

ECE 274 Digital Logic – Fall 2008


Introduction to RTL Design

Digital Design 5.1 – 5.2



Digital Design

Chapter 5: RTL Design



Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
<http://www.ddvahid.com>

Copyright © 2007 Frank Vahid

Instructors of courses requiring Vahid's Digital Design textbook (published by John Wiley and Sons) have permission to modify and use these slides for customary course-related activities, subject to keeping this copyright notice in place and unmodified. These slides may be posted as [unannotated pdf](#) versions on publicly-accessible course websites. PowerPoint source (or pdf with animations) may [not](#) be posted to publicly-accessible websites, but may be posted for students on internal protected sites or distributed directly to students by other electronic means. Instructors may make printouts of the slides available to students for a reasonable photocopying charge, without incurring royalties. Any other use requires explicit permission. Instructors may obtain PowerPoint source or obtain special use permissions from Wiley – see <http://www.Abvahid.com> for information.

RTL Design

RTL Design Method

5.2

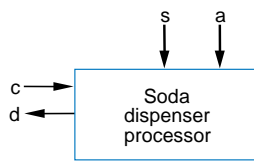
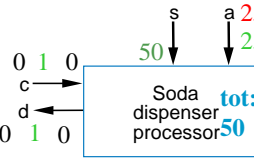
Step	Description
Step 1 <i>Capture a high-level state machine</i>	Describe the system's desired behavior as a high-level state machine. The state machine consists of states and transitions. The state machine is "high-level" because the transition conditions and the state actions are more than just Boolean operations on bit inputs and outputs.
Step 2 <i>Create a datapath</i>	Create a datapath to carry out the data operations of the high-level state machine.
Step 3 <i>Connect the datapath to a controller</i>	Connect the datapath to a controller block. Connect external Boolean inputs and outputs to the controller block.
Step 4 <i>Derive the controller's FSM</i>	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.

3

RTL Design

RTL Design Method: "Preview" Example

- Soda dispenser
 - c : bit input, 1 when coin deposited
 - a : 8-bit input having value of deposited coin
 - s : 8-bit input having cost of a soda
 - d : bit output, processor sets to 1 when total value of deposited coins equals or exceeds cost of a soda

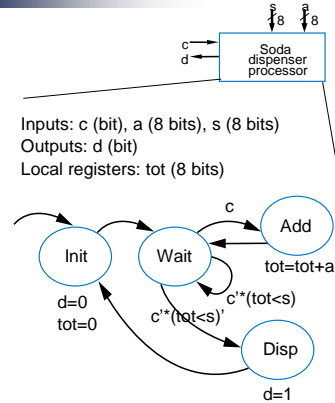
How can we precisely describe this processor's behavior?

4

RTL Design

Preview Example: Step 1 – Capture High-Level State Machine

- Declare local register *tot*
- **Init** state: Set $d=0$, $tot=0$
- **Wait** state: wait for coin
 - If see coin, go to **Add** state
- **Add** state: Update total value: $tot = tot + a$
 - Remember, *a* is present coin's value
 - Go back to **Wait** state
- In **Wait** state, if $tot \geq s$, go to **Disp**(ense) state
- **Disp** state: Set $d=1$ (dispense soda)
 - Return to **Init** state

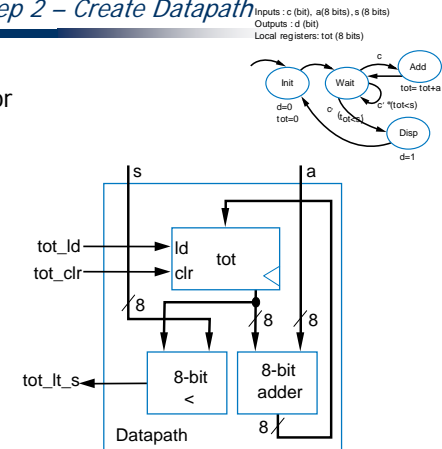


5

RTL Design

Preview Example: Step 2 – Create Datapath

- Need *tot* register
- Need 8-bit comparator to compare s and tot
- Need 8-bit adder to perform $tot = tot + a$
- Wire the components as needed for above
- Create control input/outputs, give them names

