Trinity College Dublin

The University of Dublin

CS2022 Computer Architecture

Michael Manzke

Project 2

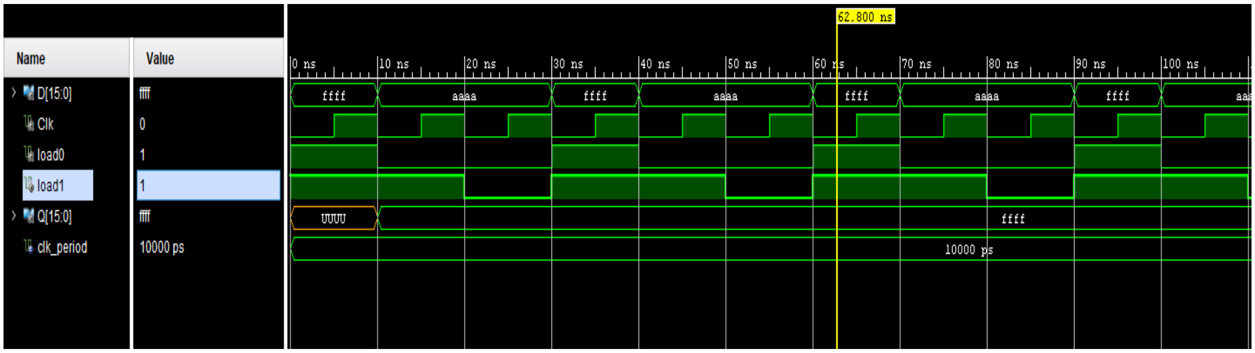
MICROCODED INSTRUCTION SET PROCESSOR

Student Name: Alexandra Silva

Student Number: 15325158

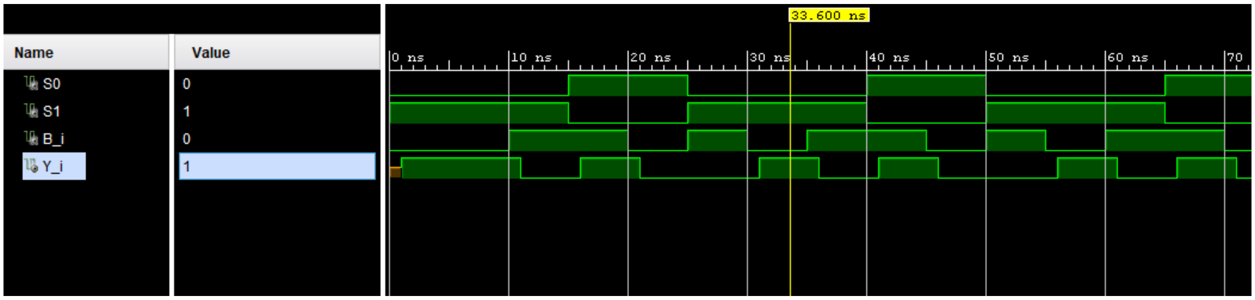
Table of contents

Register



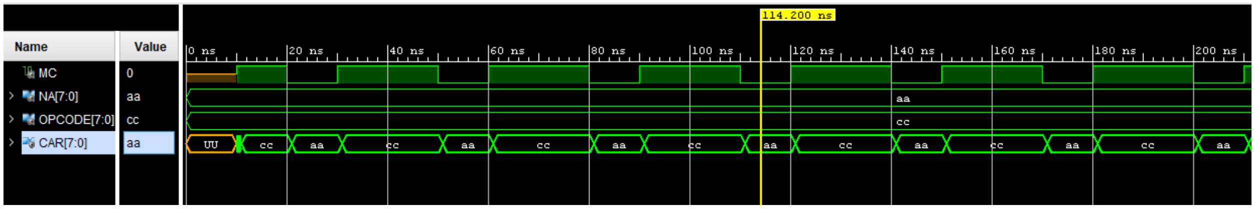
The register is basically the same code used in the precious assignments. It works as expected for each edge when the signal is high on a given rising clock.

Mux2_1



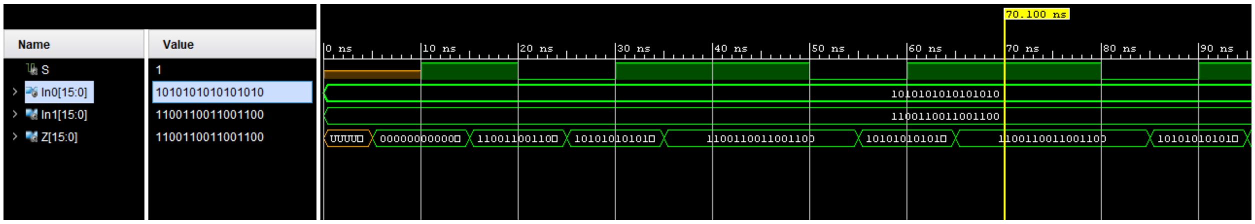
In this multiplexer the values change between 0 and 1 and the output changes accordingly.

