Class:	CPE-300L	Semester:	Spring 2020

Points		Document author:	Brysen Kokubun
		Author's email:	Brysenkokubun96@gmail.com
		Document topic:	Postlab 8

Instructor's comments:

1. Introduction / Theory of Operation

- For this lab experiment we familiarized ourselves with the TimeQuest Analyzer and used the GDP and altered various components to implement the algorithm that calculates the greatest common divisor.

a. *What is slack in time quest analysis?*

- Slack is a range value which indicates if the timing requirements are met or not met. Positive slack is when the timing requirements are met (Data received before Required). Negative slack is when the timing requiring are not met (Data received after Required).

b. *Why is it important to consider timing analysis in a digital circuit?*

- It is important to consider timing analysis in a digital circuit because it gives you the needed amount of time in order to get data to various components. If the timing is too fast, then the data may not be able to be received before required in the given time period. Therefore, the timing is crucial for all the components to work in sync.

2. Prelab report

- Attached below

3. Experiment Results

Experiment #1: General Datapath

Verilog Code

```
module GDP(Result, start, restart, clk, n, done);
// top module combines all other modules, including DP and CU
// Note: AlB checks if A < B
//       AeqB checks if A = B

input start, clk, restart;
    input [7:0] n;
    output [7:0] Result; // 1 or 0
    output done;

    wire WE, RAE, RBE, OE, AlB, IE;
    wire [1:0] WA, RAA, RBA, SH;
    wire [2:0] ALU;

CU control (IE, WE, WA, RAE, RAA, RBE, RBA, ALU, SH, OE, start, clk, restart, AlB, AeqB);

DP datapath (AlB, AeqB, Result, n, clk, IE, WE, WA, RAE, RAA, RBE, RBA, ALU, SH, OE);

    assign done = OE;

endmodule

module CU(IE, WE, WA, RAE, RAA, RBE, RBA, ALU, SH, OE, start, clk, reset, AlB, AeqB);
```

```
    input start, clk, reset;
    output IE, WE, RAE, RBE, OE;
    output [1:0] WA, RAA, RBA, SH;
    output [2:0] ALU;

    input wire AlB, AeqB;
    reg [2:0] state;
    reg [2:0] nextstate;

    parameter S0 = 0;
    parameter S1 = 1;
    parameter S2 = 2;
    parameter S3 = 3;
    parameter S4 = 4;
    parameter S5 = 5;
    parameter S6 = 6;

    initial
        state = S0;

    // State register
always @ (posedge clk)
    begin
        state <= nextstate;
    end
    // NS logic
always @ (*)
case(state)
S0:     if(start) nextstate = S1;
    else    nextstate = S0;
S1:     nextstate = S2;
S2:  nextstate = S3;
S3:  if (AeqB) nextstate = S6;
    else if (~AlB) nextstate = S4;
  else nextstate = S5;
S4: nextstate = S3;
S5: nextstate = S3;
S6: if (reset) nextstate = S0;
  else nextstate = S6;

default: nextstate = S0;
        endcase

    // output logic
    assign IE = (state == S1) || (state == S2);
    assign WE = (state == S0) || (state == S1) || (state == S2) || (state == S4) || (state == S5);
    assign WA[1] = (state == S2) || (state == S5);
    assign WA[0] = (state == S1) || (state == S4);
    assign RAE = (state == S0) || (state == S3) || (state == S4) || (state == S5) || (state == S6);
    assign RAA[1] = (state == S5);
    assign RAA[0] = (state == S3) || (state == S4) || (state == S6);

    assign RBE = (state == S0) || (state == S3) || (state == S4) || (state == S5);
    assign RBA[1] = (state == S3) || (state == S4);
    assign RBA[0] = (state == S5);

    assign ALU[2] = (state == S0) || (state == S3) || (state == S4) || (state == S5);
    assign ALU[1] = 0;
    assign ALU[0] = (state == S0) || (state == S3) || (state == S4) || (state == S5);
    assign SH[1] = 0;
    assign SH[0] = 0;
    assign OE = (state == S6);
endmodule
```

