Lab 4 Report

Michael Cooke

October 2021

Contents

1 Part A

1.1 Bus Wiring

1.1.1 Summary

For this task we have to convert all previous work over to bus wiring. As I looked ahead I have most of the stuff implemented with busses already.

1.1.2 Proof

This can be seen right the way through lab 3, for example:
QuartusFiles/FPGABCD
QuartusFiles/HexToSSDBus
QuartusFiles/Sept4x1Multiplexer

1.2 Digit Shifting

1.2.1 Summary

For this task we had to use bus wiring to enable the shifting of digits. For this I used a select line and switched between four different Sept4x1Multiplexer instances to move the digit.

1.2.2 Difficulties

It took a little bit to get my head around how to implement this, as I had do decide as where I wanted to the shifting. Additionally, I had a space in the name of the file which resulted in a compile error.

1.2.3 Proof

I hooked it up to my lab 3 top level in:
QuartusFiles/FPGAShifting
QuartusFiles/DigitShifter
SupportingVideo/Shifting

1.3 IP Catalog

1.3.1 Summary

Use `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/catalogs/lpm.pdf` to learn about LPM constant and LPM counter. For this task I removed the hex converters and used LPM constant to directly write to the a->g segments of the LED's on the board. As I had already been using LPM counter I didn't have to make any changes to tick this box.

1.3.2 Difficulties

SupportingVideo/Binary
Unfortunately, I can't write the number in binary and hence had to convert to decimal (57 92 92 6).

1.3.3 Proof

SupportingVideo/Constants
QuartusFiles/FPGALPM

2 Part B

2.1 Debouncer and Clock Divider

2.1.1 Summary

When a button is pressed it will mechanically bounce, this will cause it to activate multiple times after the initial release. Until this point it hasn't been an issue as I had only been using buttons for reset and held values. Additionally, in lab 3 I exclusively used the clock to increment values and hence avoided the issue.
To solve this a couple flip-flops are used to provide a time buffer that has to be overcome to trigger a response. Additionally, as part of this we have to implement a clock divider, but I had already done this for lab 3 with LPM counter.

2.1.2 Proof

Hooked it up to lab 3: SupportingVideo/Debouncer
QuartusFiles/Debouncer
QuartusFiles/FPGADebouncer
Note: The LPM counter is within the FourDigitSSD

2.2 Start Sequence

2.2.1 Summary

For this task we had to design a state machine that would recognise a sequence (using all four buttons) and initiate the clocked counting.

Note: This task was completed semi-simultaneously with the select sequence task as they are closely linked.

My first idea:

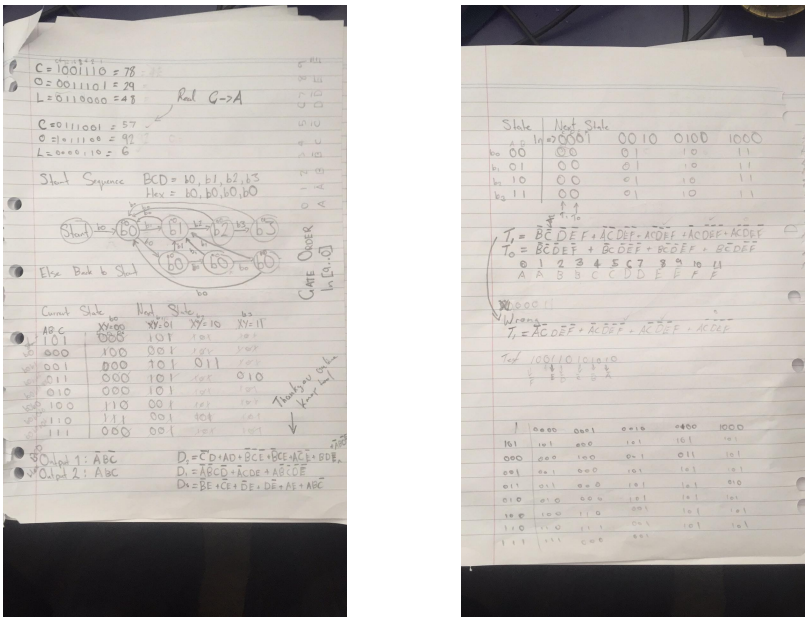

Figure 1: DLatch Recogniser and State Holder

Realised this is better with done via:

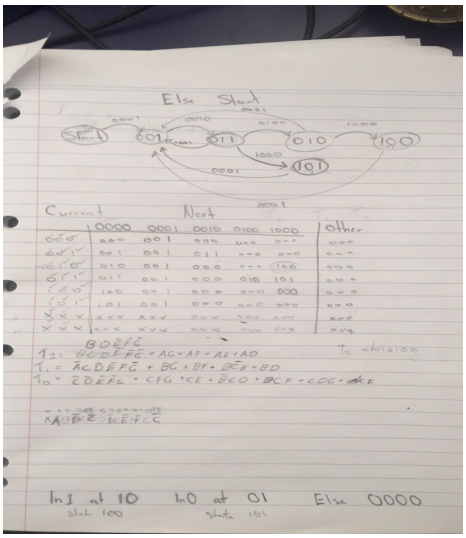

Figure 2: TFlipFlop recogniser

Unfortunately, I had already started to implement the first solution in bdf and had to restart.

