Implementing a GCD with a programmable datapath

Andrew Belcher
StudentID:17010347

December 11, 2018

source link: https://gitlab.uwe.ac.uk/a2-belcher/dd-coursework1

Contents

1 Introduction **3**
1.1 Algorithm 3

2 Control Unit **4**

3 Control Words **5**

4 Testing **6**
4.1 Input Mux 6
4.2 Register File 7
4.3 ALU 8
4.4 Shifter 9

5 GCD simulation results **10**

1 Introduction

Tasked with developing a GCD that features a programmable datapath and controller, development of 5 different components, a mux, a register file, a alu, shifter, and output enable switch were all that were needed for the core datapath. A main datapath that included the core along with a status register to determine whether to enable output or not along with a controller were to be developed. To suit both the controller and the datapath a rom was created and populated with the correct states and flags needed to transistion and process digital inputs and result in the desired outputs. This GCD unit should be able to be reprogrammed via the rom to change the behaviour of the circuit in order to serve other purposes than a GCD.

1.1 Algorithm

Euclidean Algorithm for GCD

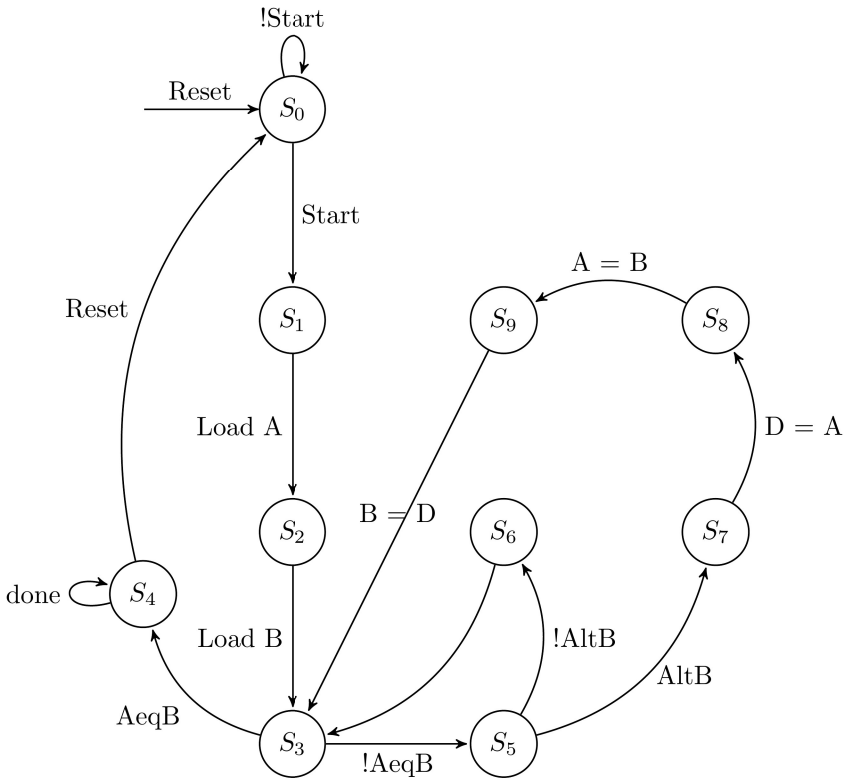
```
uint8_t A = X;
uint8_t B = Y;

while(A != Y)
{
        if(A < B)
                B = B - A;
        else
                A = A - B;
}

Z = A;
return Z;
```

2 Control Unit

Greatest Common Divisor
FSM



3 Control Words

State table for 16 bit control words used to implement a GCD for a programmable datapath.

