ECE271, Chapter 3 Reading Report

WeiHao Kuang

April 30, 2018

1 Chapter Outline

1. Introduction

   In this section the main focus will be on sequential logic; unlike combinational logic, sequential logic requires memory to run because the output depends on the current and previous inputs. Sequential logic is usually made up of other parts that are simpler to understand: such as latches and flip-flops. The reason that simpler parts are used is because sequential circuits are much harder analyze since the output changes based on a number of variables. One of the common ways to simplify the design process would be to add combinational logic circuits and a bunch of flip-flops that will help contain the state of the circuit, since the combinational logic is simple to analyze and one flip-flop only store one-bit of state information. State machines can also be used to help design sequential logic circuits.

2. Latches and Flip-Flops

   There are many types of simple sequential circuits SR: latches, D latches, D flip-flops, enabled flip-flops, resettable flip-flops. Enable and resettable flip-flops are made by using different configurations of D flip-flops, incorporating extra inputs produce different characteristics; Enable flip-flops determine whether data is on when a signal is send to the flip-flop, changes if there is a new input, and recycles old input if the enable is "off", and resettable flip-flops are useful when we need to force a known value into the circuit when it powers on. SR latches are composed of two cross coupled NOR gates the greatest shortcoming of this sequential circuit is that it ceases to function correctly if the two inputs Set and Reset are TRUE, on the other hand the same problem can be avoided if D latches were used. D latches incorporate the time element into the logic therefore guaranteeing that inputs Set and Rest are never asserted at the same time therefore never producing erroneous outputs. Next are the flip-flops which are the most efficient compared to the previous two mentioned earlier, since they on change when they need to change when they need to unlike D latches. When you combine X flip-flops together they are known as registers, registers share on common CLOCK (CLK) input. Together latches and flip-flops become the fundamental building blocks for more complex sequential circuits.

3. Synchronous Logic Design

   There are two types of sequential logic circuits: synchronous and asynchronous circuits, synchronous logic circuits are circuits that only change on the clock edge hence the name synchronous; these circuits effectively eliminate races that are inherent in asynchronous circuits, due to the added time input. On the other hand, asynchronous circuits do not change according to a signal edge and responds directly based on the inputs that the circuit receives. Since the asynchronous circuits do not use the clock as part of the output, the only thing that matters is the propagation delays of the components that are in the circuit.[1] A key trait of synchronous circuits are their generality, they can use any kind of feedback. This makes them very versatile and appealing to use, but in general, synchronous circuits are used more often in the field because they are easier to design compared to the asynchronous circuits.

4. Finite State Machines

   In general there are two types of finite state machines: Moore machines, and Mealy Machines. Moore machine outputs depend only on the state of the machines, whereas the Mealy machine output depends on both the current state and the current inputs. Designing sequential circuits is pretty complicated, but the complexity is easier to manage if a finite state machines (FSM) is used to design so long as the proper procedures are followed, designing isn't an impossible task. On top of design an FSM there are different types of encoding that can be used when designing to make the FSMs more efficient such as one-hot encoding, one-cold encoding, and binary encoding. One-hot as the name suggest only has one bit that is hot or "TRUE", at anytime, and one-cold is the exact opposite of that, since there are less gates are used to encode, decoding time will be shortened as well. Binary encoding is just using binary numbers to represent the number of states that are in the circuit, this form of encoding is reliable and works all the time but is slower and less efficient when compared to one-hot encoding. When designing sequential or other complex circuits FSMs are often used because of the systematic process to break down the more complex of circuits.

5. Timing of Sequential Logic

   The timing of sequential circuits is super important because one mistake can literally cost companies millions of dollars. When designing a sequential circuit; designers have to figure out the Setup time constraints and the Hold time constraints, these time constraints dictate what the maximum and minimum delays are from the combinational logic between flip-flops. The reason the maximum and minimum delays are important in circuit design because these are the parameters that the circuit needs to be designed around. The time violation of the minimum hold delay can cost the company lots of money because the whole circuit has to redesign, since there are no easy fixes compared to a violation of the setup delay. Designers also have to consider clock skews; this is when registers do not receive their signals at the same time due to a number of variables, such as wire length from clocks, or whether a clock is gate or not. Having too much clock skew will really screw up a circuit and there will be undesirable outputs as a result. The next topic was on synchronizers; synchronizers onto a circuit will guarantee a stable output of 1 or 0, which effectively eliminates metastable states that occur when inputs change within the aperture time of an element. These are all things that designers have to consider; which is why sequential circuit design takes lots of time and money.