I ended up coding this part in Verilog as I was getting sick of drawing so many gates and taking sooo much time to name bus outputs. I did try VHDL first, but sequential circuits are easier to convert to Verilog because a T flip flop can be written:

```
always@(posedge Clock)
begin
        STATE <= STATE ^ INPUT;
end
```

I tried to keep the formatting the same as that of the bdf's that I had written to keep consistency.

2.2.2 Difficulties

I had some trouble naming my DLatch as the name initially matched that of the primitive which caused conflicts. Also I made two mistakes when doing the kmap data entry that had to be fixed, but `https://www.charlie-coleman.com/experiments/kmap/` is a great tool that saved me a lot of time.

2.2.3 Proof

QuartusFiles/EdgeDetector

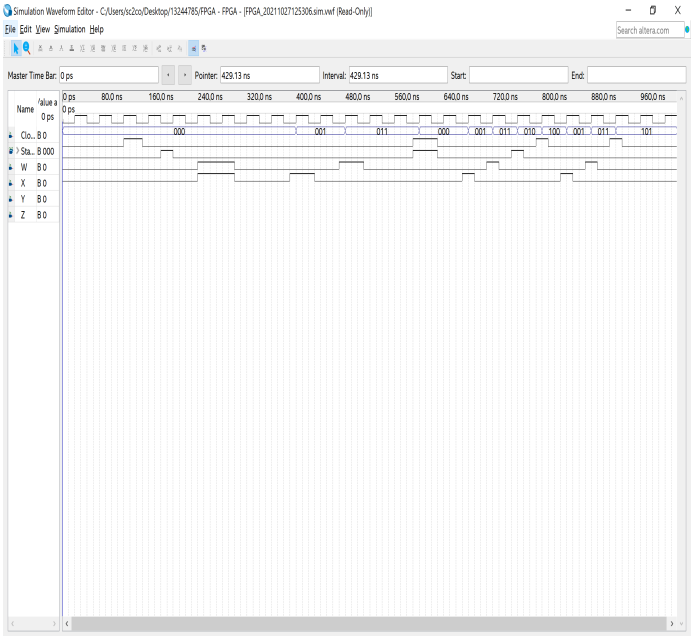

Figure 3: Before Edge Trigger

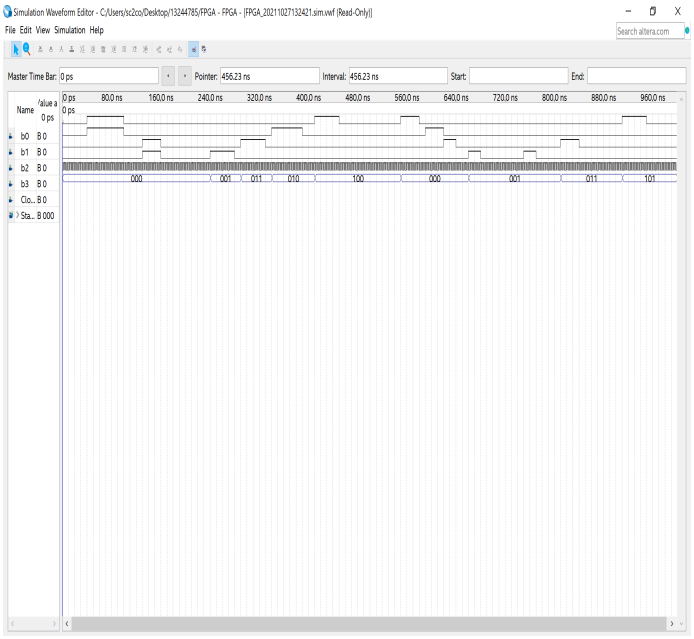

Figure 4: Edge Triggered

2.3 Select Sequence

2.3.1 Summary

For this task we had to extend the previous state machine to have two passwords, each one triggering a different type of counting. To do this, I needed to make sure that the buttons delivered a pulse that was equal to the clock length so that the state wasn't automatically reset in the next clock cycle. I could have made the states loop when the same key was pressed, but this wasn't the functionality I was seeking.

2.3.2 Difficulties

Looping was initially an issue with my first idea for a sequence therefore I rewrote my sequence as I had already spent so long on this Lab. Luckily, the implementation of the selector was straightforward, but initially I had an issue with the edge trigger when introducing it to the rest of the system.

