also instantiate a register for each data output)
 - Examine every state and transition, and instantiate datapath components and connections to implement any data computations

Instantiate: to introduce a new component into a design.

12

Step 2 Example: Laser-Based Distance Measurer

(a) Make data inputs/outputs be datapath inputs/outputs

(b) Instantiate registers into the datapath (also instantiate a register for each data output)

(c) Examine every state and transition, and instantiate datapath components and connections to implement any data computations

Inputs B (S1 bit each) Outputs L (bit), D (16 bits)
LocalRegisters Dctr (16 bits)

Datapath components: Dreg_clr, Dreg_id, Dctr_clr, Dctr_cnt, clear count, Dctr: 16-bit up-counter, clear load, Dreg: 16-bit register, I, O, 16-bit bus.

13

Step 2 Example: Laser-Based Distance Measurer

(c) (continued) Examine every state and transition, and instantiate datapath components and connections to implement any data computations

Inputs B (S1 bit each) Outputs L (bit), D (16 bits)
LocalRegisters Dctr (16 bits)

Datapath components: Dreg_clr, Dreg_id, Dctr_clr, Dctr_cnt, clear count, Dctr: 16-bit up-counter, clear load, Dreg: 16-bit register, I, O, 16-bit bus, >>1 (right shift).

14

Step 2 Example Showing Mux Use

(a) $R = E + F$

(b) $R = R + G$

(c) $R = R + G$ (with mux)

(d) $R = R + G$ (with mux and adders)

LocalRegisters E, F, G, R (16 bits)

Introduce mux when one component input can come from more than one source

15

Step 3: Connecting the Datapath to a Controller

from button B, to display D (16 bits), 300 MHz Clock

Controller signals: Dreg_clr, Dreg_id, Dctr_clr, Dctr_cnt

Datapath signals: Dreg_clr, Dreg_id, Dctr_clr, Dctr_cnt

to laser L, from sensor S

- Laser-based distance measurer example
- Easy – just connect all control signals between controller and datapath

16

Step 4: Deriving the Controller's FSM

FSM has same structure as high-level state machine

- Inputs/outputs all bits now
- Replace data operations by bit operations using datapath

Inputs B (S1 bit each) Outputs L (bit), D (16 bits)
LocalRegisters Dctr (16 bits)

FSM States and Transitions:

- S0: L=0, Dreg_clr=1 (laser #), Dreg_id=0, Dctr_clr=0, Dctr_cnt=0 (clear Deg)
- S1: L=0, Dreg_clr=0, Dreg_id=0, Dctr_clr=1 (clearcount), Dctr_cnt=0 (clearcount)
- S2: L=1, Dreg_clr=0, Dreg_id=0, Dctr_clr=0, Dctr_cnt=0 (laser on)
- S3: L=0, Dreg_clr=0, Dreg_id=0, Dctr_clr=0, Dctr_cnt=1 (load Deg with Dctr/2) (count up)
- S4: L=0, Dreg_clr=0, Dreg_id=1 (laser #), Dctr_clr=0, Dctr_cnt=0 (stop counting)

17

Step 4: Deriving the Controller's FSM

Using shorthand of outputs not assigned implicitly assigned 0

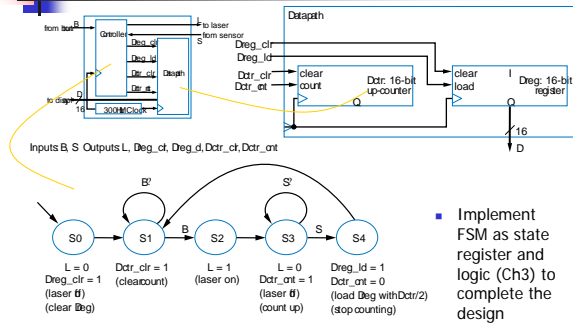
Inputs B (S1 bit each) Outputs L (bit), D (16 bits)
LocalRegisters Dctr (16 bits)

FSM States and Transitions (shorthand):

- S0: L=0, Dreg_clr=1 (laser #), Dreg_id=0, Dctr_clr=0, Dctr_cnt=0 (clear Deg)
- S1: Dctr_clr=1 (clearcount), Dreg_id=0, Dctr_cnt=0 (clearcount)
- S2: L=1 (laser on), Dreg_id=0, Dctr_clr=0, Dctr_cnt=0
- S3: L=0, Dreg_id=1 (laser #), Dctr_clr=0, Dctr_cnt=0 (count up)
- S4: Dreg_id=1 (laser #), Dctr_clr=0, Dctr_cnt=0 (load Deg with Dctr/2) (stop counting)

18

Step 4



19

Design Challenge

Design Challenge

- Design a 4-bit up-counter using the RTL design process.
 - Design a high-level state machine for a 4-bit up-counter with count control input *cnt*, count clear input *clr*, and a terminal count output *tc*.
 - Use the RTL design method to convert the high-level state machine to a controller and a datapath.
 - Use a register and adder in the datapath, not a counter itself.
 - Design the datapath and controller to structure.
- Due:
 - Next Lecture (Monday, October 17)
- Extra Credit (*Homework*)
 - 2 points

20