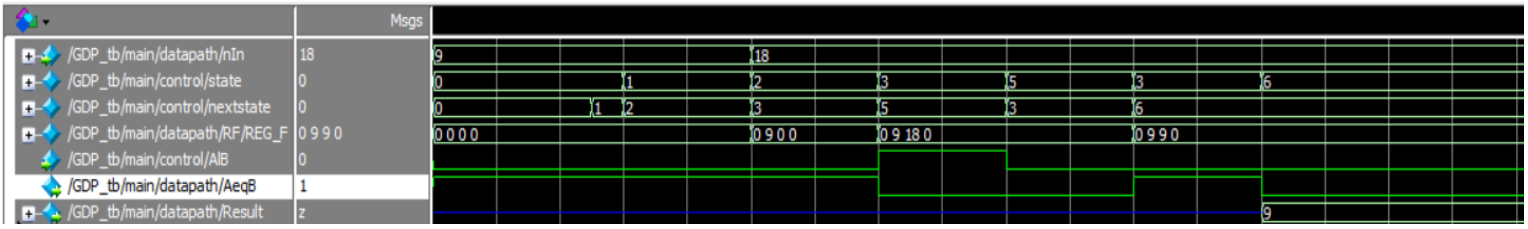
```
module DP(AlB, AeqB, Result, nIn, clk, IE, WE, WA, RAE, RAA, RBE, RBA, ALU, SH, OE);

    input clk, IE, WE, RAE, RBE, OE;
    input [1:0] WA, RAA, RBA, SH;
    input [2:0] ALU;
    input [7:0] nIn;

    output AlB, AeqB;
    output wire [7:0] Result;

    reg [7:0] rfIn;
    wire [7:0] RFa, RFb, aluOut, shOut, n;

     initial
     rfIn = 0;

    always @ (*)
     rfIn = n;

    mux8 mux (n, shOut, nIn, IE);
    Regfile RF (clk, RAA, RFa, RBA, RFb, WE, WA, rfIn, RAE, RBE);
    alu theALU (aluOut, RFa, RFb, ALU);
    shifter SHIFT (shOut, aluOut, SH);
    buff bufferl (Result, shOut, OE);

    assign AlB = (aluOut[7]) & (|aluOut[6:0]);
    assign AeqB = ~(|aluOut[7:0]);

endmodule
// final buffer
module buff(output reg [7:0] result, input[7:0] a, input bufl);
    always @(*)
        if(bufl == 1)
            result = a;
        else
            result = 8'bzzzz_zzzz;
endmodule


// Shifter
module shifter (out,a,sh);
    input [7:0] a;
    input [1:0] sh;
    output reg [7:0] out;

    always @ (*)
    begin
        case(sh)
            3'b00: out=a;
            3'b01: out=a << 1;
            3'b10: out=a >> 1;
            3'b11: out={a[6],a[5],a[4],a[3],a[2],a[1],a[0], a[7]} ;
        endcase
    end
endmodule
```

```
// 2-to-1 mux (sel = 0 -> choose a)
module mux8(result, a, b, sel);

    output reg[7:0] result;
    input[7:0] a;
    input[7:0] b;
    input sel;

    always @(*)
        if(sel == 0)
            result = a;
        else
            result = b;

endmodule

// Regfile for GDP
module Regfile(
                    input clk,
                    input   [1:0]  RAA, // Port A Read address
                    output  [7:0] ReadA,   // Port A
                    input   [1:0] RBA,   // Port B Read address
                    output  [7:0] ReadB,   // Port B
                    input   WE,    // Write enable
                    input   [1:0] WA,  // Write port register                          //address
                    input   [7:0] INPUT_D,  // Write data port
                    input   RAE,  // Port A decoder enable
                    input   RBE // Port B decoder enable

                    );
                // width          depth
                reg [7:0]  REG_F [0:3];

// Write only when WE is asserted
initial
begin

   REG_F[0] = 0;

    REG_F[1] = 0;

     REG_F[2] = 0;

  REG_F[3] = 0;

  end

always @(posedge clk)
if (WE == 1)  REG_F[WA] <= INPUT_D;

  //reading to Port A and B, combinational
  assign ReadA = (RAE)? REG_F [RAA]:0;
  assign ReadB = (RBE)? REG_F [RBA]:0;

endmodule
// ALU
module alu (out,a,b,sel);
    input [7:0] a,b;
    input [2:0] sel;
    output [7:0] out;
    reg [7:0] out;

    always @ (*)
    begin
        case(sel)
            3'b000: out=a;
            3'b001: out=a&b;
            3'b010: out=a|b;
            3'b011: out=!a;
            3'b100: out=a+b;
            3'b101: out=a-b;
            3'b110: out=a+1;
            3'b111: out=a-1;
        endcase
    end
endmodule
```

