Finite State Machine with Datapath

Martin Schoeberl

Technical University of Denmark
Embedded Systems Engineering

March 18, 2021

Overview

- Jacob from Syosil presents on verification
- Counter based circuits
- Finite-state machines (FSMs)
- FSM with Datapath

Midterm Evaluation

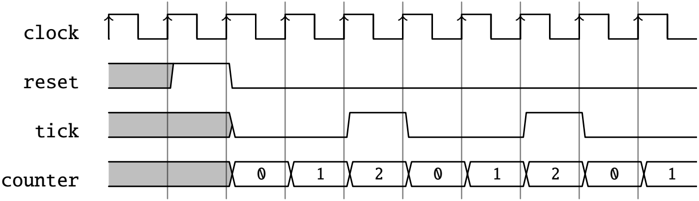
- An anonymous Google form (no login required)
- 15 minutes time during the break
- We will look into it after the break

Last Lab

- Generate a timing and a free running counter
  - Two counters
  - One for the counting from 0 to 15
  - One to generate a *tick* at about 2 Hz
- Did you finish the exercises?
- I will show the solution later

Generating Timing with Counters

- Generate a `tick` at a lower frequency
- We used it in Lab 1 for the blinking LED
- Used for last lab
- Use it for driving the display multiplexing at 1 kHz



The Tick Generation

```
val tickCounterReg = RegInit(0.U(32.W))
val tick = tickCounterReg === (N-1).U

tickCounterReg := tickCounterReg + 1.U
when (tick) {
  tickCounterReg := 0.U
}
```

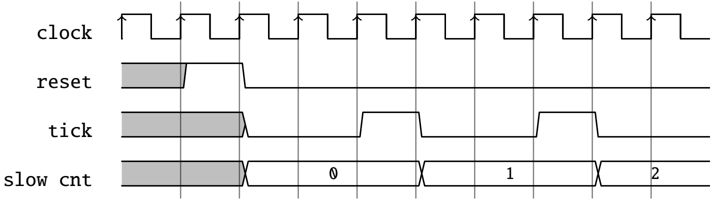
Using the Tick

- A counter running at a *slower frequency*
- By using the `tick` as an enable signal

```
val lowFrequCntReg = RegInit(0.U(4.W))
when (tick) {
  lowFrequCntReg := lowFrequCntReg + 1.U
}
```

The *Slow* Counter

- Incremented every `tick`



What is the Use of This *Slow* Counter?

- This was your lab exercise last week!
- Is a preparation for the display multiplexing (next week)
- Then you need to generate a timing of 1 kHz (1 ms)

One Possible Solution for Last Lab

```
val MAX_CNT = 50000000.U // use a smaller value
   for waveform viewing

val tickCntReg = RegInit(0.U(32.W))
val cntReg = RegInit(0.U(4.W))

val tick = tickCntReg === MAX_CNT
tickCntReg := Mux(tick, 0.U, tickCntReg + 1.U)
when (tick) {
  cntReg := cntReg + 1.U
}

val m = Module(new SevenSegDec())
m.io.in := cntReg
sevSeg := m.io.out
```

A Self-Running Tester

- CountSevenSeg is a self-running circuit
- Has no input
- Needs no stimuli (poke)
- Just run for a few cycles

```
class SevenSegTest(dut: CountSevenSeg) extends
    PeekPokeTester(dut) {
  step(100)
}
```

Call the Tester

- Using here ScalaTest
- Note `Driver.execute`
- Note `Array("--generate-vcd-output", "on")`

```
class SevenSegCountSpec extends
  FlatSpec with Matchers {

  "SevenSegTest " should "pass" in {
      chisel3.iotesters.Driver.execute(
      Array("--generate-vcd-output", "on"),
      () => new CountSevenSeg)
        { c => new SevenSegTest(c)}
        should be (true)
  }
}
```

Running the Test

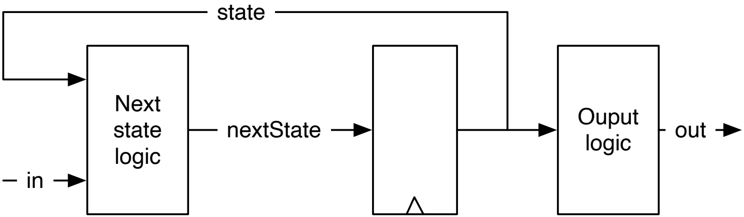
- Does not really do any testing
- Just generated the waveform for debugging
- Just running 100 cycles does not show much
- Increase the number of running cycles to 100000000?
- Or use a different constant for testing?
- Let us explore now
- This issue will be the same for your display multiplexing