Mux2_8



This multiplexer takes the various inputs and produces an 8 bit output.

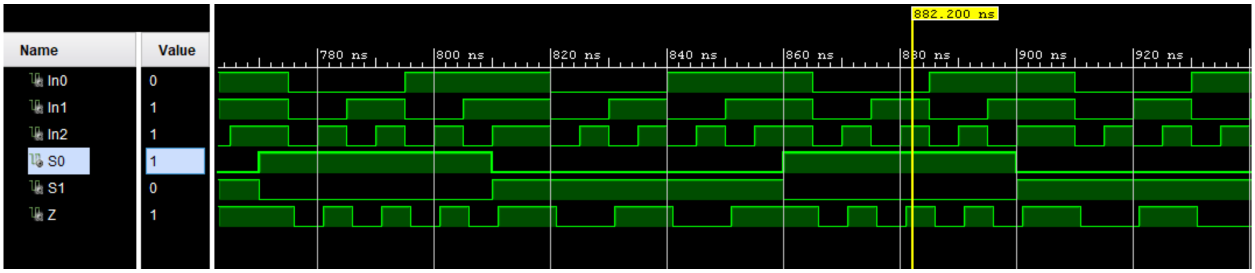
Mux2_16



Name	Value
S	1
In0[15:0]	1010101010101010
In1[15:0]	1100110011001100
Z[15:0]	1100110011001100

This multiplexer goes through a series of changes as S goes between the values of 1 and 0. The output switches between the values of in0 and in1.

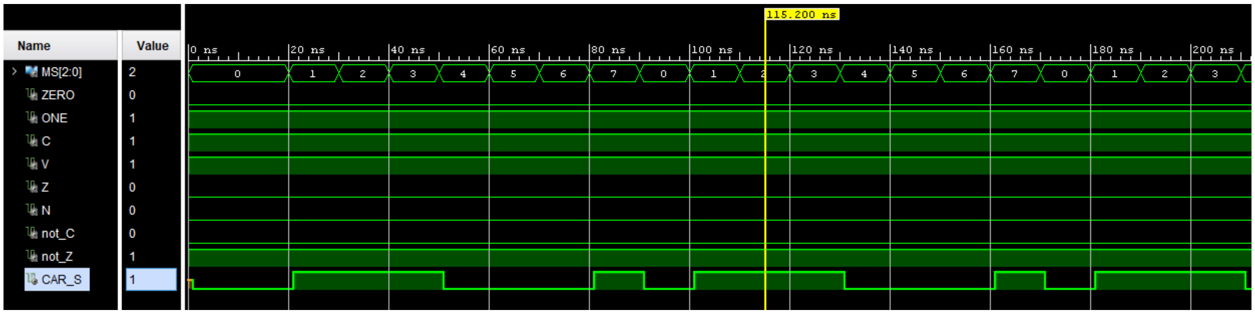
Mux3_1



Name	Value
In0	0
In1	1
In2	1
S0	1
S1	0
Z	1

This multiplexers output, Z, changes values according to the changing inputs of in0, in1, in2, s0 and s1.

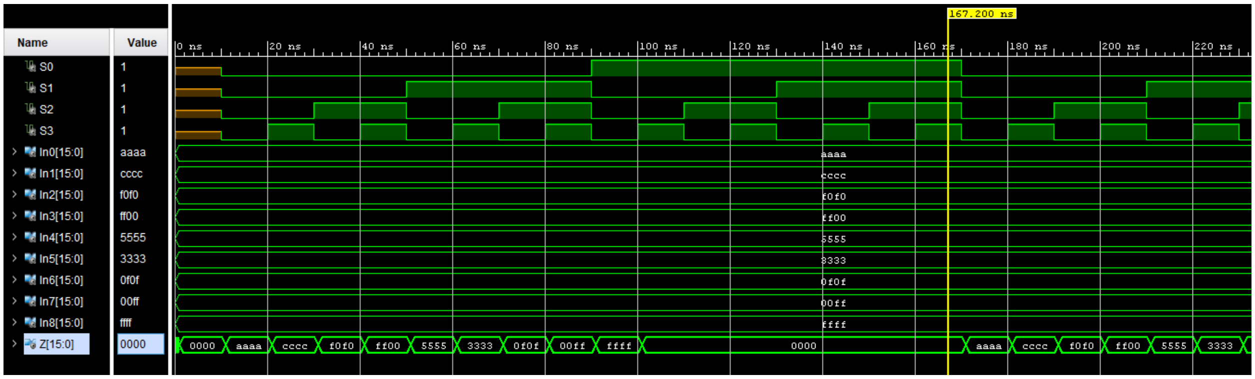
Mux8_1



Name	Value
MS[2:0]	2
ZERO	0
ONE	1
C	1
V	1
Z	0
N	0
not_C	0
not_Z	1
CAR_S	1

This multiplexer takes in various inputs and produces a 1 bit output.