TestBench Code

```
module GDP_tb;
    wire displayRes;
    reg start, restart, clk;
    wire [7:0] A;
    reg [7:0] In;

    initial
     begin
        start = 0;
        restart = 0;
        clk = 0;

        forever
          #2 clk = ~clk;
    end

    always
    begin
        In = 9;
        #5;
        start = 1;
        $write("Input A: %2d " , In);
        #5;
        In = 18;
        #5;
        $write("Input B: %2d " , In);
        while(displayRes != 1) // Wait for completion signal
            #5 begin end

    $write("Output: %2d,  ", A);

    if(A == 9)          // calculates if A = 9
            $display("Correct!");
            else
                $display("Incorrect!");
            #5;
            restart = 1;
            start = 0;
      #5;

        $stop;
    end
//Initialize GDP
GDP main (A, start, restart, clk, In, displayRes);
endmodule
```

Waveform



Unable to grab DE2 board due to lab being closed during COVID-19 shutdown

Experiment #2: TimeQuest analysis

- Results of TimeQuest with 1ns clock period. Received a skew of -7.044, therefore the timing requirements are not met. Resulting in data being received **after** it is required.

Setup: clk

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-7.044	CU:control|state.S4	CU:control|state.S5	clk	clk	1.000	0.000	8.082
2	-7.042	CU:control|state.S4	CU:control|state.S4	clk	clk	1.000	0.000	8.080
3	-7.040	CU:control|state.S4	DP:datapath|Regfile:RF|REG_F~31	clk	clk	1.000	-0.003	8.075
4	-7.039	CU:control|state.S4	DP:datapath|Regfile:RF|REG_F~7	clk	clk	1.000	-0.003	8.074
5	-6.958	CU:control|state.S4	DP:datapath|Regfile:RF|REG_F~19	clk	clk	1.000	0.000	7.996
6	-6.941	CU:control|state.S4	CU:control|state.S4	clk	clk	1.000	0.000	7.979
7	-6.938	CU:control|state.S4	CU:control|state.S5	clk	clk	1.000	0.000	7.976
8	-6.921	CU:control|state.S4	CU:control|state.S4	clk	clk	1.000	0.000	7.959

Path #1: Setup slack is -7.044 (VIOLATED)

Path Summary | Statistics | Data Path | Waveform

Data Arrival Path

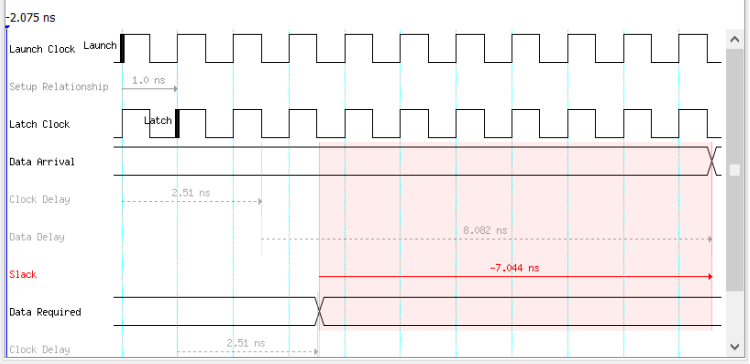
	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	2.510	2.510					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1	PIN_H2	clk
3	1.016	1.016	RR	CELL	1	IOC_X0_Y6_N0	clk|combout
4	1.148	0.132	RR	IC	1	CLKCTRL_G2	clk~clkctrl|inclk[0]
5	1.148	0.000	RR	CELL	39	CLKCTRL_G2	clk~clkctrl|outclk
6	1.908	0.760	RR	IC	1	LCFF_X10_Y8_N5	control|state.S4|clk