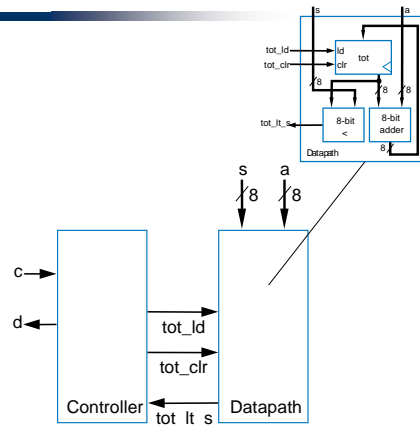


6

RTL Design

Preview Example: Step 3 – Connect Datapath to a Controller

- Controller's inputs
 - External input c (coin detected)
 - Input from datapath comparator's output, which we named tot_lt_s
- Controller's outputs
 - External output d (dispense soda)
 - Outputs to datapath to load and clear the *tot* register

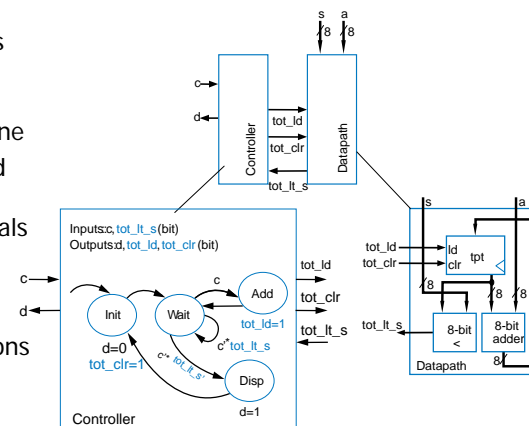


7

RTL Design

Preview Example: Step 4 – Derive the Controller's FSM

- Same states and arcs as high-level state machine
- But set/read datapath control signals for all datapath operations and conditions



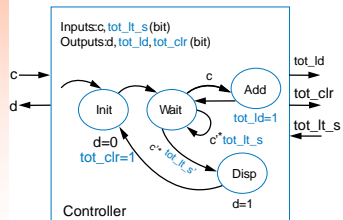
8

RTL Design

Preview Example: Completing the Design

- Implement the FSM as a state register and logic

- As in Ch3
- Table shown on right



	s1	s0	c	s' tot	n1	n0	d	tot_clr	tot_clr
Init	0	0	0	0	0	1	0	0	1
	0	0	0	1	0	1	0	0	1
	0	0	1	0	0	1	0	0	1
	0	0	1	1	0	1	0	0	1
Wait	0	1	0	0	1	1	0	0	0
	0	1	0	1	0	1	0	0	0
	0	1	1	0	1	0	0	0	0
	0	1	1	1	1	0	0	0	0
Add	1	0	0	0	0	1	0	1	0

Disp	1	1	0	0	0	0	1	0	0

9

RTL Design

High-Level State Machine

- Soda dispenser example

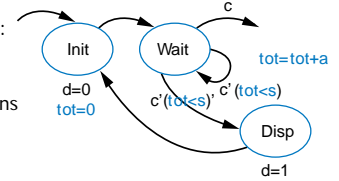
- Not an FSM because:

- Multi-bit (data) inputs *a* and *s*
- Local register *tot*
- Data operations $tot=0$, $tot<s$, $tot=tot+a$.