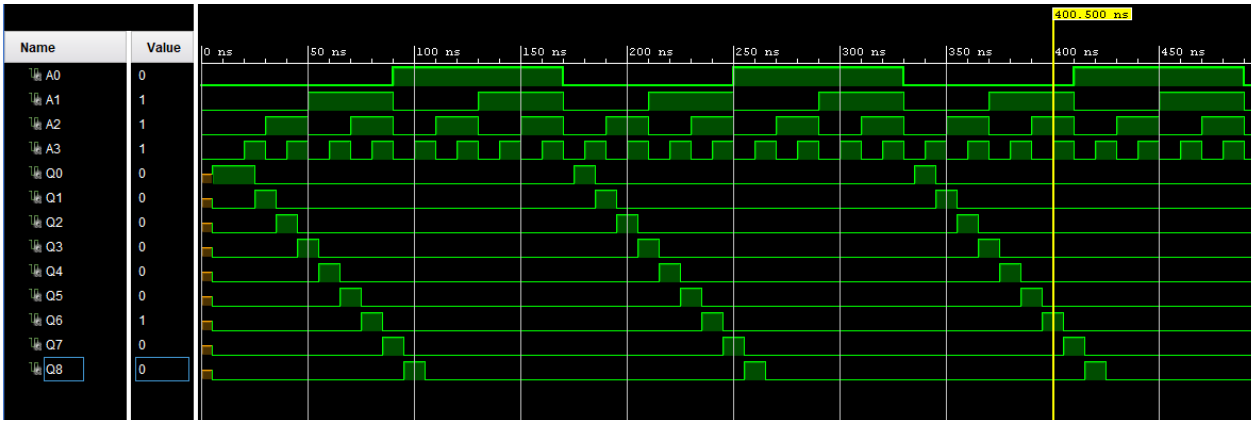
Mux9_16



Name	Value
S0	1
S1	1
S2	1
S3	1
In0[15:0]	aaaa
In1[15:0]	cccc
In2[15:0]	f0f0
In3[15:0]	ff00
In4[15:0]	5555
In5[15:0]	3333
In6[15:0]	0f0f
In7[15:0]	00ff
In8[15:0]	ffff
Z[15:0]	0000

This multiplexer cycles through in0 to in 8 and takes all combinations of its inputs s0 to s3 to get the output signal Z.

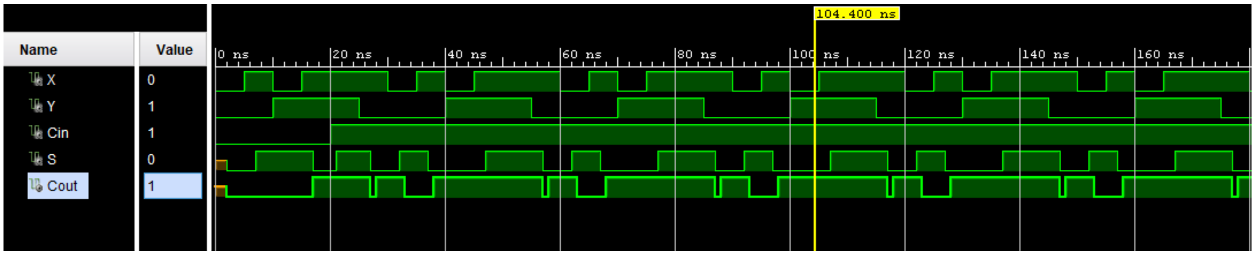
Decoder



Name	Value
A0	0
A1	1
A2	1
A3	1
Q0	0
Q1	0
Q2	0
Q3	0
Q4	0
Q5	0
Q6	1
Q7	0
Q8	0

This waveform is correct as it goes through every possible combination of inputs, causing each and every output to turn high via register selection.