6. Parallelism

   There are two types of parallelism, spatial and temporal. Spatial parallelism which is obtained when there are multiple copies of hardware that is used complete multiple instructions that is has been sent. Whereas in temporal parallelism (pipeline) there just one set of hardware but the instructions are broken up into different parts (stages), allowing task overlap. One of the benefits of pipelining is the fact that no extra hardware is needed obtain it, it also allows higher clock speeds since task are simplified and the stages are shortened. However, there are weaknesses in parallelism, one of which completely breaks the process, this is known as dependency, where if current task is dependent on something before it, then the current task will halt until the prior task is completed therefore breaking the parallel working of a set of instructions because this are not working simultaneously anymore.

7. Summary

   Sequential circuits are much more complex than combinational circuits; because of their high complexity, sequential circuits are often broken down into smaller and more simplistic blocks that are easier to design and analyze: latches and flip-flops. One way that we can easily design sequential circuits is through the use of finite state machines (FSM). The importance of timing is one of the main focuses in this chapter; that is because timing can seriously affect designing if it is not considered seriously, which can also end up costing companies millions of dollars to redesign due the violation the Hold delays. Parallelism was also discussed, the advantages and how it can improve system throughput was interesting. Despite, being great at increasing throughput parallelism has one critical flaw that completely breaks it, which is dependency, when a current task depends on a prior task the system of parallelism ceases to function correctly because the task are no longer working simultaneously. The topic that were discussed in this chapter was super useful for discovering the uses and workings of sequential logic and circuits.

2 Grey Box Exploration

1. The first blurb is on page 132, where *An easy way to remember the difference between the two types of finite state machines is that a Moore machine typically has more states than a Mealy machine for a given problem.* The difference in states is not the only thing sets Moore and Mealy machines apart, the fundamental difference between the two types of machines are output dependencies, Moore's machines depends only on the current state of the circuits whereas the Mealy machines depends on the current state and the inputs.[2][3]

   (a) Moore machines have more logic gates, which are used to decode the output signals. Having more logic gates further increase the circuit delays that are inherent in sequential circuits. Moore machines also place their outputs their states, keeping everything a bit safer, because the state only changes on the clock edge. One of the perks with a bulkier and slower circuits like a Moore machine is that it is predictable, if they are designed correctly.[2][3]

   (b) Mealy Machines on the other hand are much faster since they are not synchronous with the clock which means that output changes as soon as input transition occurs, they take less hardware to make because it does not have as many states as Moore machines. Mealy machines are also faster but more unpredictable in a sense that if the delays are not handled correctly there will be erroneous outputs.[2][3]

2. The second blurb is on page 142, this blurb states that there has been a significant increase in microprocessor clock speeds where, *In the three decades from when one of the authors' families bought an Apple II+ computer to the present time of writing, microprocessor clock frequencies have increased from 1 MHz to several GHz, a factor of more than 1000. This speedup partially explains the revolutionary changes computers have made in society.* One interesting things about the CPUs is that they can be over-clocked; the highest CPU clock speed that has been recorded so far is a world record high of 8.772 GHz, with the help of liquid nitrogen cooling. This a very high clock speed but it is not practical because the only way to achieve that clock speed is to use liquid nitrogen cooling, which is bulky and more of an inconvenience than it is worth to everyday people. So lets talk about the highest consumer clock speed, and why it hasn't increase in the last few years. The current speeds for stock consumer CPUs is around 3.8 to the 4.8 range, and it has stayed this way for the past few years, a lot of experts think that this clock speed platea is due to Moore's law which states that the number of transistors will double every year on a CPU. The problems with doubling transistors counts every year is that there will not be enough room and physical limitations such as the atoms size of the silicon will definitely cause issues in the design and fabrication end. So when it really comes down to why clock speeds are not increasing the blame really goes physical limitations, and thermal issues since more transistors equals more heat generated; a trend that I noticed for newer computer components is that they get hotter every year, this is probably due to the increase transistors in every new generation of hardware that is made. [4][5]

3 Figures

Two figures have been chosen from the book because they convey useful information about topics that are covered in chapter 3. Figure 3.22 was selected because it shows what the fundamental difference between Mealy and Moore machines are, which is the output. Moore outputs only depend on state of the circuit, while Mealy machines depend on the inputs and the state of the circuit.