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

- Useful high-level state machine:

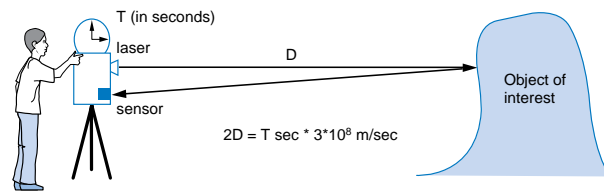
- Data types beyond just bits
- Local registers
- Arithmetic equations/expressions



10

RTL Design

Step 1 Example: Laser-Based Distance Measurer

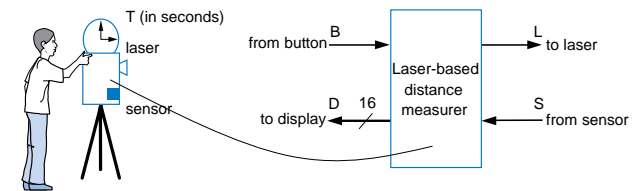


- Example of how to create a high-level state machine to describe desired processor behavior
- Laser-based distance measurement – pulse laser, measure time *T* to sense reflection
 - Laser light travels at speed of light, $3 \cdot 10^8$ m/sec
 - Distance is thus $D = T \text{ sec} \cdot 3 \cdot 10^8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

11

RTL Design

Step 1 Example: Laser-Based Distance Measurer



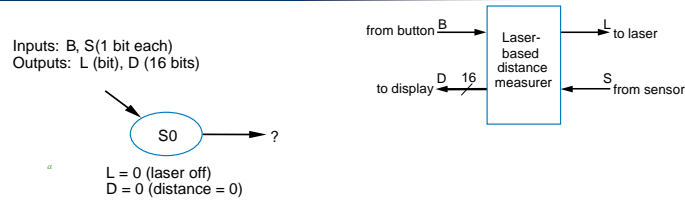
- Inputs/outputs

- 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

12

RTL Design

Step 1 Example: Laser-Based Distance Measurer

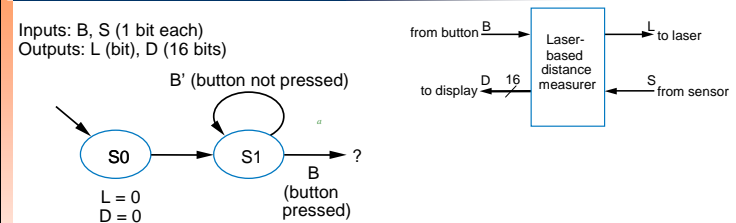


- Step 1: Create high-level state machine
- Begin by declaring inputs and outputs
- Create initial state, name it **S0**
 - Initialize laser to off ($L=0$)
 - Initialize displayed distance to 0 ($D=0$)

13

RTL Design

Step 1 Example: Laser-Based Distance Measurer



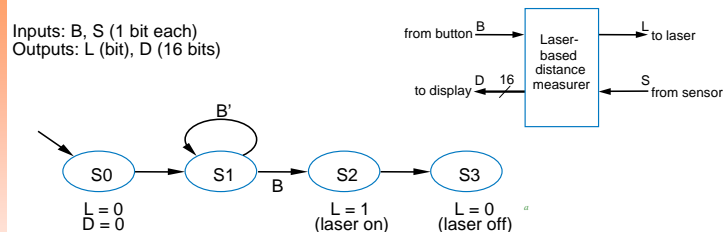
- Add another state, call **S1**, that waits for a button press
 - B' – stay in **S1**, keep waiting
 - B – go to a new state **S2**

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

14

RTL Design

Step 1 Example: Laser-Based Distance Measurer



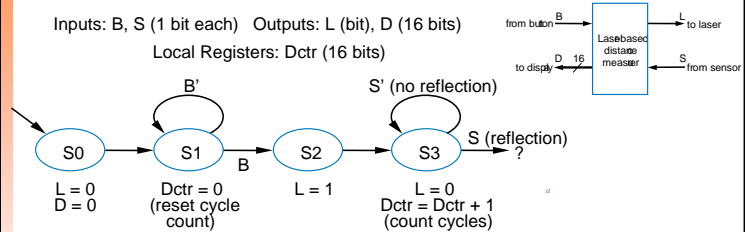
- Add a state **S2** that turns on the laser ($L=1$)
- Then turn off laser ($L=0$) in a state **S3**