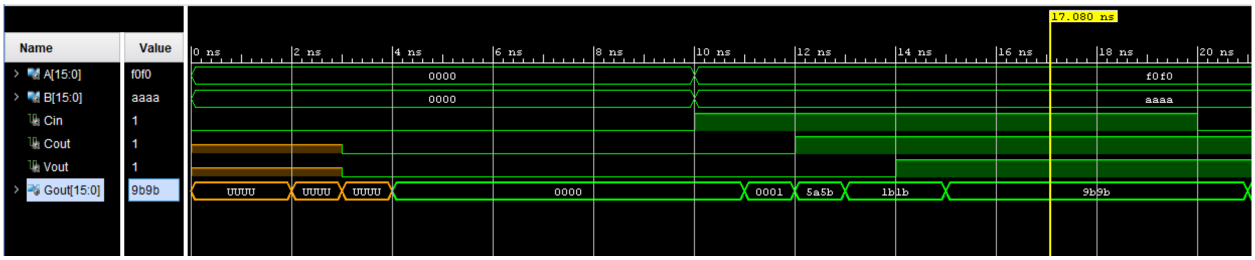
Full Adder



Name	Value
X	0
Y	1
Cin	1
S	0
Cout	1

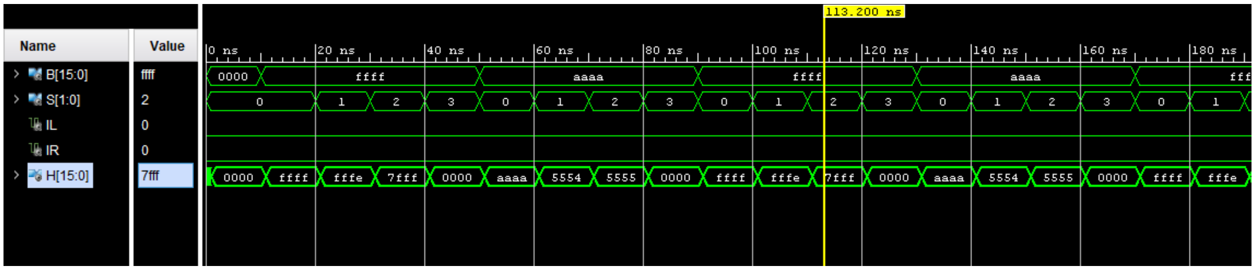
The full adder determines the correct carry for each input. Then the appropriate operations are performed resulting in the output shown.

Ripple Adder



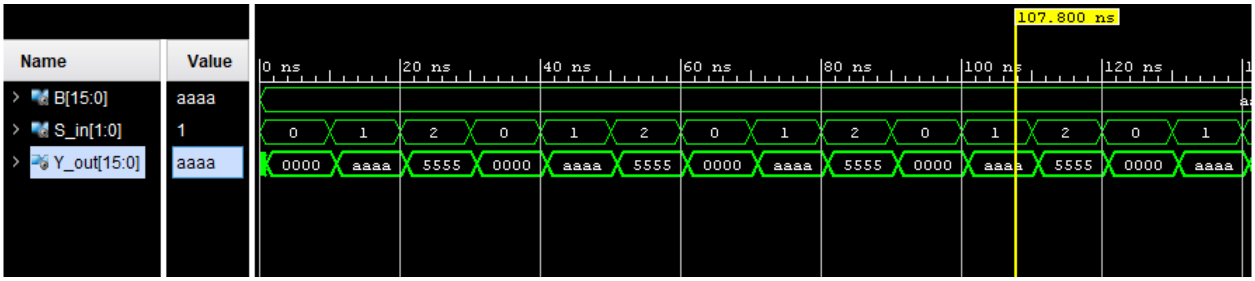
The ripple adder has two inputs of 16 bits A and B. It determines the Cin, Cout and Vout flags which will all be either 0 or 1. The output Gout then gathers all this information to produce the signal shown.

Shifter



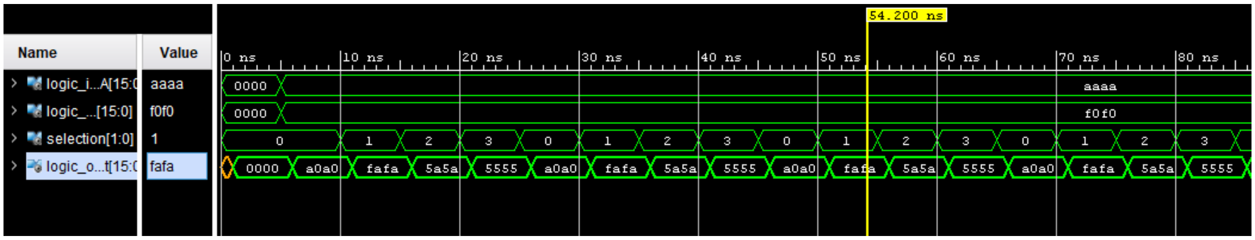
The shifter outputs left and right shifts according to the inputs given by B and S.

