Lab 7 Report

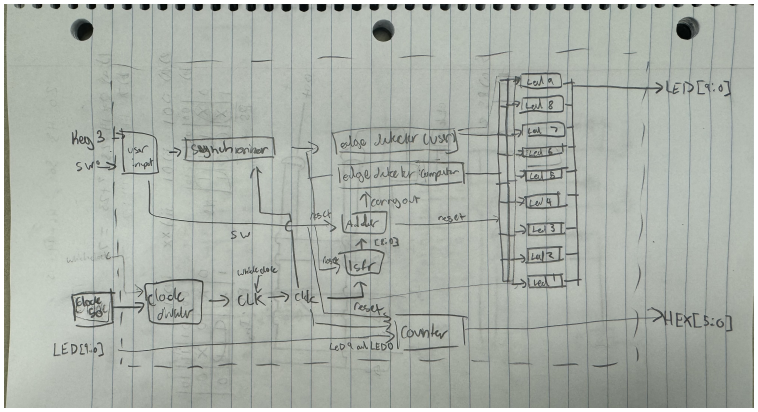
Sathvik Kanuri

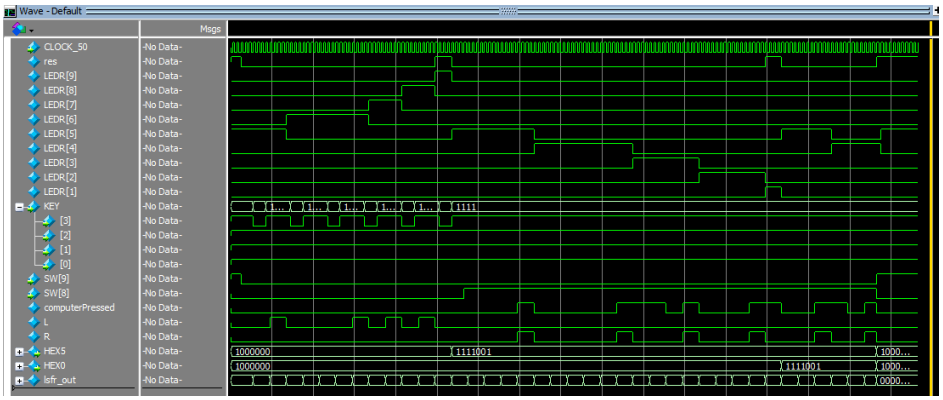
CSE 369: Intro to Digital Design

November 20 2024

Top-Level Block Diagram

Question:
The top-level block diagram, showing the major modules and how they are interconnected.





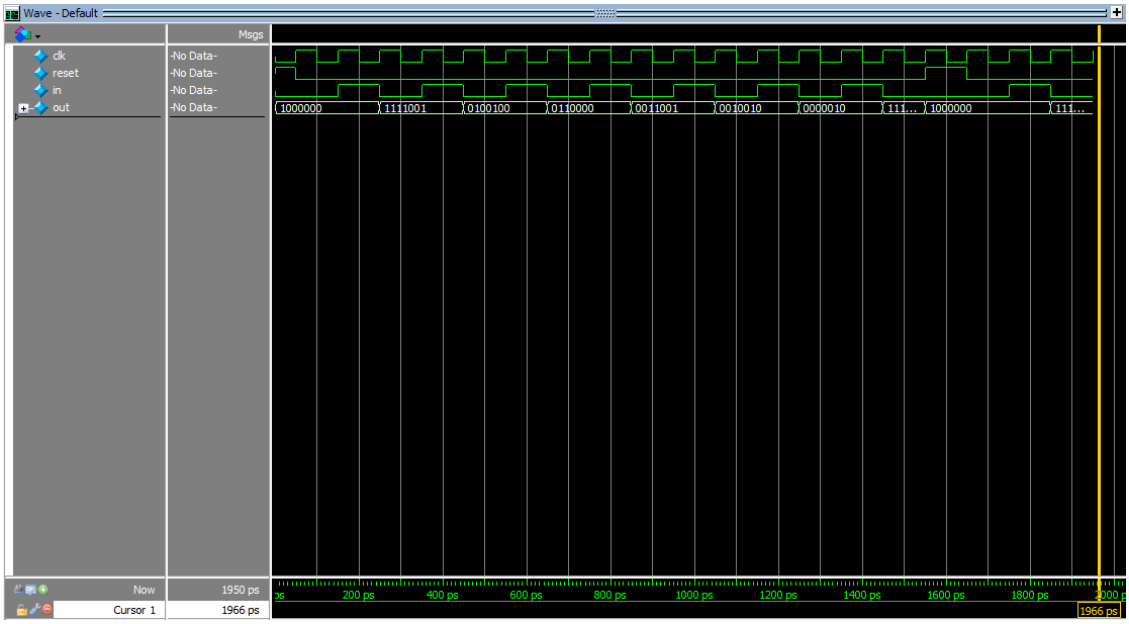
The top image shows the block diagram for the tug of war game. Input is fed into the synchronizer which goes into one edge detector which feeds input into all the LEDS modules, which then output to the LEDs on the board. They also take in input from the LEDs to determine when to switch. The clock is fed into a clock divider, which stores its information, which is then fed to pretty much every component. The computer is made by feeding the clokc into a LSFR, which is then fed into an adder that also takes in input from the switches 0-8. Finally, this is put into an edge detector and fed to the LEDs as well. THhe counter takes input from the LEDs, switches and clock, and outputs to the hex display.