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

15

RTL Design

Step 1 Example: Laser-Based Distance Measurer



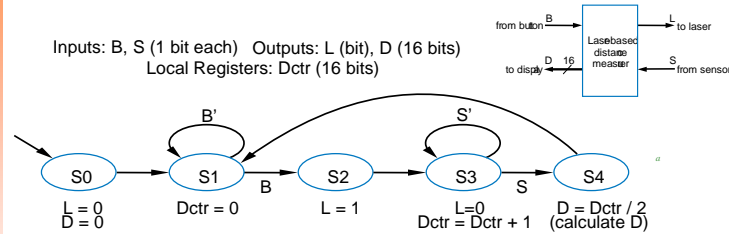
- Stay in **S3** until sense reflection (S)
- To measure time, count cycles for which we are in **S3**
 - To count, declare local register $Dctr$
 - Increment $Dctr$ each cycle in **S3**
 - Initialize $Dctr$ to 0 in **S1**. **S2** would have been O.K. too

16

RTL Design

Step 1 Example: Laser-Based Distance Mesurer

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



- 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

17

RTL Design

Step 2: Create a Datapath

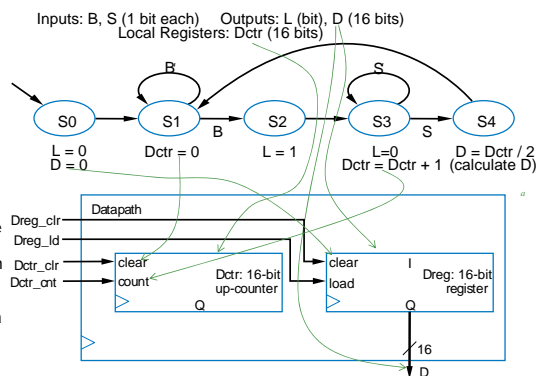
- Datapath must
 - Implement data storage
 - Implement data computations
- Look at high-level state machine, do three substeps
 - (a) Make data inputs/outputs be datapath inputs/outputs
 - (b) Instantiate declared registers into the datapath (also instantiate a register for each data output)
 - (c) Examine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

18

RTL Design

Step 2 Example: Laser-Based Distance Mesurer

- (a) Make data inputs/outputs be datapath inputs/outputs
- (b) Instantiate declared 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

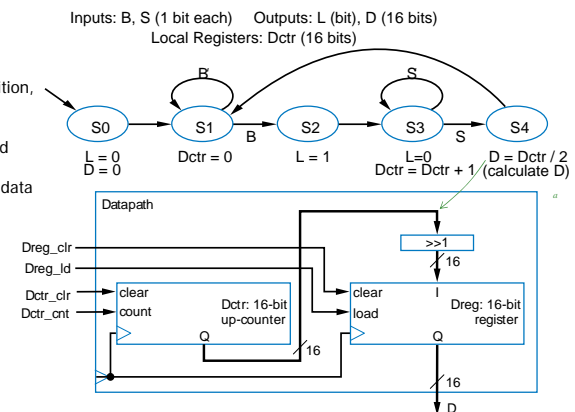


19

RTL Design

Step 2 Example: Laser-Based Distance Mesurer

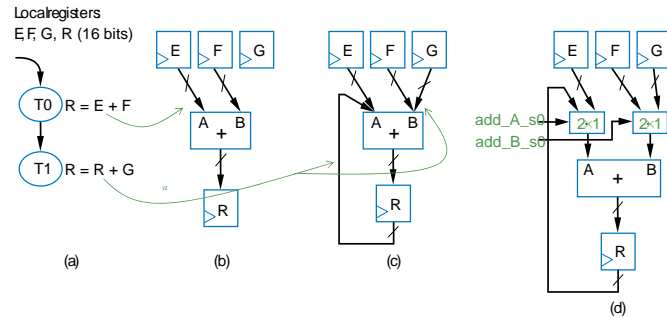
- (c) (continued) Examine every state and transition, and instantiate datapath components and connections to implement any data computations



