Overview:

**Step 2: Create a Datapath**
- Datapath must:
  - Implement data storage
  - Implement data computations
- Look at high-level state machine, do three substeps:
  1. Make data inputs/outputs be datapath inputs/outputs
  2. Instantiate declared registers into the datapath (also instantiate a register for each data output)
  3. Examine every state and transition, and instantiate datapath components and connections to implement any data computations

**Step 2 Example: Laser-Based Distance Measurer**
- 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

**Step 3: Connecting the Datapath to a Controller**
- Laser-based distance measurer example
- Easy – just connect all control signals between controller and datapath
### 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, S (1 bit each)
- L = 0
- D = 0
- Dctr_cnt = 0
- Dctr_clr = 0
- Dreg_ld = 0
- Dreg_clr = 1

**Outputs:** L (bit), D (16 bits)
- D = Dctr / 2

**Representations:**
- **L = 0** L = 1 L = 0 L = 0
- **D = Dctr / 2** (stop counting: Dctr_cnt = 0)

**Design Examples and Issues**

- We’ll use several more examples to illustrate RTL design

**Example:** Bus Interface
- Master processor can read register from any peripheral
  - Each register has unique 4-bit address
    - Assume 1 register/periph.
  - Sets rd=1, A = address
  - Appropriate peripheral places register data on 32-bit D lines
    - Periph’s address provided on Faddr inputs (maybe from DIP switches, or another register)

**Bus Interface**

- **Master processor**
  - **Part A**
  - **Part B**
  - **Part 15**

- **Peripheral**
  - **Bus Interface**
    - **Faddr inputs:**
      - **Part 0**
      - **Part 4**

**RTL Example: Bus Interface**

- **Inputs:** rd (bit); Q (32 bits); A, Faddr (4 bits)
- **Local register:** Q1 (32 bits)

**States:**
- **WaitMyAddress**
  - Output ‘nothing’ (‘Z’) on D, store peripheral’s register value Q into local register Q1
  - Wait until this peripheral’s address is seen (A=Faddr) and rd=1
- **State SendData**
  - Output Q1 onto D, wait for rd=0 (meaning main processor is done reading the D lines)
Step 1: Create a datapath
(a) Datapath inputs/outputs
(b) Instantiate declared registers
(c) Instantiate datapath components and connections

Step 3: Connect datapath to controller

Step 4: Derive controller’s FSM

Design Challenge

Create a high-level state machine that describes the following system behavior:

- The system has an 8-bit input A, a single-bit input d, and a 32-bit output S.
- On every clock cycle, if d=1, the system should add A to a running sum and output that sum on S.
- If d=0, the system should instead subtract. Ignore issues of overflow and underflow.
- Don’t forget to include an initialization state.
- Hint: declare and use an internal register to keep the sum.

Using the RTL design method, convert the high-level state machine to a datapath and controller.

Due:
- Next Lecture (Wednesday, October 26)

Extra Credit (Homework)
- 2 points.