Circuit B



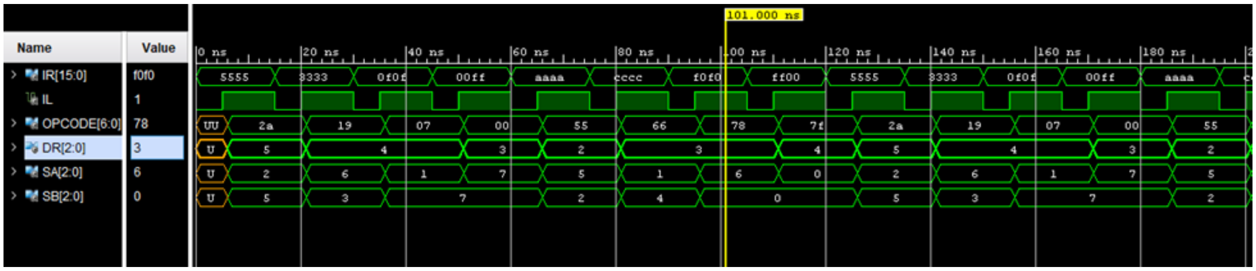
Logic Circuit B provides ab output depending on the constant input B and the varying input Sin.

Circuit A to B



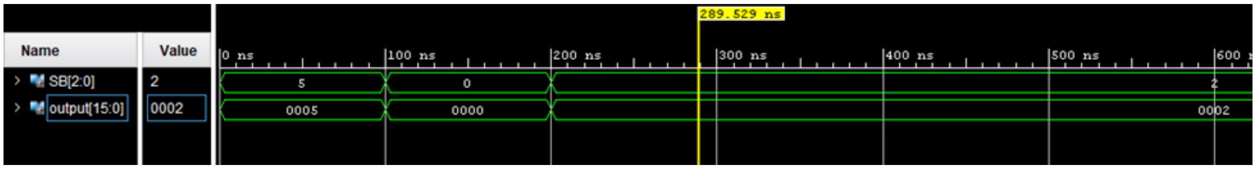
Logic Circuit A to B has a selection pin that determines the operations that will be performed on the inputs. This is what causes the output to change despite the two constant inputs.

Instruction



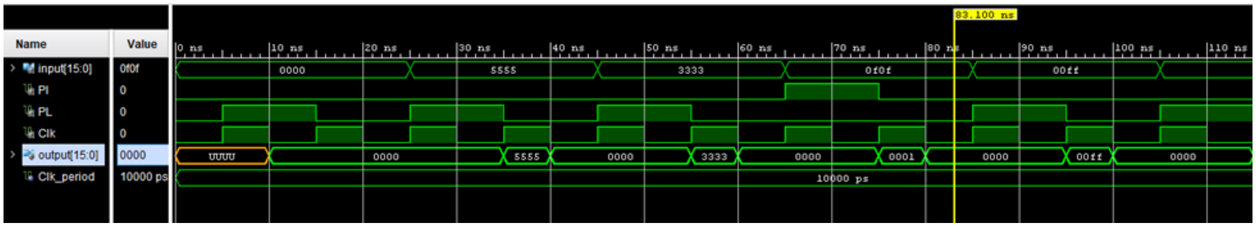
Instruction has 4 outputs and 3 inputs. The way it works is that it loads an instruction from memory and performs it on the inputs. This then gives us our outputs.

Zero Fill

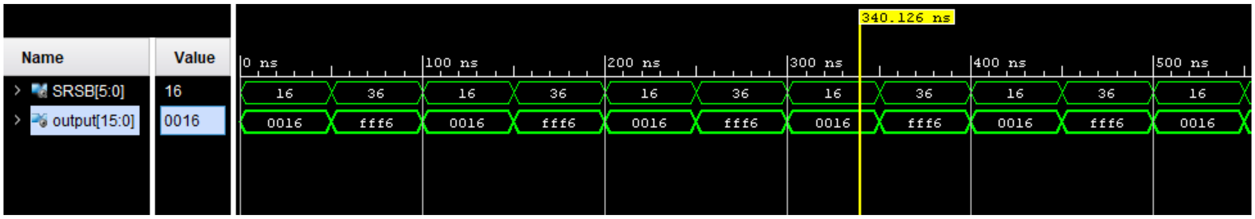


Zero Fill simply changes the input SB. As seen the output correlates to the changing input.