Figure 3.43 was selected for it shows the time graph/diagram for the critical short, and general cases of sequential logic design. I personally think that this part is the hardest to understand and having this diagram really help with describing the timing associated with sequential circuits.

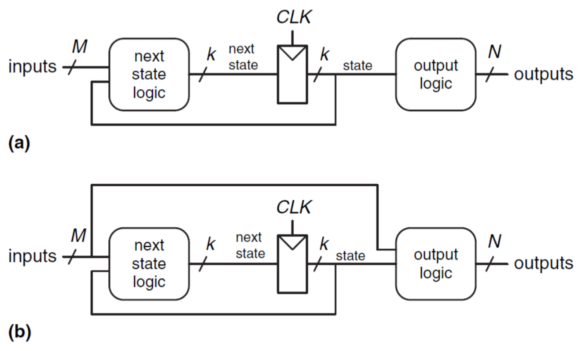


Figure 3.22 **Finite state machines: (a) Moore machine, (b) Mealy machine**

Figure 1: Mealy and Moore abstract circuits

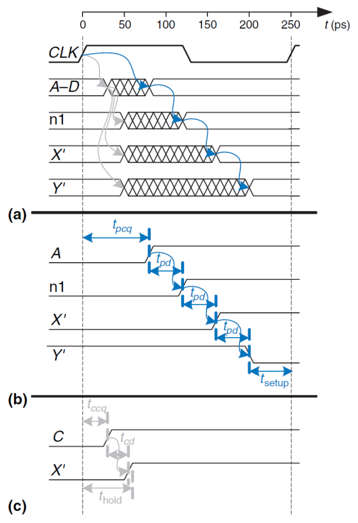


Figure 3.43 **Timing diagram: (a) general case, (b) critical path, (c) short path**

Figure 2: Time graphs/diagram for short, general, and critical paths

4 Example Problems

Example problems are attached at the end

5 Glossary

These terms where all obtained from Google, by using there definition searcher.

1. bistable:

   noun:

   1. an electronic circuit that has two stable states.

   adjective:

   1. (of a system) having two stable states.

2. metastable:

   adjective:

   1. (of a state of equilibrium) stable provided it is subjected to no more than small disturbances.

3. synchronous:

   adjective:

1. existing or occurring at the same time.

2. (of a satellite or its orbit) making or denoting an orbit around the earth or another celestial body in which one revolution is completed in the period taken for the body to rotate about its axis.

4. asynchronous:

   adjective:

   1. (of two or more objects or events) not existing or happening at the same time.

   2. of or requiring a form of computer control timing protocol in which a specific operation begins upon receipt of an indication (signal) that the preceding operation has been completed.

   3.of a machine or motor) not working in time with the alternations of current.

   4. (of a satellite) revolving around the parent planet at a different rate from that at which the planet rotates.

5. encoding:

   verb:

   1. (computing) convert (information or an instruction) into a particular form.

   2. (biology) (of a gene) be responsible for producing (a substance or behavior).

6. aperture:

   noun:

   1. an opening, hole, or gap.

   2. (photography) a space through which light passes in an optical or photographic instrument, especially the variable opening by which light enters a camera.

7. pipeline:

   noun:

   1. a long pipe, typically underground, for conveying oil, gas, etc., over long distances.

   2. a linear sequence of specialized modules used for pipelining.

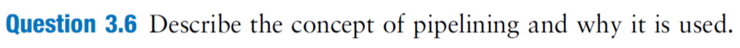
   3. (in surfing) the hollow formed by the breaking of a large wave.

   verb:

   1. convey (a substance) by a pipeline.

   2. design or execute (a computer or instruction) using the technique of pipelining.

6 Interview Question



Question 3.6 Describe the concept of pipelining and why it is used.