GCD Control Word Table

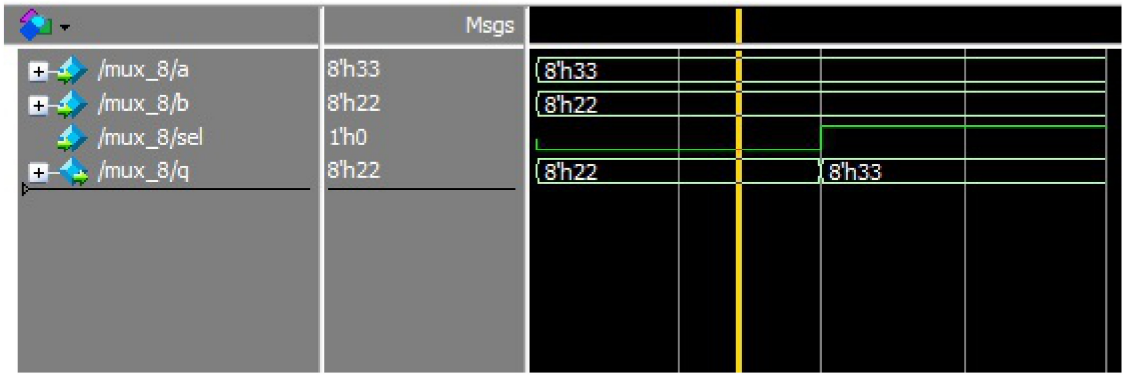
CW	IE	WA	RAA	RAB	ALU op	Shift op	OE
0	0	none	x	x	pass	pass	0
1	1	R0	x	x	pass	pass	0
2	1	R1	x	x	pass	pass	0
3	0	R2	R0	R1	sub	pass	0
4	0	none	R0	x	pass	pass	1
5	0	none	R2	x	pass	pass	0
6	0	R0	R2	x	pass	pass	0
7	0	R3	R0	x	pass	pass	0
8	0	R0	R1	x	pass	pass	0
9	0	R1	R3	x	pass	pass	0

GCD Register Roles

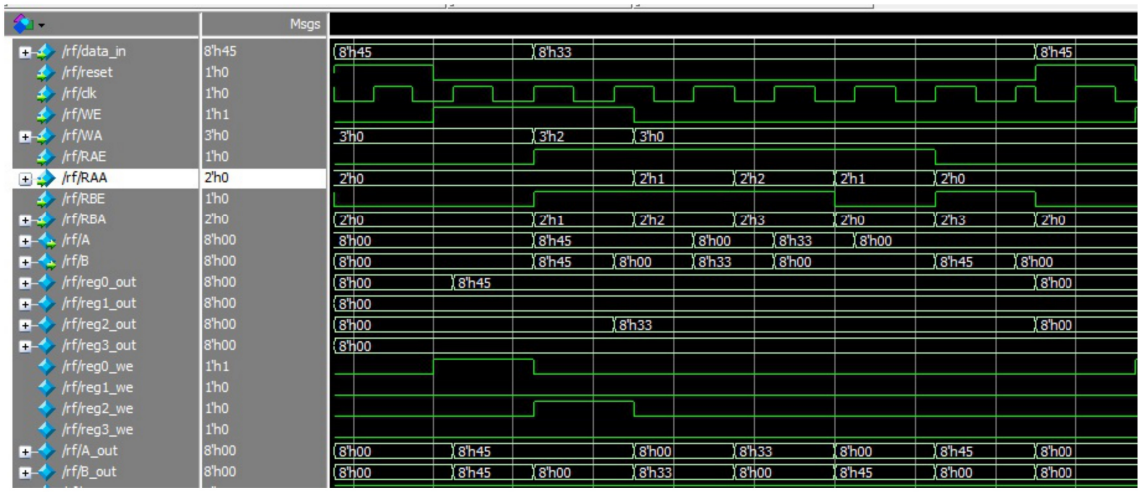
Register	Role
R0	A
R1	B
R2	Result(C)
R3	Temp (D)

