Arnav Goyal

Digital Design

Self Study

Preface

The purpose of this document is to act as a comprehensive note for my understanding on the subject matter. I may also use references aside from the lecture material to further develop my understanding, and these references will be listed here.

This document should eventually serve as a standalone reference for learning or review of the subject matter. There is also a lot of organization within these documents, please refer to the table of contents within your PDF viewer for ease of navigation.

References

- Provided Course Materials from ECE 2277, ECE 3375
- Digital Design with an Introduction to the Verilog HDL - 5e - M. Mano, D. Ciletti
- Verilog Complete Tutorial - VLSI Point (YouTube Link)

Finite State Machines

A **finite state machine** or[1] (**FSM**) is a system that can be in a finite number of **states**, accept a finite set of **inputs**, producing a finite state of **outputs**.

[1] Sometimes an FSM is called a Finite State Automaton

- Listing these states, the possible transitions between states based on the input, and the conditions required for each possible output, provides a complete function description of an FSM.

A FSM is a general concept, a mathematical concept. However, when dealing with a specific subset of then called **deterministic FSMs**, we can actually implement them with sequential circuits.

- An FSM is **deterministic** if every combination of current state and input results in only one transition (no probabilistic transitions).

Mathematical Formalism

There are two types of FSMs. A **Mealy** and a **Moore** FSM.

- The output of a Moore FSM depends on only the state it is currently in.
- The output of a Mealy FSM depends on transitions between states.

Lets make things more organized by introducing some basic mathematical notation to this.

- Let the set of all possible states in an FSM be denoted by Q, lets also call the starting state Q_0
- Let the set of all possible inputs be called I
- Let the transition function be denoted $\delta(Q, I)$, it determines the next state as a function of the current state and next input, we say: $\delta : Q \times I \mapsto Q$
- Let the output function be given by $f(\cdot)$, it determines the output as a function of something.

This means that any FSM M can be described by the following basic description[2]

$$\mathrm{M} = (Q, I, Q_0, \delta, f)$$

[2] Here the functions are written as variables for clarity, we also probably cant define these functions in any way other than a truth/state table. For now just assume that they can.

We can differentiate mathematically between Mealy and Moore FSMs through the description of their output functions. Let the set O be the set of all possible outputs.

- A Moore FSMs output function is defined[3] as $f : Q \mapsto O$.
- A Mealy FSMs output function is defined[4] as $f : \delta \mapsto O$

[3] A Moore FSMs output depends ONLY on its current state

[4] A Mealy FSMs current output depends on the SPECIFIC transition it is experiencing

Clock Domain Crossing

Clock Domains & Metastability

So far we've only discussed **synchronous** circuits[5]. In the real world, it is extremely common to have **asynchronous** circuits[6]. Consider some circuit C with two design blocks $C_1, C_2 \subseteq C$

- we say that the circuit is synchronous if the only clock within C is f_1 or derivatives[7] of f_1
- we say that the circuit is asynchronous if there are multiple clocks - if C_1 is clocked by f_1, and C_2 clocked by f_2

A problem occurs[8] when trying to send data from C_1 to C_2, we are trying to send data at some frequency f_1, and trying to sample it on some frequency f_2. Sending data through different clock domains like this is called **clock domain crossing** or (**CDC**)

Whenever this occurs, the data we are trying to send has the possibility of becoming **metastable**.

- When a signal is metastable, it means that it isn't a well defined 0 or 1.

[5] circuits that share the same clock.

[6] circuits with multiple clocks

[7] Any clocks with constant phase relationships, such as $f_1/2, f_1/4$, etc.

[8] This is the problem with Clock Domain Crossing (CDC)

- Metastability arises from violating the **setup and hold times**[9] of a flip-flop.

Crossing Clock Domains

When crossing clock domains, metastability arises, it is not possible to completely remove metastability from a design, but with the proper procedures we can reduce the chances of it happening.

- A measure of metastability can be approximated through the notion of **mean time between failures** also (**MBTF**)[10]
- We *deal* with metastability by increasing the MBTF as much as we can through the use of some **CDC techniques**.
- A metastable signal entering a flip flop will either produce a metastable output, or a well defined 0 or 1. There is no way to tell, as this is probabilistic in nature.

Now we will learn the most common techniques to deal with CDC

- The *n*-flop synchronizer
- Handshake Synchronizer[11]
- Asynchronous FIFO (Queue)[12]

These are extremely complex circuits, so we will only be learning about the first method.

[9] Setup time t_s refers to the amount of time the input must be stable before the clock edge arrives. Hold time t_h refers to the amount of time the input must be stable after the clock edge arrives

[10] This is an estimate due to the probabilistic nature of metastability. The below equation is provided by cadence's CDC paper

$$\text{MBTF} = \frac{\exp\left(k_1 t_{\text{meta}}\right)}{k_2 \cdot f_{\text{clk}} \cdot f_{\text{data}}}$$

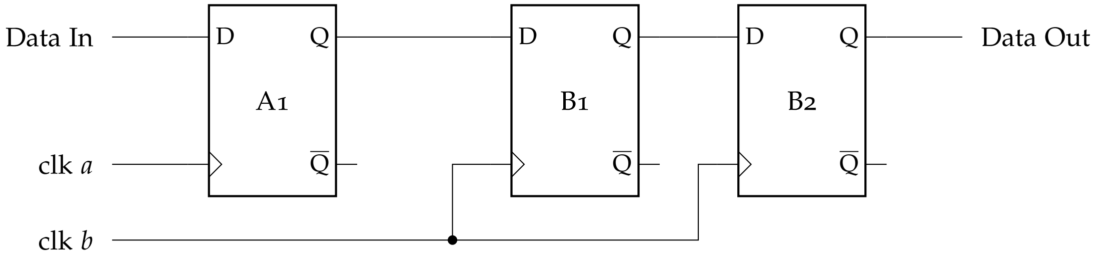
[11] works through a 'request' and 'acknowledge' mechanism

[12] tx writes into the queue, and rx reads from the queue.

n-Flop Synchronizer

The most common way to deal with CDC is to construct an **n-flop synchronizer**[13]. The circuit of 2-flop synchronizer looks like this.

[13] usually n is 2 or 3



It's easy to see how this circuit can be arbitrarily extended to contain n-flops. Adding a flop flop to the chain increased MBTF, but also adds latency to our design.

- This design works because even if the data becomes metastable after propagating through B1, it has another clock cycle to settle and become a well defined 0 or 1 before propagating through B2.
- Each additional flip-flop in the chain gives the (possibly) metastable signal more time to settle before being sent out.

Reset Domain Crossing

Reset Domains & Metastability

Consider a circuit with many different flip-flops. We define a **reset domain** as the set of any flip flops that share the same reset signal *rstn*.

- The reset signal guarantees that the data output is 0 during its **assertion**.
- Metastability can only occur during the **deassertion** of a reset signal.