Figure 3: Parallelism, the pipeline kind

A pipeline is a type of computer process implementation where there is a continuous and overlapping movement of instructions sent to a processor, where they are split into different stages almost like an assembly line, and each of these stages are connected together forming a pipe that that creates a set of full instructions. Splitting everything into different stages allows for high clock speeds and since there does not need to be hardware doubling it it very appealing when designer are considering implementing pipes in their designs. The reasons that people use pipelining is because its ability to speed up a circuit without duplicating the hardware which also increase the clocked speeds of the circuit.[6][7][8]

Some reasons to not use pipelining is because pipelining often increases the time it takes to complete instructions, it can also require more resources to run compared to a circuit with no pipelining, and lastly pipelining can contribute heavily to latency which is a big reason why it is not used in real-time systems since latency can cause a heap of issues for its users.[6][7][8]

7 Reflection

A lot on things were discussed in this chapter things that are very interesting about sequential circuits. The thing that I was most surprised about was how much more complex and intricate the designing and analyzing process is for sequential circuits versus combinational circuits. It is really cool that more complex of sequential circuits can be made up of smaller and more manageable parts. The reason why I found this chapter very interesting is because these circuits can be controlled using a time variable that affects their behaviors, which is much more exciting than static input and output analysis. When I read through this book some things made a lot sense, such as using smaller and simpler circuits to build more complex circuits: latches and flip-flops. And the timing parameters that designers must meet in order to get their circuits working, but there were a few things that I had trouble understanding at first: which was using finite state machines (FSMs) and using them to create a functional sequential schematic and using a schematic to get a FSM. I re-read the that section, and looked it up online and it started to make more sense to me, I'd figured I should probably bring topic up in class so I can talk about the things I don't get. The next thing I had issues with was the understanding the encoding portion of 3.4, I not really confused about encoding and what it does, but rather which type of encoding should I use when I am building my FSM, one-hot, one-cold, or binary? Is there a set of rules (like FSM designing) that should be followed when considering the type of encoding that I should use? The final topic that was discussed was parallelism, which to me as amazing, I like the fact that circuits can be replicated to do the same job to increase the throughput (in spatial parallelism), but what fascinated me the most was temporal parallelism aka pipelining. Pipelining is interesting because it is parallelism at its most efficient since there doesn't need a second set of hardware to get the parallelism characteristics. Other than the issues I talk about I thought the chapter was a good insight into sequential logic and I look forward into applying the things I learned in labs and/or classes.

8 Questions for Lecture

1. Moore and Mealy machines are the dominate FSMs? Are they different types of FSMs that are not describe in the book, if so are they more efficient or less.
2. Could you analyze and example of a sequential circuit that would be a question on the final exam or a future quiz?
3. Is there a way to solve the dependency issue with parallelism that was discussed in 3.6

References

[1] S. UK, "Asynchronous vs. synchronous." http://www.ee.surrey.ac.uk/Projects/CAL/seq-switching/synchronous_and_asynchronous_cir.htm, 2010.

[2] T. Point, "Moore and mealy machines." https://www.tutorialspoint.com/automata_theory/moore_and_mealy_machines.htm.

[3] Sidhartha, "Mealy vs. moore machine." http://www.vlsifacts.com/mealy-vs-moore-machine/, 2016.

[4] HWBOT, "Cpu frequency hall of fame." https://hwbot.org/benchmark/cpu_frequency/halloffame, 2018.

[5] A. Fox, "Why cpu clock speed isn't increasing." https://www.maketecheasier.com/why-cpu-clock-speed-isnt-increasing/, 2018.

[6] G. PRABHU, "Pipelining." http://web.cs.iastate.edu/~prabhu/Tutorial/PIPELINE/pipe_title.html, 2017.

[7] Admin, "Pipelining and it pros and cons." http://hwinterview.com/index.php/2016/09/18/pipelining-pros-cons/, 2016.

[8] R. H. Ramon Centeno-Colon, "What is pipelining." https://whatis.techtarget.com/definition/pipelining, 2005.

Example 3.1 D-Flip Flop transistor count.