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	1.000	1.000					latch edge time
2	3.510	2.510					clock path
1	1.000	0.000					source latency
2	1.000	0.000			1	PIN_H2	clk
3	2.016	1.016	RR	CELL	1	IOC_X0_Y6_N0	clk|combout
4	2.148	0.132	RR	IC	1	CLKCTRL_G2	clk~clkctrl|inclk[0]
5	2.148	0.000	RR	CELL	39	CLKCTRL_G2	clk~clkctrl|outclk
6	2.908	0.760	RR	IC	1	LCFF_X10_Y8_N29	control|state.S5|clk

Path #1: Setup slack is -7.044 (VIOLATED)

Path Summary | Statistics | Data Path | Waveform



- Results of TimeQuest with 8.5ns clock period. Received a skew of 0.480, therefore the timing requirements are met. Resulting in all data being received **before** it is required.

Setup: clk

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.480	CU:control|state.S6	DP:datapath|Regfile:RF|REG_F~7	clk	clk	8.500	-0.003	8.055
2	0.482	CU:control|state.S6	DP:datapath|Regfile:RF|REG_F~31	clk	clk	8.500	-0.003	8.053
3	0.583	CU:control|state.S6	CU:control|state.S5	clk	clk	8.500	0.000	7.955
4	0.583	CU:control|state.S6	CU:control|state.S4	clk	clk	8.500	0.000	7.955
5	0.613	CU:control|state.S6	DP:datapath|Regfile:RF|REG_F~7	clk	clk	8.500	-0.003	7.922
6	0.615	CU:control|state.S6	DP:datapath|Regfile:RF|REG_F~31	clk	clk	8.500	-0.003	7.920
7	0.624	CU:control|state.S6	DP:datapath|Regfile:RF|REG_F~30	clk	clk	8.500	-0.003	7.911
8	0.624	CU:control|state.S3	DP:datapath|Regfile:RF|REG_F~7	clk	clk	8.500	-0.003	7.911

Path #1: Setup slack is 0.480

Path Summary | Statistics | Data Path | Waveform

Data Arrival Path

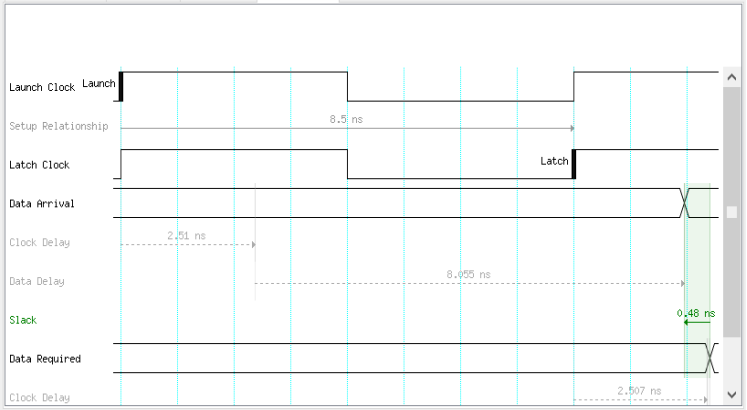
	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	2.510	2.510					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1	PIN_H2	clk
3	1.016	1.016	RR	CELL	1	IOC_X0_Y6_N0	clk|combout
4	1.148	0.132	RR	IC	1	CLKCTRL_G2	clk~clkctrl|inclk[0]
5	1.148	0.000	RR	CELL	39	CLKCTRL_G2	clk~clkctrl|outclk
6	1.908	0.760	RR	IC	1	LCFF_X10_Y8_N1	control|state.S6|clk

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	8.500	8.500					latch edge time
2	11.007	2.507					clock path
1	8.500	0.000					source latency
2	8.500	0.000			1	PIN_H2	clk
3	9.516	1.016	RR	CELL	1	IOC_X0_Y6_N0	clk|combout
4	9.648	0.132	RR	IC	1	CLKCTRL_G2	clk~clkctrl|inclk[0]
5	9.648	0.000	RR	CELL	39	CLKCTRL_G2	clk~clkctrl|outclk
6	10.405	0.757	RR	IC	1	LCFF_X14_Y8_N27	datapath|RF|REG_F~7|clk

Path #1: Setup slack is 0.480

Path Summary | Statistics | Data Path | Waveform



4. Questions

1. Find out the information about the TimeQuest analyzer.

- TimeQuest analyzer has many powerful tools to specify timing constraints for simple and complex digital designs. The TimeQuest timing analyzer compilation report shows you various tools and setting which you can select and view. For this lab experiment we selected the clock and viewed the TimeQuest GUI. The TimeQuest GUI allowed us to see our slack, from node to node, setup relationship, latch clock, arrival of data, clock delay, clock skew, data delay and when data is required. One also has the ability to see any desired path from the design and calculate the proper clock period and frequency in order for the data to be successfully received before being required for that path.