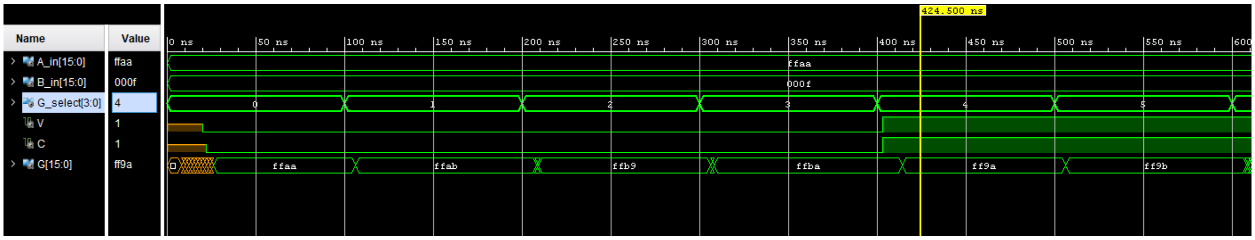
Program Counter



Extend

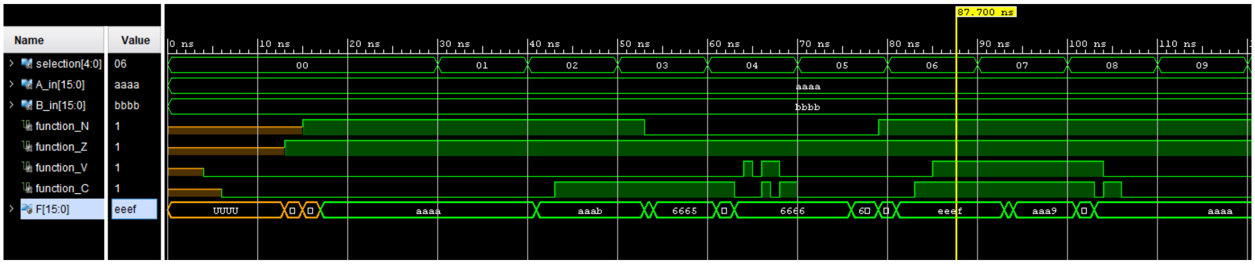


Arithmetic Logic Unit



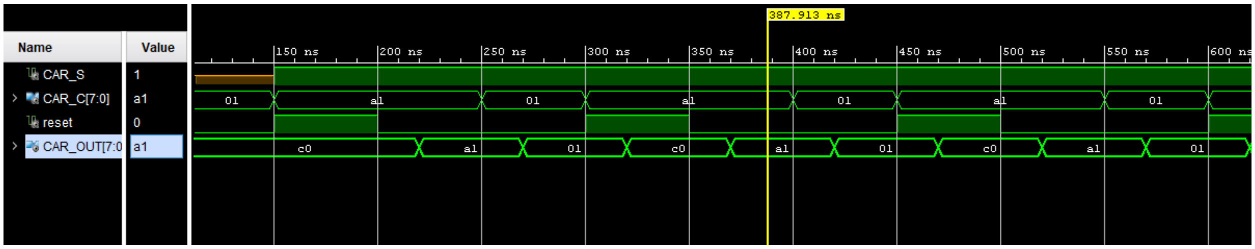
This gives us the output G using the given inputs and a selct