DFlip-Flop = 2 D latches = (2 AND + SR latch + 1 Not)2
= (2AND + 2 NOR + 1 Not)2

$$2\begin{pmatrix} \text{AND} = 6 = 2\text{AND} = 12 \\ \text{Nor} = 4 = 2\text{Nor} = 8 \\ \text{Not} = 2 = 2 \end{pmatrix} \Rightarrow \begin{matrix} 24 \\ 16 \\ 4 \end{matrix} \rightarrow \boxed{\text{44 transistors in a D flip flop.}}$$

Variation: How many Transistors are in a D latch.

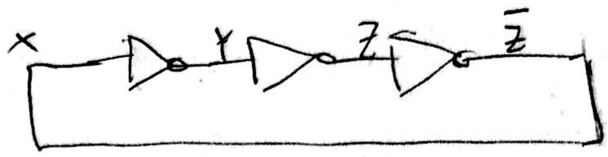
D latch = 2AND + SR latch + 1Not = 2AND + 2Nor + 1Not

$$\left.\begin{matrix} 2\text{AND} = 12 \text{ transistors} \\ 2\text{Nor} = 8 \text{ transistors} \\ 1\text{Not} = 2 \text{ transistors} \end{matrix}\right\rangle \boxed{\text{22 transistors}}$$

In this problem all I did was look at the schematic of the circuits and counted the gates on them and put them into an equation and just expanded until I had gates that I knew the transistor count to and then I just added them up.

Example 7.3. Astab circuits.

Three inverter loop.



These loops are essentially sequential circuits that do not take an input its completely self sufficient. The characteristic of this circuit is known as a ring oscillator. That is be X and $\bar{Z}$ do not equal each other, ever.

If $x = 0$, $y = 1$, $z = 1$, $\bar{z} = 0$

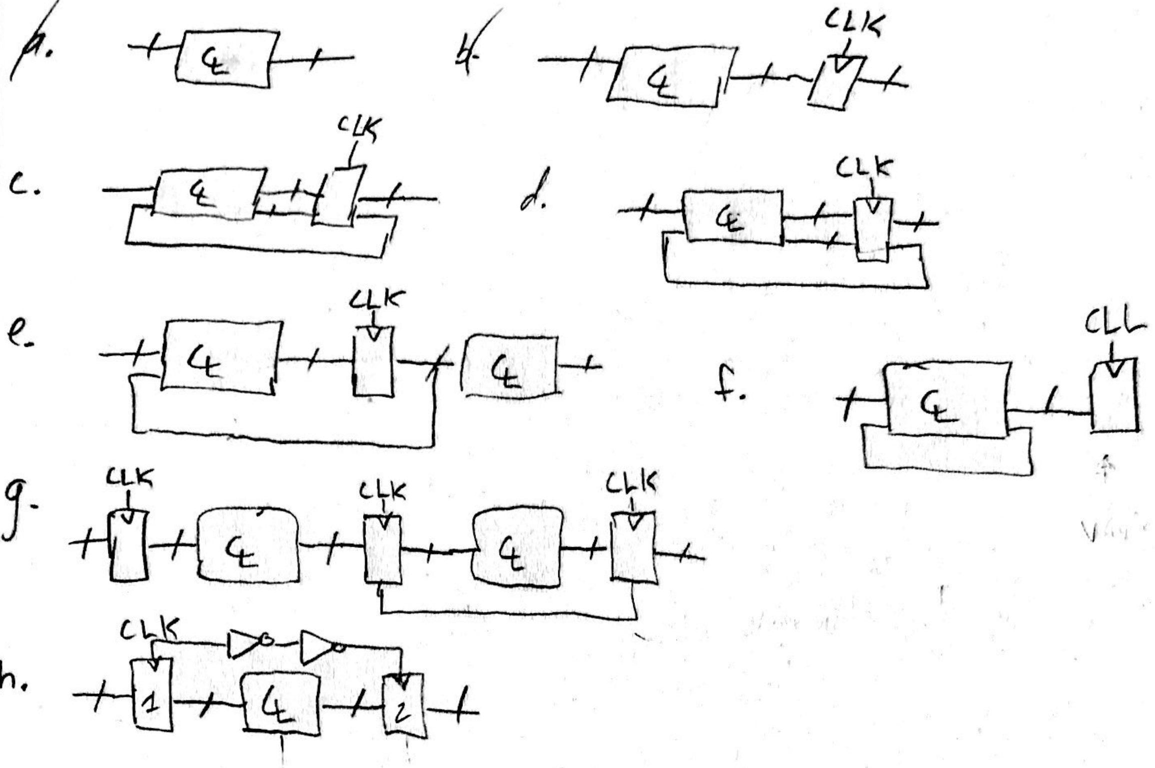
$x \neq \bar{z}$

If $x = 1$, $y = 0$, $z = 1$, $\bar{z} = 0$

$x \neq \bar{z}$.