The TimeQuest analyzer GUI allowed us to visually see when our data is being received and when it is required. We used the slack timing from the TimeQuest analyzer report to see exactly what the clock period should be in order to fulfil the timing requirements for our design. Our original clock period was 1ns but after looking at the TimeQuest analyzer, we concluded that the proper clock period was 8.5ns for our design to fulfil the timing requirements.

2. List the purposes of initializing the memory in Quartus II

- It is important to initialize memory in Quartus II because it allows the storage of the data in the memory to be neater. Another important purpose to initialize memory in Quartus II is so that when you want to see how where in memory data is being stored, it is easier to reference when you initialize the memory at a specific location.

5. Conclusions

- For this lab experiment we used the TimeQuest analyzer and altered various components of the GDP in order to implement the algorithm that calculates the greatest common divisor. A problem that we encountered was that we did not finish within the allocated time, so we had to meet up outside of the lab to finish. Another encountered problem was trying to find the proper amount of delay within the test bench for verifying if our algorithm worked. But after messing around with the different delay amounts, we were able to properly verify the testbench with the waveform. Overall this lab taught us how to use the TimeQuest analyzer and how to manipulate the GDP and control unit in order to implement the algorithm that calculates the greatest common divisor.

Class:	CPE-300L	Semester:	Spring 2020

Points		Document author:	Brysen Kokubun
		Author's email:	Brysenkokubun96@gmail.com
		Document topic:	Prelab 8

Instructor's comments:

Introduction / Theory of operation

- For this lab experiment we will be using the GDP that was created in lab experiment 6, to implement the algorithm that calculates the greatest common divisor of 2 natural numbers. We will also be learning how to set up timing constraints and how to obtain timing information with the TimeQuest analyzer.

Prelab main content

1. Screenshots of the timing report.

Setup: clock

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-1.997	reg_A[1]	reg_sum[7]	clock	clock	1.000	-0.065	2.927
2	-1.992	reg_A[1]	reg_sum[7]	clock	clock	1.000	-0.065	2.922
3	-1.992	reg_B[1]	reg_sum[7]	clock	clock	1.000	-0.065	2.922
4	-1.991	reg_A[1]	reg_sum[7]	clock	clock	1.000	-0.065	2.921
5	-1.987	reg_B[1]	reg_sum[7]	clock	clock	1.000	-0.065	2.917
6	-1.986	reg_B[1]	reg_sum[7]	clock	clock	1.000	-0.065	2.916

Path #1: Setup slack is -1.997 (VIOLATED) Path #1: Setup slack is -1.997 (VIOLATED)