Function unit

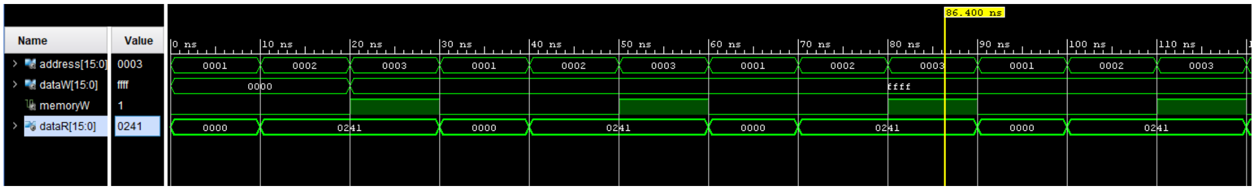


This raises the N, Z, C and V flags

Control Address Register

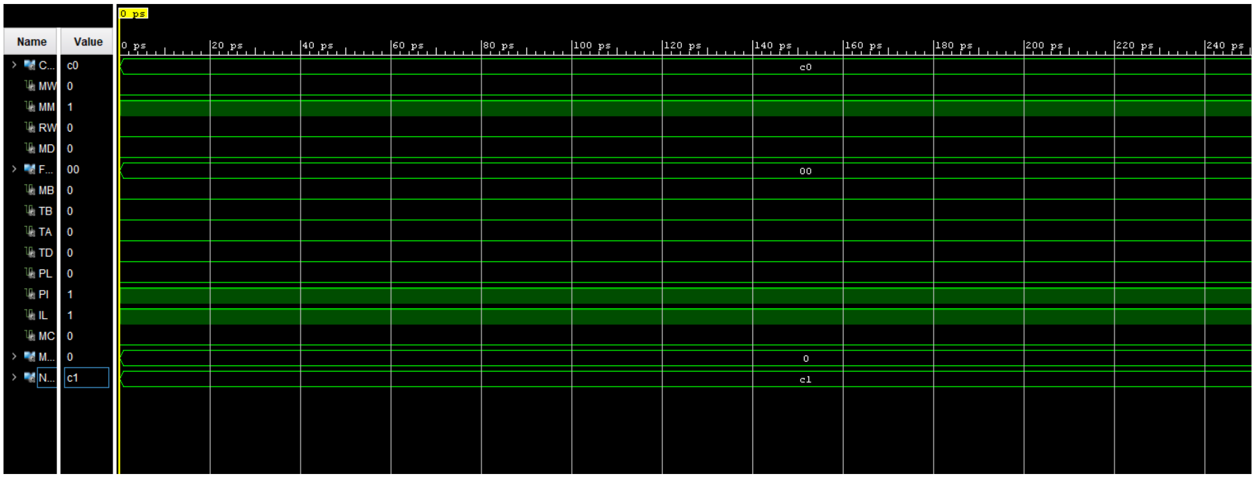


Memory

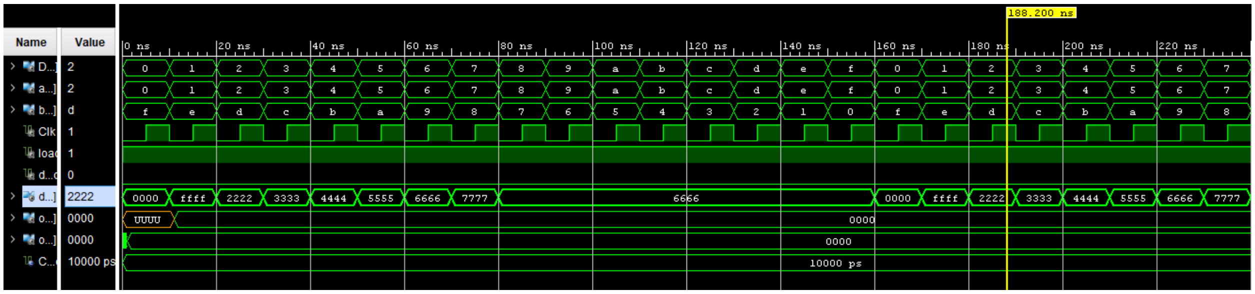


Memory both loads data in and stores it. Here I have loaded in values from the memory address 1 and 2 and stored them in the memory address 3. The data coming in is kept constant while the other input switches between the values of 1 and 0. When its 1 we get a value of 0000001001000001 and when its zero we get a value of 0000000000000000

Control Memory

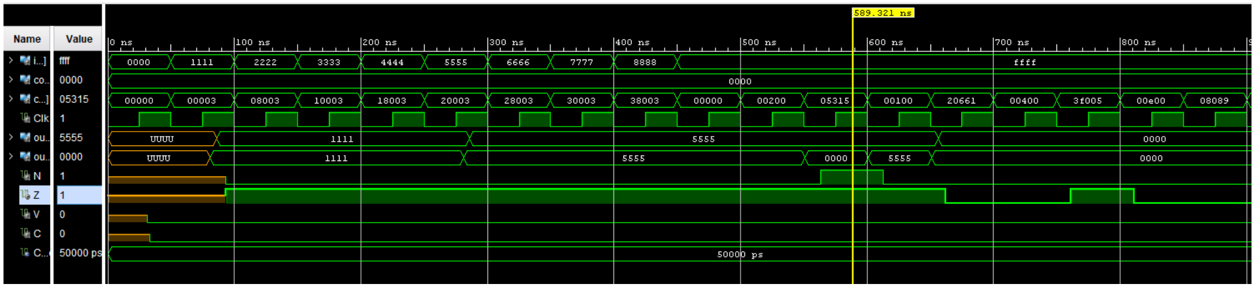


Register File



This shows the registers, decoder and multiplexer8_16 working together. There is a slight delay at the beginning due to the amount of entities working and initializing. Output B is undefined for about 10ns

