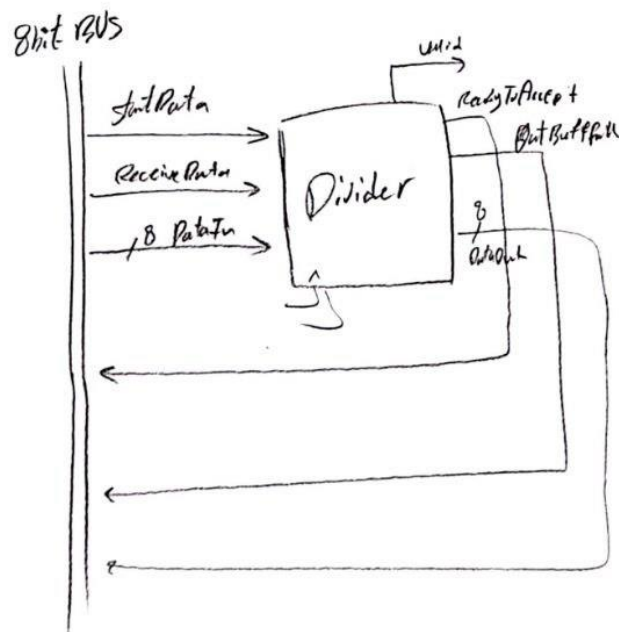
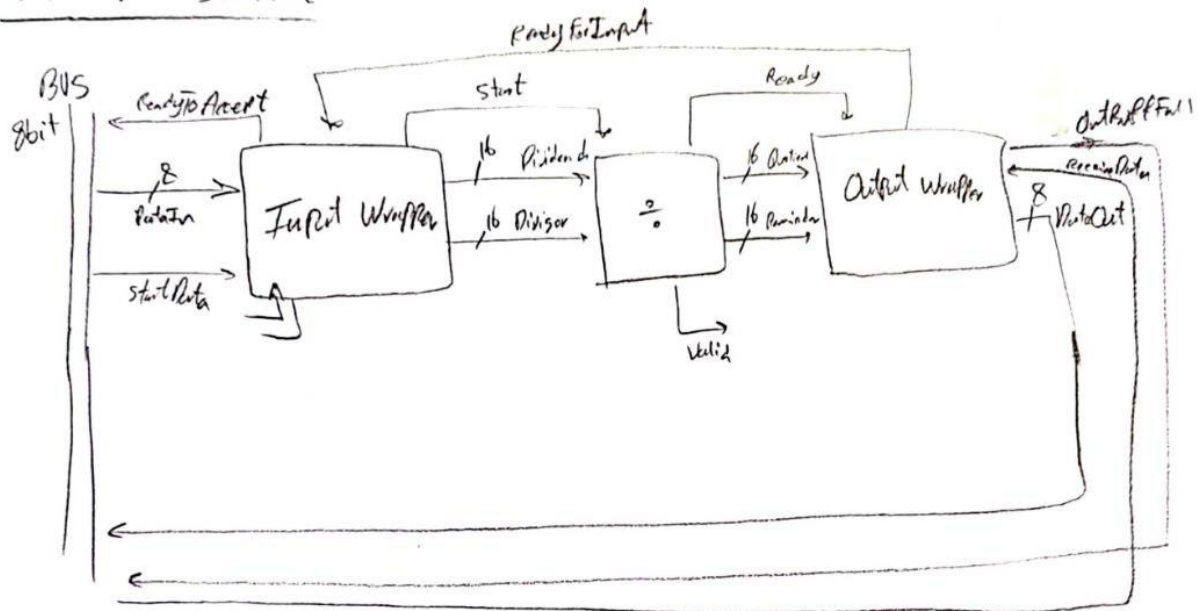