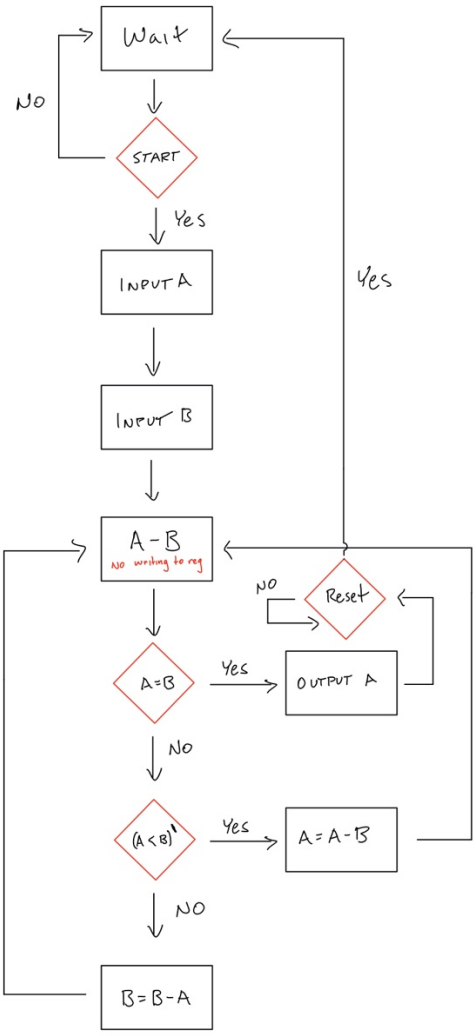
Setup: clock

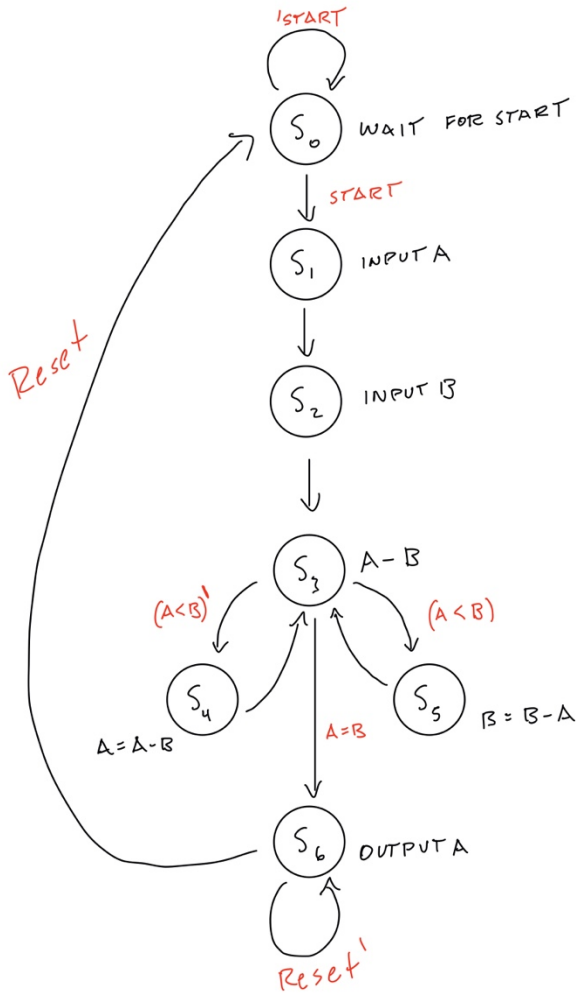
Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.691	reg_B[1]	reg_sum[9]~reg0	clock	clock	4.000	0.324	3.630
2	0.697	reg_B[1]	reg_sum[9]~reg0	clock	clock	4.000	0.324	3.624
3	0.698	reg_B[2]	reg_sum[8]~reg0	clock	clock	4.000	0.324	3.623
4	0.698	reg_B[1]	reg_sum[9]~reg0	clock	clock	4.000	0.324	3.623
5	0.698	reg_B[2]	reg_sum[8]~reg0	clock	clock	4.000	0.324	3.623
6	0.701	reg_A[1]	reg_sum[9]~reg0	clock	clock	4.000	0.324	3.620
7	0.707	reg_A[1]	reg_sum[9]~reg0	clock	clock	4.000	0.324	3.614
8	0.708	reg_A[1]	reg_sum[9]~reg0	clock	clock	4.000	0.324	3.613

Path #1: Setup slack is 0.691

2. *Flowchart, state graph and control words for the algorithm*





state	IE	WE	WA(10)	RAE	RAA(10)	RBE	RBA(10)	ALU(210)	SH(10)	OE	
s0	0	1	00	1	00	1	00	101	xx	0	//Wait for start
s1	1	1	01	0	xx	0	xx	000	xx	0	//Input A
s2	1	1	10	0	xx	0	xx	xxx	xx	0	//Input B
s3	0	0	xx	1	01	1	10	101	00	0	//A-B no write to Reg
s4	0	1	01	1	01	1	10	101	00	0	//A = A-B
s5	0	1	10	1	10	1	01	101	00	0	//B = B-A
s6	x	0	xx	1	01	0	xx	000	00	1	//Output A