20

RTL Design

Step 2 Example Showing Mux Use

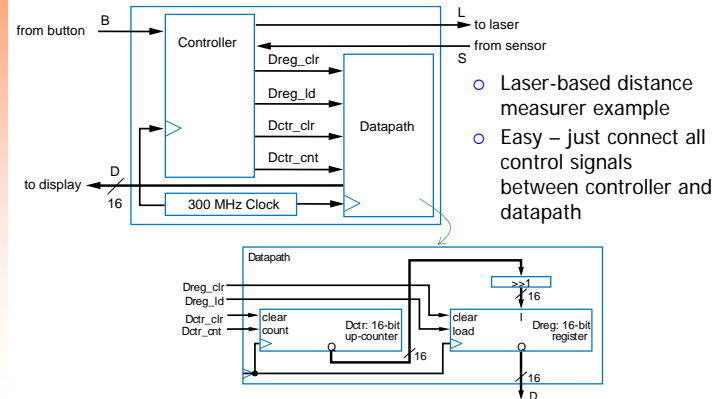


- Introduce mux when one component input can come from more than one source

21

RTL Design

Step 3: Connecting the Datapath to a Controller

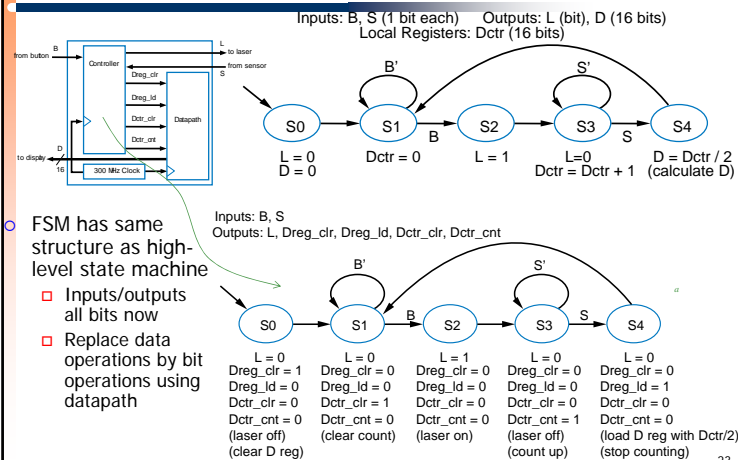


- Laser-based distance measurer example
- Easy – just connect all control signals between controller and datapath

22

RTL Design

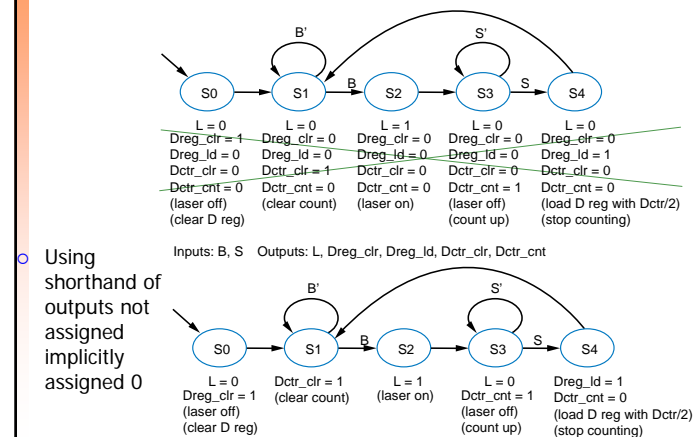
Step 4: Deriving the Controller's FSM



23

RTL Design

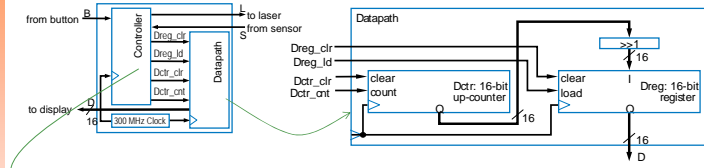
Step 4: Deriving the Controller's FSM



24

RTL Design

Step 4: Deriving the Controller's FSM



Inputs: B, S Outputs: L, Dreg_clr, Dreg_ld, Dctr_clr, Dctr_cnt