The modelsim showcases the test bench for the module. It showcases the computer using input, and then also the win condition and a reset. The SW[9] is the reset switch for the whole board. The Key 3 represents user input. The playfield is marked by LEDs 1-9. The computers count

is marked by HEX0, and the players by HEX5. For both players, the reaching led 9 and led 1 resets the playfield and increments the counter. the randmoness is shown by the lsfr output.

3-Bit Counter

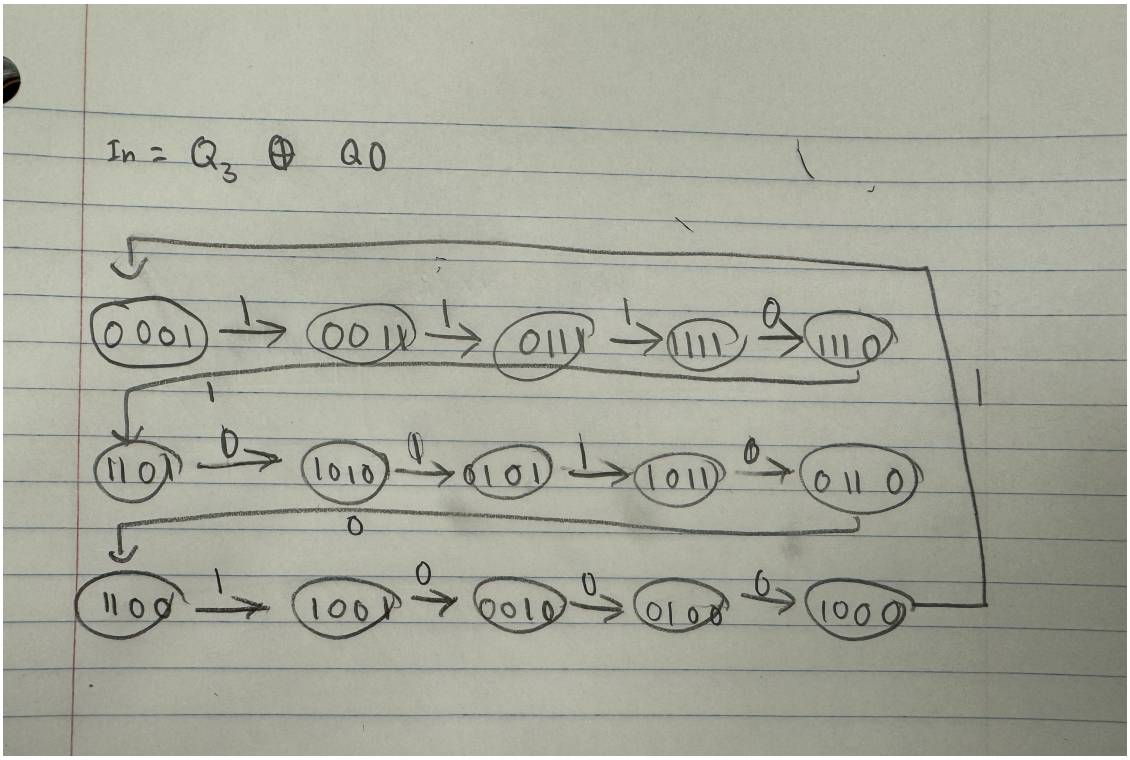
Question:
Simulation of your working 3-bit counter.