# OVERVIEW:

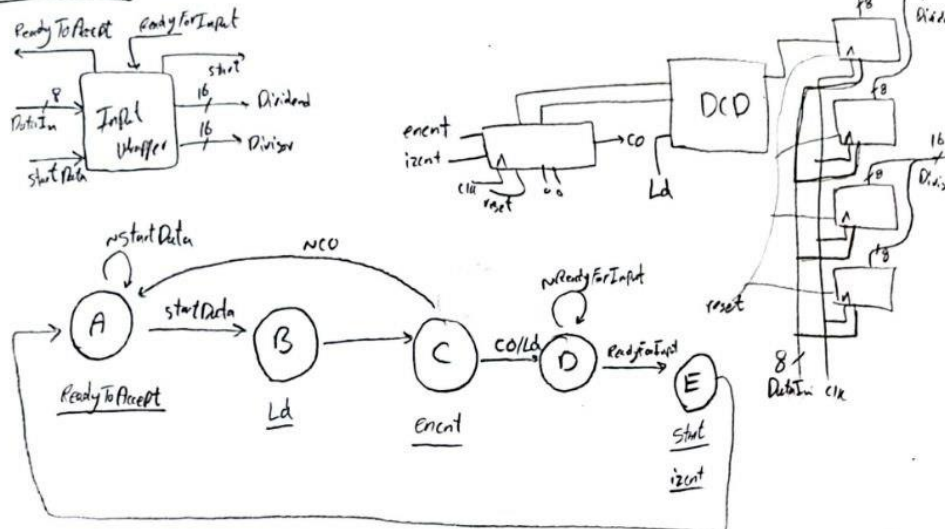
Divider With Interface



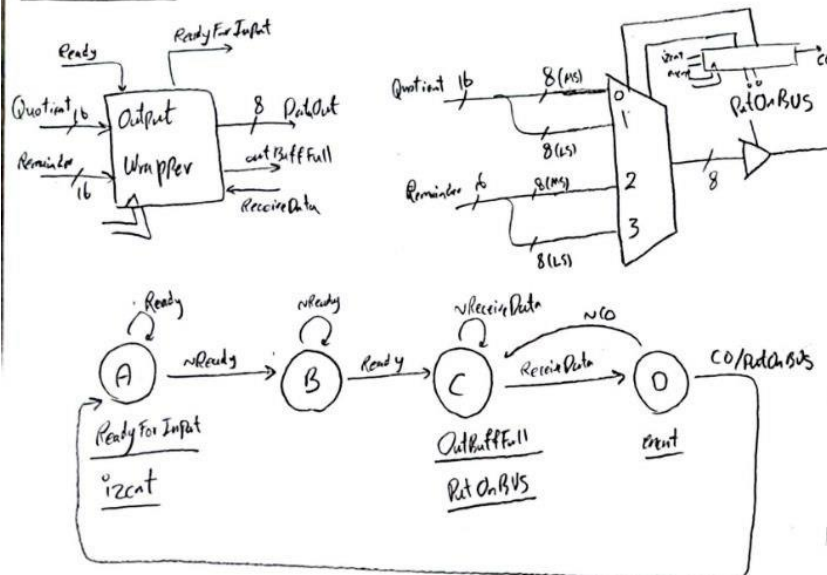
# DESIGN- DATA PATH AND CONTROLLER

Adin Tarassoli 810102543 CA6

## Input Wrapper



## Output Wrapper



# INPUT WRAPPER:

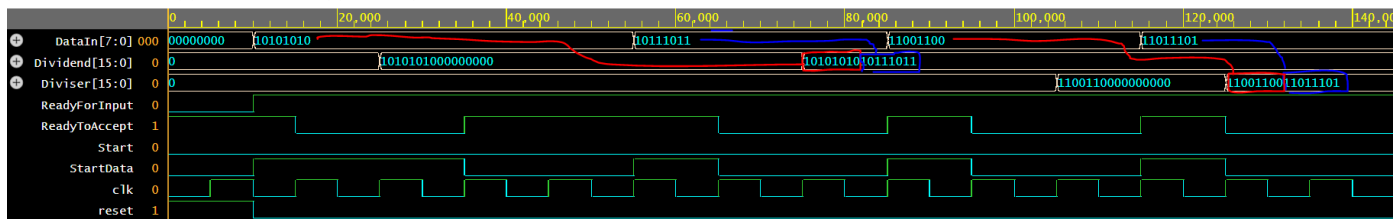
## RESULTS

First 8-bit: Dividend Most Significant

Second 8-bit: Dividend Least Significant

Third 8-bit: Divisor Most Significant

Forth 8-bit: Divisor Least Significant



## Code:

```
1  module Counter2bit (input enCnt, reset, izcnt, clk, output logic [1:0] cnt, output Co);
2      assign Co = &cnt;
3      always @(posedge clk or posedge reset) begin
4          if (reset) cnt <= 2'b0;
5          else if(izcnt) cnt <= 2'b0;
6          else if (enCnt) cnt <= cnt + 1'b1;
7      end
8  endmodule
9
10 module Dcd2to4 (input en, reset, input [1:0] in, output logic [3:0] out);
11     always @(reset, en, in) begin
12         out = 4'b0;
13         if (reset) out = 4'b0;
14         else if(en) begin
15             case(in)
16                 2'd0: out = 4'b0001;
17                 2'd1: out = 4'b0010;
18                 2'd2: out = 4'b0100;
19                 2'd3: out = 4'b1000;
20                 default: out = 4'b0;
21             endcase
22         end
23         else out = 4'b0;
24     end
25 endmodule
26
27 module Register8bit (input [7:0] in, input clk, load, reset, output logic [7:0] out);
28     always @(posedge clk or posedge reset) begin
29         if (reset) out <= 8'b0;
30         else if (load) out <= in;
31     end
32 endmodule
```

```

34 module Input_Wrapper_Datapath(input encnt,izcnt,reset,load,clk, input [7:0]DataIn,
35     output [15:0] Dividend,Diviser, output Co);
36     wire [1:0] cnt;
37     Counter2bit counter(.enCnt(encnt), .reset(reset), .izcnt(izcnt), .clk(clk), .cnt(cnt), .Co(Co));
38     wire [3:0] registers_load;
39     Dcd2to4 dcd(.en(load), .reset(reset), .in(cnt), .out(registers_load));
40     wire [7:0] Dividend_8bitMS, Dividend_8bitLS, Diviser_8bitMS, Diviser_8bitLS;
41     Register8bit rg1(.in(DataIn), .clk(clk), .load(registers_load[0]), .reset(reset), .out(Dividend_8bitMS));