4 Testing

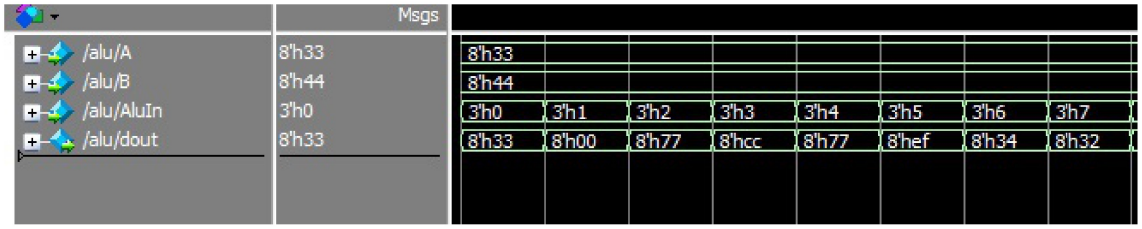
4.1 Input Mux



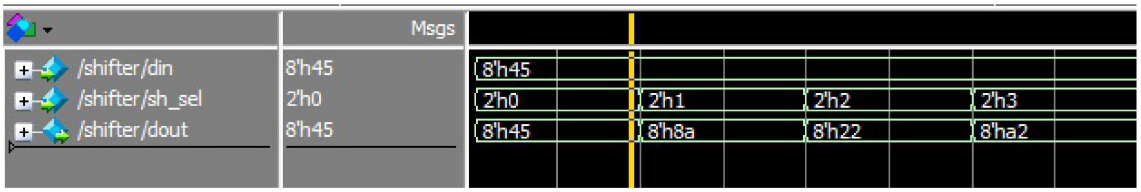
4.2 Register File



4.3 ALU

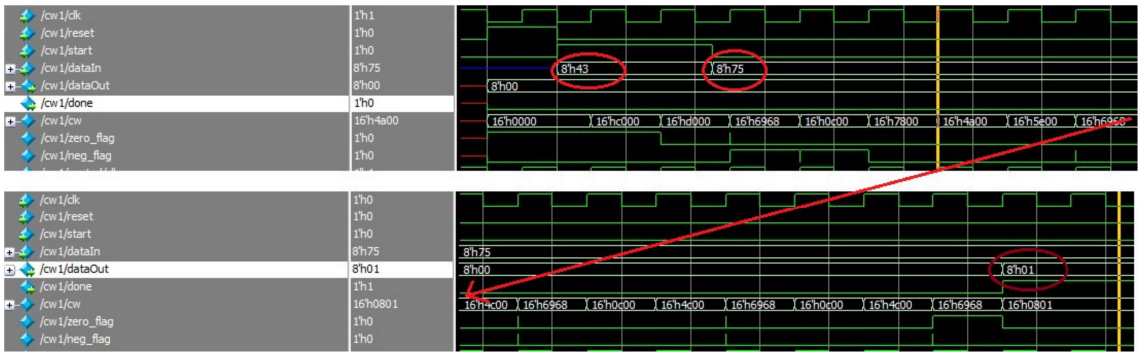

Msgs
/alu/A
/alu/B
/alu/AluIn
/alu/dout
8'h33
8'h44
3'h0
8'h33
8'h33
8'h44
3'h0
3'h1
3'h2
3'h3
3'h4
3'h5
3'h6
3'h7
8'h33
8'h00
8'h77
8'hcc
8'h77
8'hef
8'h34
8'h32

4.4 Shifter


Msgs
/shifter/din
/shifter/sh_sel
/shifter/dout
8'h45
2'h0
8'h45
8'h45
2'h0
2'h1
2'h2
2'h3
8'h45
8'h8a
8'h22
8'ha2