Finite-State Machine (FSM)

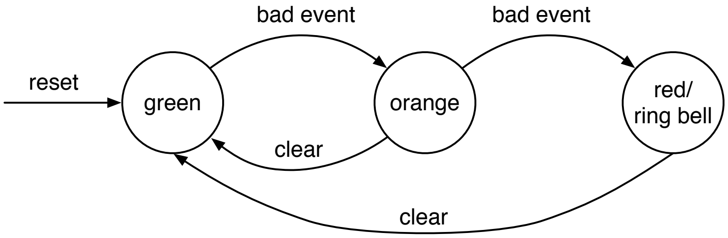
- Has a register that contains the state
- Has a function to computer the next state
  - Depending on current state and input
- Has an output depending on the state
  - And maybe on the input as well
- Every synchronous circuit can be considered a finite state machine
- However, sometimes the state space is a little bit too large

Basic Finite-State Machine

- A state register
- Two combinational blocks
  - Next state logic
  - Output logic



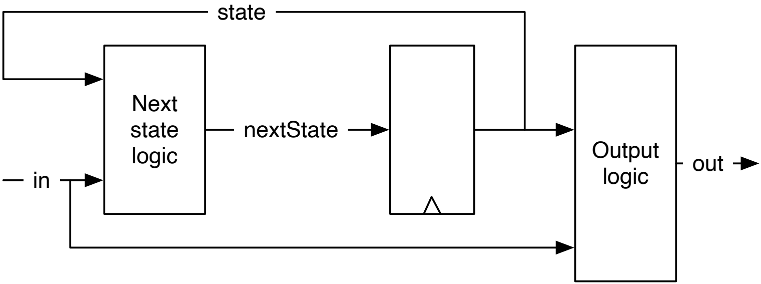
State Diagrams are Convenient



- States and transitions depending on input values
- Example is a simple alarm FSM
- Nice visualization
- Will not work for large FSMs

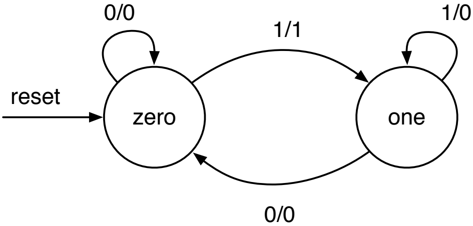
A Mealy FSM

- Similar to the former FSM
- Output also depends in the input
- Output is *faster*
- Less composable as we may have combinational circles



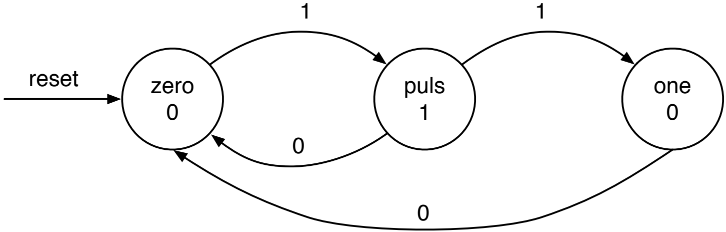
The Mealy FSM for the Rising Edge

- Output is also part of the transition arrows



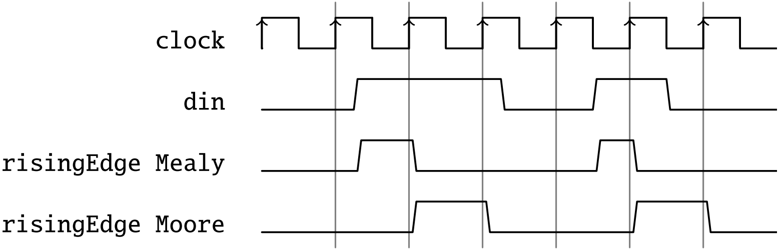
State Diagram for the Moore Rising Edge Detection

- We need three states



Comparing with a Timing Diagram

- Moore is delayed by one clock cycle compared to Mealy



What is Better?