A model sim representing the 3-bit counter. The output of the counter shows a display of values from 0-7, and stops counting after the value has reached 7. The reset switch shown sets the value and the display back to 0. The counter increments on the value of in being 1.

4-Bit LFSR

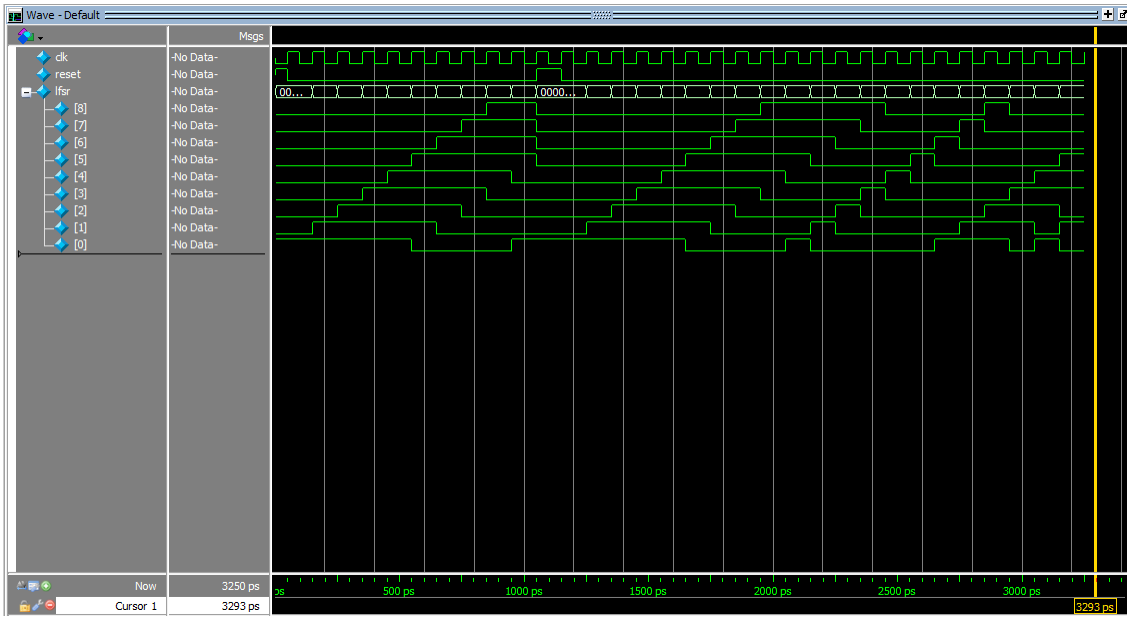
Question:
State diagram of the 4-bit LFSR.



Shows the diagram with all 16 states for the 4-bit LFSR. Each state only has one transition because the result of XOR will always be the same for each state.