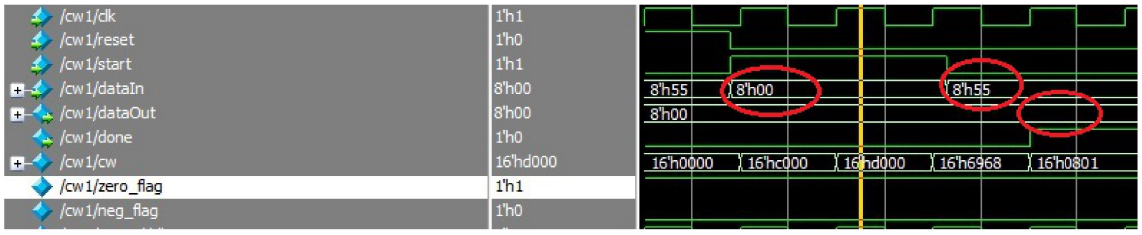
5 GCD simulation results

67 - 117 = 1

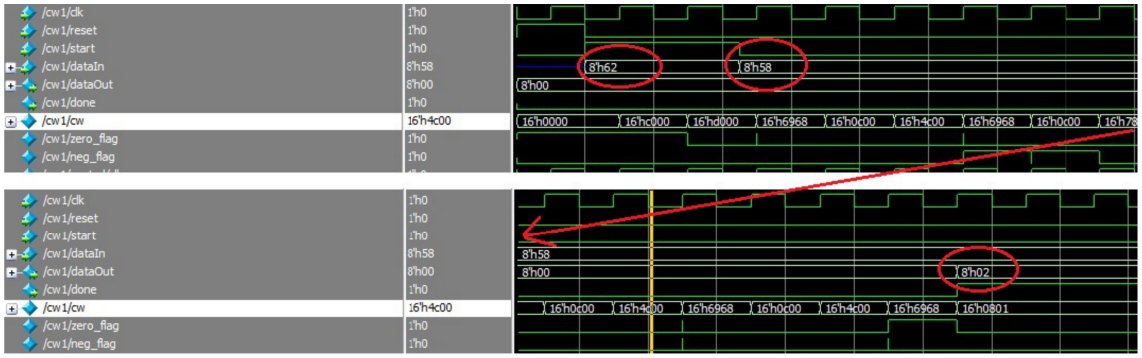


51 - 85 = 17

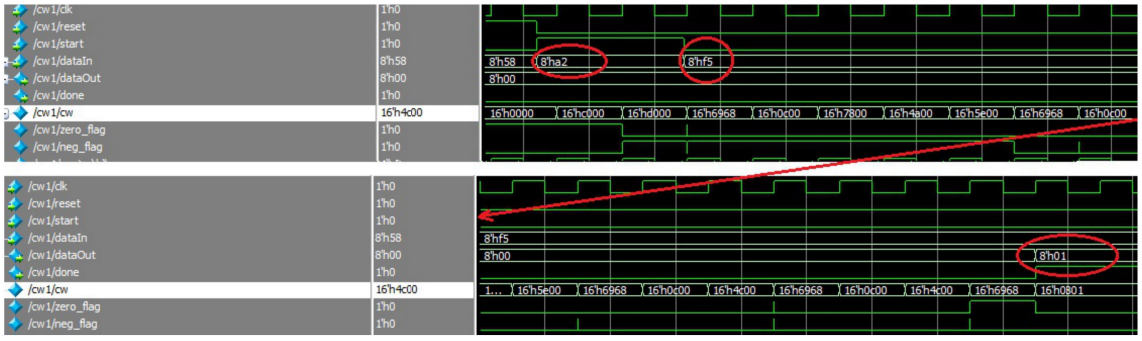
0 - 85 = 0



98 - 88 = 2


/cw1/clk
/cw1/reset
/cw1/start
/cw1/dataIn
/cw1/dataOut
/cw1/done
/cw1/cw
/cw1/zero_flag
/cw1/neg_flag
1'h0
1'h0
1'h0
8'h58
8'h00
1'h0
16'h4c00
1'h0
1'h0
8'h62
8'h58
8'h00
16'h0000
16'hc000
16'hd000
16'h6968
16'h0c00
16'h4c00
16'h6968
16'h0c00
8'h58
8'h00
8'h02
16'h0c00
16'h4c00
16'h6968
16'h0c00
16'h4c00
16'h6968
16'h0801

(-94) - (-11) = 1


/cw1/clk
/cw1/reset
/cw1/start
/cw1/dataIn
/cw1/dataOut
/cw1/done
/cw1/cw
/cw1/zero_flag
/cw1/neg_flag
1'h0
1'h0
1'h0
8'h58
8'h00
1'h0
16'h4c00
1'h0
1'h0
8'h58
8'ha2
8'hf5
8'h00
16'h0000
16'hc000
16'hd000
16'h6968
16'h0c00
16'h7800
16'h4a00
16'h5e00
16'h6968
16'h0c00
8'hf5
8'h00
8'h01
16'h5e00
16'h6968
16'h0c00
16'h4c00
16'h6968
16'h0c00
16'h4c00
16'h6968
16'h0801

GCD Test Values

A	B	Result
67	117	1
51	85	17
0	85	0
98	88	2
-94	-11	1
56	48	8
-11	-56	1

.do script for testing GCD

```
force reset 0
force start 0
force clk 0 0, 1 10 -repeat 20
run 20

force reset 1
force start 0
run 20

force start 1
force reset 0
force dataIn f5
run 45

force start 0
force dataIn c8
run 1000
```