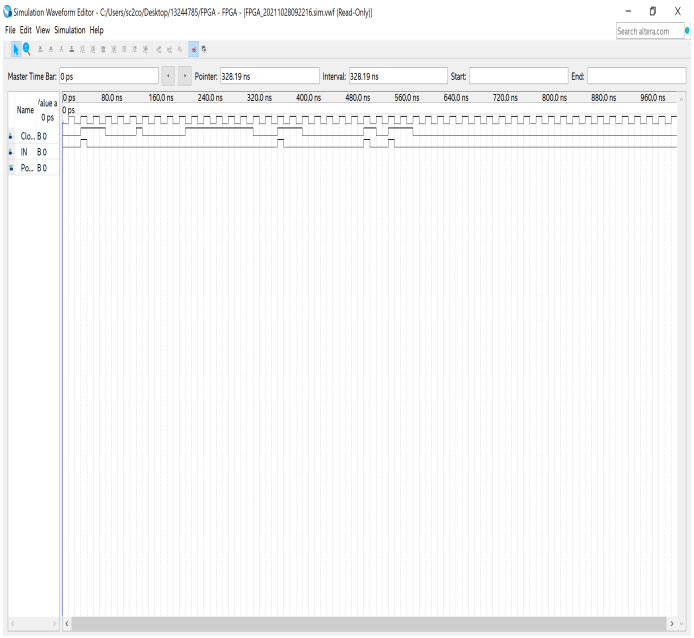

Figure 5: Edge Problem

Essentially, the edge would only be detected when it coincided with a clock rise, I fixed adding a not gate such that the edge was detected when it did not coincided with a clock rise. This was a problem as although the debouncer only sends on a clock rise there is gate delay between the two.

2.3.3 Proof

There is a lot going on in this next waveform so here is a line by line summary:

- The clock is set to 10ns
- bo -> b3 are the buttons that are active low with a dummy input before the two password sequences
- A bus shows the debounced inputs
- A pulse of clock_length is generated on the debounced rising edge
- This corresponds to a state change within the sequence recogniser
- In states 100 and 101 the clock starts counting
- This is converted to SSD
- dis0 -> dis3 are the segment that is active

It is important to note that I had to increase the count rate so that it would show in the simulation, but I didn't increase the multiplexing rate which is why the first display is always active

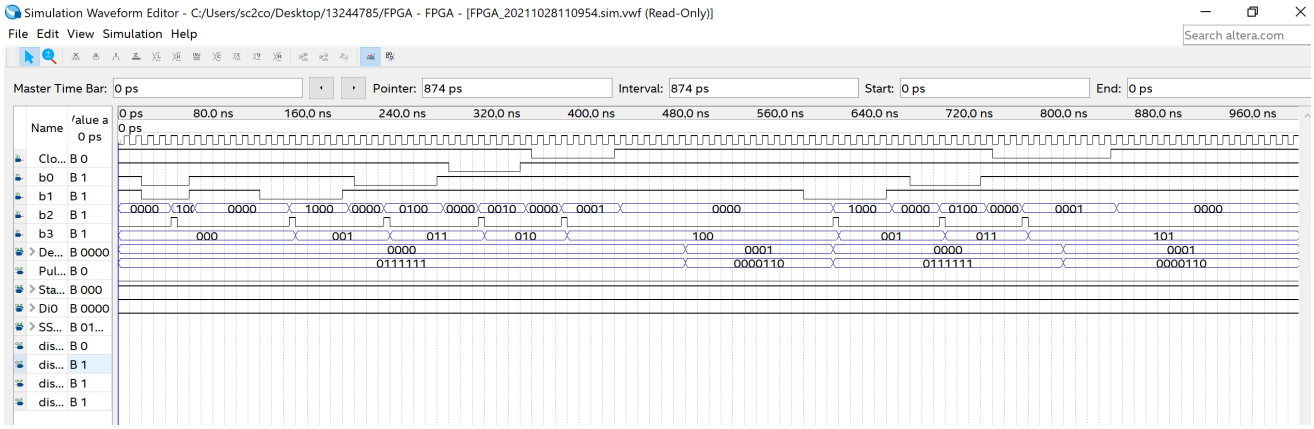

Figure 6: Working

QuartusFiles/FPGAPassword
SupportingVideo/PasswordWorking
NOTE: The flashing points are because of the test outputs that are not assigned pins, deleting them or assigning them to GND will prevent this.

3 System I/O

Parameter	Min	Max	Typical	Units	Description
VDD	4.5	5.5	5	V	Power
B0	0	1	1	V/logic	Button S1
B1	0	1	1	V/logic	Button S2
B2	0	1	1	V/logic	Button S3
B3	0	1	1	V/logic	Button S4
Cathode[6..0]	0000000	1111111	1111110	V/logic	Segments to light up
Anode[3..0]	-	-	-	V/logic	Display Active (greycode)

Table 1: System Inputs and Outputs

4 Use Case

When the four buttons are pressed in the right order the clock will begin to increment from 0 in either decimal or hexadecimal. To count in decimal press the buttons S1,S2,S3,S4. To count in hexadecimal enter S1,S2,S3 instead. The default state is all LED's displaying 0 and the system needs to be powered by 5V via the USB connection or a battery.