42     Register8bit rg2(.in(DataIn), .clk(clk), .load(registers_load[1]), .reset(reset), .out(Dividend_8bitLS));
43     Register8bit rg3(.in(DataIn), .clk(clk), .load(registers_load[2]), .reset(reset), .out(Diviser_8bitMS));
44     Register8bit rg4(.in(DataIn), .clk(clk), .load(registers_load[3]), .reset(reset), .out(Diviser_8bitLS));
45     assign Dividend = {Dividend_8bitMS, Dividend_8bitLS};
46     assign Diviser = {Diviser_8bitMS, Diviser_8bitLS};
47 endmodule
48
49 module Input_Wrapper_Controller(input ReadyForInput,StartData,Co,clk,reset,
50     output logic ReadyToAccept, Start, encnt,izcnt, output load);
51     typedef enum {A,B,C,D,E} states;
52     states state,next_state;
53     always @(posedge clk or posedge reset) begin
54         if(reset) state <= A;
55         else state <= next_state;
56     end
57     assign load = (Co || (state == B)) ? 1'b1 : 1'b0;
58     always @(state, StartData, Co, ReadyForInput) begin
59         {ReadyToAccept, encnt, Start, izcnt} = 0;
60         //next_state = state;
61     case (state)
62     A: begin
63         next_state = StartData ? B : A;
64         ReadyToAccept = 1;
65     end
66     B: begin
67         next_state = C;
68     end
69     C: begin
70         encnt = 1;
71         next_state = Co ? D : A;
72     end
73     D: begin
74         next_state = ReadyForInput ? E : D;
75     end
76     E: begin
77         Start = 1;
78         izcnt = 1;
79         next_state = A;
80     end
81     endcase
82 end
83 endmodule
84
85
86

```

```

87 module Input_Wrapper(input ReadyForInput,StartData, reset,clk, input [7:0]DataIn,
88     output [15:0] Dividend,Diviser, output ReadyToAccept, Start);
89
90     wire encnt, izcnt, load, Co;
91     Input_Wrapper_Datapath Dp(.encnt(encnt), .izcnt(izcnt), .reset(reset), .load(load), .clk(clk),
92         .DataIn(DataIn), .Dividend(Dividend), .Diviser(Diviser), .Co(Co));
93
94     Input_Wrapper_Controller controller(.ReadyForInput(ReadyForInput), .StartData(StartData), .Co(Co), .clk(clk), .reset(reset),
95         .ReadyToAccept(ReadyToAccept), .Start(Start), .encnt(encnt), .izcnt(izcnt), .load(load));
96 endmodule