The circuit will produce the same results after the period it takes to go through all three of the inverters. since the pattern repeats that is why its called an oscillator.

Example 3.5: Which of the circuits are a-h



The only circuits above that are synchronous are d and e. because they are both Finite state machines, because there is a cylic path that connects the registers to the combinatonal logics.

a-b) are not synchronous because they lack a cylic path connecting the combinatonal logic to the register.

c). is not synchonous because the symbol for a register is wrong, the cyclic path connects to a non register or combinational.

f. The cylic path is redundant and doesn't connect to the register available

g. lacks cylic paths connecting the first combinational logic to other registers

h. the registers 1 and 2 don't receive the same signal because the buffer changes what 2 gets.

Scanned with CamScanner

Example 3.12 Timing analysis (clock skew) / find period and freq.

Assume that a circuit with a short path of 55ps, tsetup of 50ps and a 3tpd of 120ps and a tcq of 80ps, and a hold time of 60ps is introduced to a clock skew of 70ps.

Then the equation we would use to find Tc is:

$T_c \geq t_{pcq} + 3t_{pd} + t_{setup} + t_{skew}$

$T_c = 80 + 120 + 50 + 70 = \boxed{320\,ps}$

$\text{Frequency} = 1/T_c = 1/320\,ps = \boxed{3.125\,GHz}$

clock skew effectively changes the setup and hold delays that is why the period function has tskew in it in order to correctly calculate the period. Also I noticed that the higher the skew the lower the resulting frequency clock, which would be bad if there frequency limit that circuit needs to follow.

since clock skew affects hold times the new hold time of the circuit will be $\boxed{60 + 70 = 130\,ps}$.

skew in circuit often create huge problems for staying within the hold and setup delay parameters.

Example 3.15 Cookie throughput & latency.
(variation).

It takes 5 minuts to make a batch of cookies to put into a tray. and then 10 minuts to bake them. The process can be repeated. What is the througput and lateny for this cookie baking setup?

to make a tray of cooked cookies it take 15 minutes with is also $\frac{1}{4}$ hour, meaning in $\frac{1}{4}$ hour per batch of cookies can be made. (latency).

The throughput will be how much batchs can be made in one hour.

$$1 \text{ batch} = \frac{1}{4} \text{ hour}$$

$$4 \text{ batch} = 1 \text{ hour}.$$

So in 1 hour 4 batchs of cookies can be made (throughputs)

using fractions and a little bit of time calculation throughput and latency of a simple system above can be easily calculated. The same logic applies to larger systems it just that there will be more calculation need to get the results.

3.16 Cookie paralism spatial vs pipelining

spatial parallism - 2 people are making cookie same speed and time, time it take cookie batch is 10 minuts and baking is 20 minuts.
↳ two people simentaneously make cookies, (-2 cookie trays 2 ovens)

latency for one person

$\frac{1}{2}$ hour per batch

Through put:

2 batches per hours

since two people are doing the same thing that through put is doubled.

Through put of 2 people $\boxed{\text{4 batches an hour}}$

with pipelining: only one oven and 2 cookie trays are used.

pipeling effective get rid of the 10 minutes it take to match the raw cookie dough. therefor the time it take to make an batch is 20 minutes, latency = $\frac{1}{3}$ hour

through put = 3 batchs/hours

→ when it come down to it spatial parallelism faster and produces more, but the problem is that in order to increase through put we need to double the equipement used. which may be a deal breaker if size and money is an issue designer have to consider

on the other hand, pipeling had more through put and the only thing that need to be added was an extra set of hands pipelining is appealling when size and money is a factor in design.