- It depends ;-)
- Moore is on the save side
- Moore is composable
- Mealy has *faster* reaction
- Both are tools in you toolbox
- Keep it simple with your vending machine and use a Moore FSM

FSM with Datapath

- A type of computing machine
- Consists of a finite-state machine (FSM) and a datapath
- The FSM is the master (the controller) of the datapath
- The datapath has computing elements
    - E.g., adder, incrementer, constants, multiplexers, ...
- The datapath has storage elements (registers)
    - E.g., sum of money payed, count of something, ...

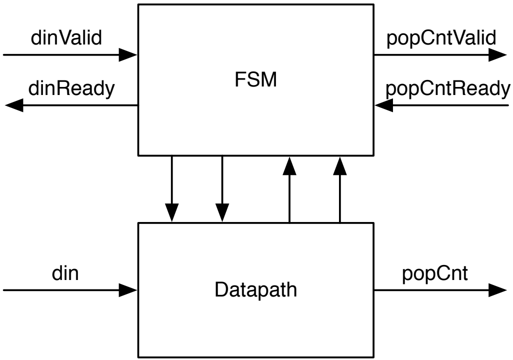
FSM-Datapath Interaction

- The FSM controls the datapath
  - For example, add 2 to the sum
- By controlling multiplexers
  - For example, select how much to add
  - Not adding means selecting 0 to add
- Which value goes where
- The FSM logic also depends on datapath output
  - Is there enough money payed to release a can of soda?
- FSM and datapath interact

Popcount Example

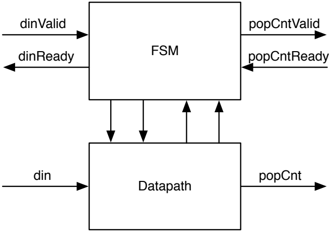
- An FSMD that computes the popcount
- Also called the Hamming weight
- Compute the number of ‘1’s in a word
- Input is the data word
- Output is the count
- Code available at PopCount.scala

Popcount Block Diagram



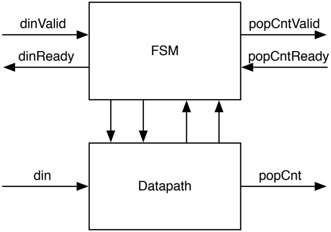
Popcount Connection

- Input `din` and output `popCount`
- Both connected to the datapath
- We need some handshaking
- For data input and for count output

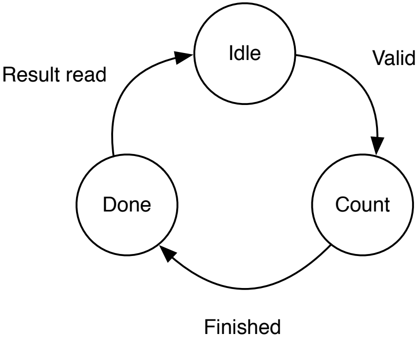


Popcount Handshake

- We use a ready-valid handshake
- When data is available valid is asserted
- When the receiver can accept data ready is asserted
- Transfer takes place when both are asserted

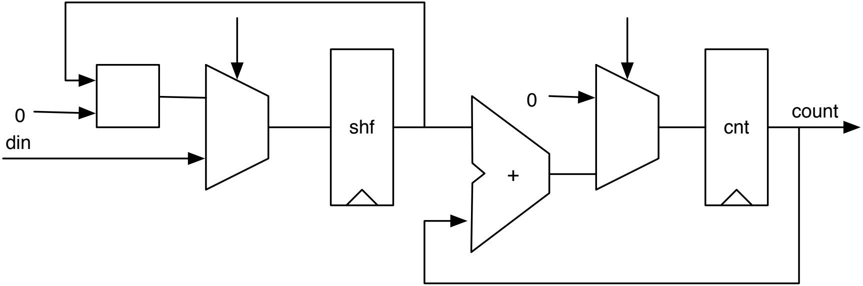


The FSM



- A Very Simple FSM
- Two transitions depend on input/output handshake
- One transition on the datapath output

The Datapath



Let’s Explore the Code

- In PopCount.scala

Usage of an FSMD

- Maybe the main part your vending machine is an FSMD?

Today Lab

- Paper & pencil exercises
- Exercises on FSM
- From the Dally book
- Just sketch the Chisel code
- On paper or in a plain text editor
- As usual, show and discuss with a TA

Summary

- Counters are used to generate timing
- Adapt your counter maximum values for simulation
- An FSM can control a datapath, an FSMD
- An FSMD is a computing machine