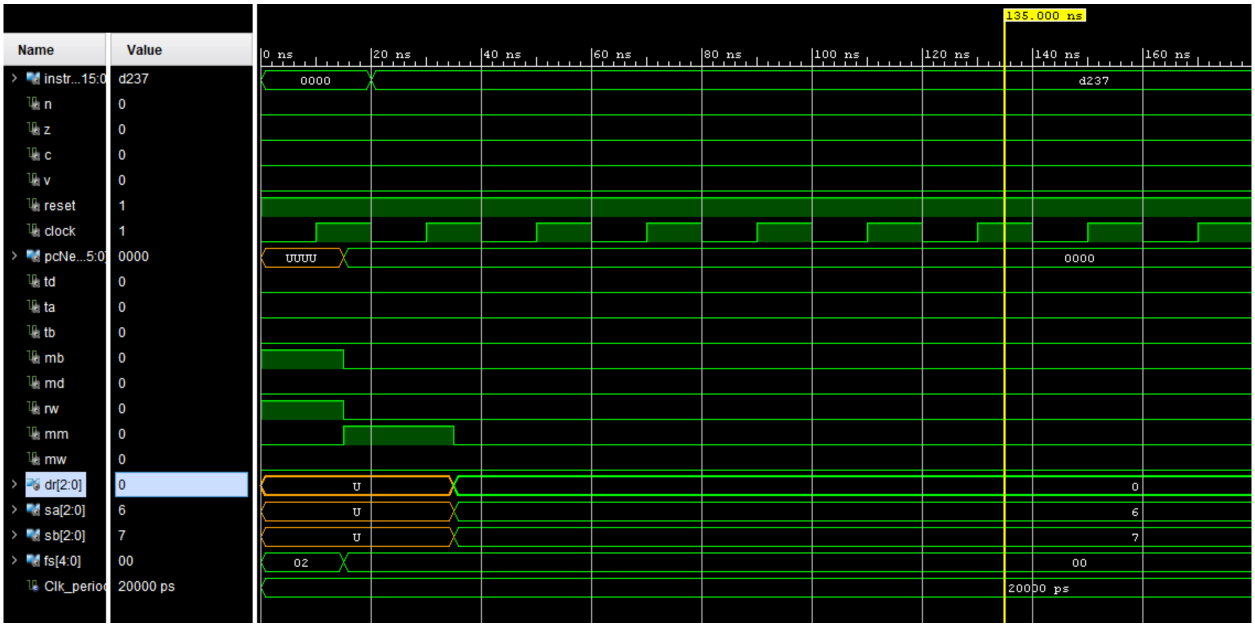
Datapath



Name	Value
i...]	ffff
co...	0000
c...]	05315
Clk	1
ou...	5555
ou...	0000
N	1
Z	1
V	0
C	0
C...	50000 ps

589.321 ns
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800 ns
0000 1111 2222 3333 4444 5555 6666 7777 8888 ffff
0000
00000 00003 08003 10003 18003 20003 28003 30003 38003 00000 00200 05315 00100 20661 00400 3f005 00e00 08089
UUUU 1111 5555 0000
UUUU 1111 5555 0000 5555 0000
50000 ps

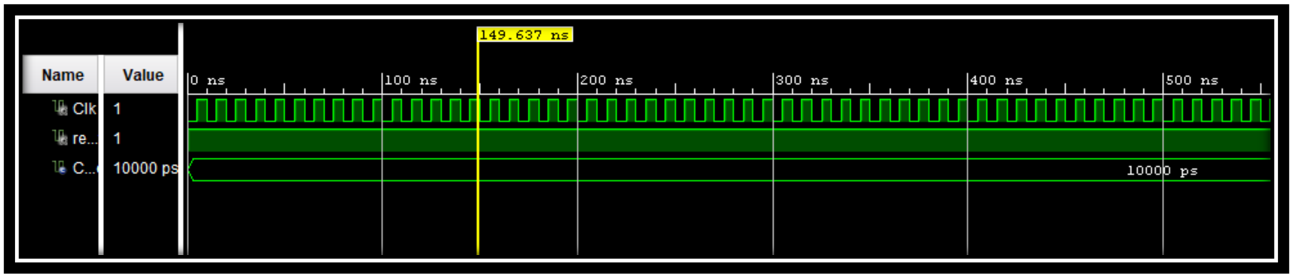
Microprogrammed Controller



Name	Value
instr...15:0	d237
n	0
z	0
c	0
v	0
reset	1
clock	1
pcNe...5:0	0000
td	0
ta	0
tb	0
mb	0
md	0
rw	0
mm	0
mw	0
dr[2:0]	0
sa[2:0]	6
sb[2:0]	7
fs[4:0]	00
Clk_period	20000 ps

The Value I used to test tis out was 1101001000110111. With this input the following results should occur dr = 000, sa = 110, sb = 111 & td, ta, tb, mb, fs etc = 0.

Full Microprocessor



Name	Value
Clk	1
re...	1
C...	10000 ps