9-Bit LFSR

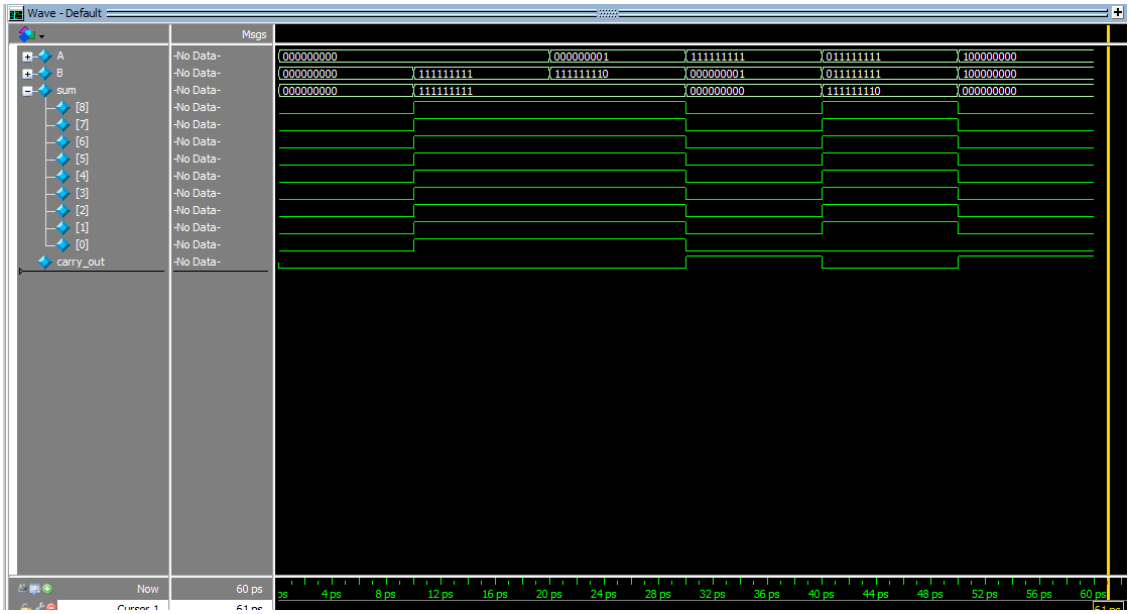
Question:
Simulation of your working 9-bit LFSR with the state cycle length indicated.



The 9 bit lsfr used to generate "random" numbers. It XNOR's the 9th and 5th bits together, and on a reset sets the value back to 1 to start again. There are 512 unique states that the lsfr can go through, and so there are 511 states it can go through before repeating. Thus, the state cycle length is 511.

9-Bit or 10-bit Adder

Question:
Simulation of your working 9- or 10-bit adder that covers the 6 situations described.



I created a 9-bit adder, with a carry out. The modelsim shows the 6 test cases, with the output put into the sum variable. The carry out represents the value of the 10th bit, which can be easily used to check if the sum of two values is greater than or equal to 512.

Resource Utilization

Question:
A screenshot of the "Resource Utilization by Entity" page, showing your design's computed size.

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks	Pins	Virtual Pins	Full Hierarchy Name	Entity Name	Library Name
1	⌄ |tugOfWar	34 (0)	27 (0)	0	0	67	0	|tugOfWar	tugOfWar	work
1	|centerLight:l5|	1 (1)	1 (1)	0	0	0	0	|tugOfWar|centerLight:l5	centerLight	work
2	|clock_divider:cdiv|	16 (16)	16 (16)	0	0	0	0	|tugOfWar|clock_divider:cdiv	clock_divider	work
3	|counter:leftCounter|	10 (10)	3 (3)	0	0	0	0	|tugOfWar|counter:leftCounter	counter	work
4	|edgeDetector:edl|	1 (1)	1 (1)	0	0	0	0	|tugOfWar|edgeDetector:edl	edgeDetector	work
5	|edgeLight:l9|	1 (1)	1 (1)	0	0	0	0	|tugOfWar|edgeLight:l9	edgeLight	work
6	|normalLight:l6|	1 (1)	1 (1)	0	0	0	0	|tugOfWar|normalLight:l6	normalLight	work
7	|normalLight:l7|	1 (1)	1 (1)	0	0	0	0	|tugOfWar|normalLight:l7	normalLight	work
8	|normalLight:l8|	1 (1)	1 (1)	0	0	0	0	|tugOfWar|normalLight:l8	normalLight	work
9	|synch:sL|	2 (2)	2 (2)	0	0	0	0	|tugOfWar|synch:sL	synch	work

The resources utilization by the top-level entity. The utilization is $34 + 27 = 61$.

Reflection

Question:
How many hours (estimated) it took to complete this lab in total, including reading, planning, designing, coding, debugging, and testing.

The lab took me around 2-2.5 hours to complete.