```

## Testbench:

```
2  module Input_Wrapper_tb;
3      reg ReadyForInput, StartData, reset, clk;
4      reg [7:0] DataIn;
5      wire [15:0] Dividend, Diviser;
6      wire ReadyToAccept, Start;
7      Input_Wrapper dut (
8          .ReadyForInput(ReadyForInput),
9          .StartData(StartData),
10         .reset(reset),
11         .clk(clk),
12         .DataIn(DataIn),
13         .Dividend(Dividend),
14         .Diviser(Diviser),
15         .ReadyToAccept(ReadyToAccept),
16         .Start(Start)
17     );
18
19     initial begin
20         clk = 0;
21         forever #5 clk = ~clk;
22     end
23
24     initial begin
25         ReadyForInput = 0; StartData = 0; reset = 1; DataIn = 8'h00;
26         #10 reset = 0;
27
28         ReadyForInput = 1; StartData = 1; DataIn = 8'hAA;
29         #10;
30         wait (ReadyToAccept == 1);
31         StartData = 0; #20;
32
33         StartData = 1; DataIn = 8'hBB;
34         #10;
35         wait (ReadyToAccept == 1);
36         StartData = 0; #20;
37
38         StartData = 1; DataIn = 8'hCC;
39         #10;
40         wait (ReadyToAccept == 1);
41         StartData = 0; #20;
42
43         StartData = 1; DataIn = 8'hDD;
44         #10;
45         wait (ReadyToAccept == 1);
46         StartData = 0; #20;
47         $stop;
48     end
49 endmodule
```

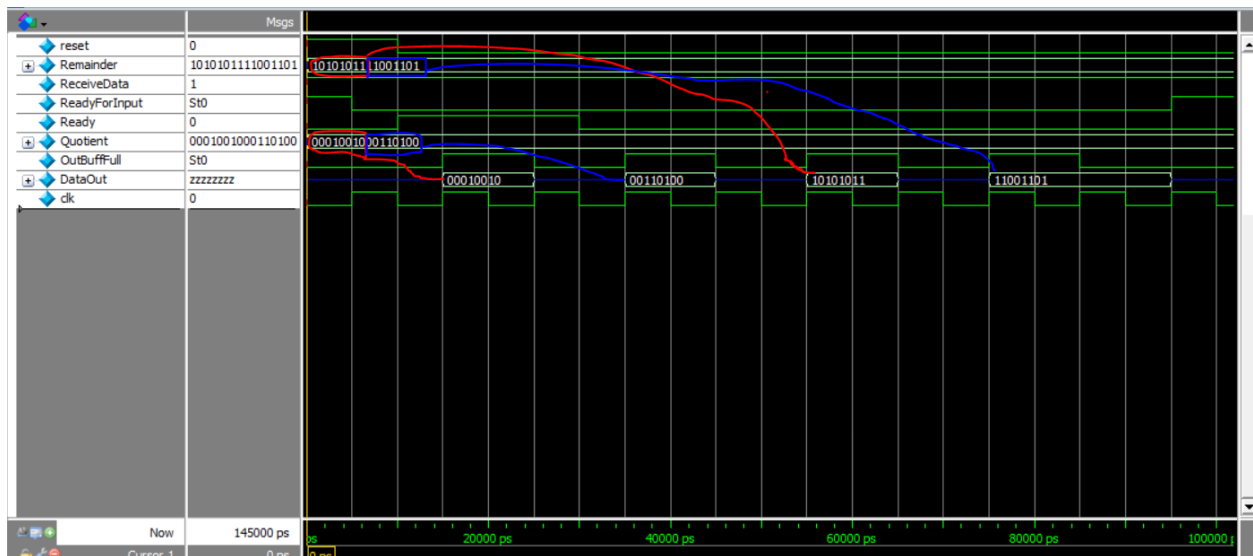
# OUTPUT WRAPPER:

First 8-bit: Quotient Most Significant

Second 8-bit: Quotient Least Significant

Third 8-bit: Remainder Most Significant

Forth 8-bit: Remainder Least Significant



Code:

```
1 module Counter2bit (input enCnt, reset, izcnt, clk, output logic [1:0] cnt, output Co);
2     assign Co = &cnt;
3     always @(posedge clk or posedge reset) begin
4         if (reset) cnt <= 2'b0;
5         else if (izcnt) cnt <= 2'b0;
6         else if (enCnt) cnt <= cnt + 1'b1;
7     end
8 endmodule
9
10 module Multiplexer4to1(input [1:0] select, input [31:0] in, output logic [7:0] out);
11     always @(select, in) begin
12         out = 7'b0;
13         case (select)
14             2'd0: out = in[31:24];
15             2'd1: out = in[23:16];
16             2'd2: out = in[15:8];
17             2'd3: out = in[7:0];
18             default: out = 7'b0;
19         endcase
20     end
21 endmodule
22
23 module Buffif1(input [7:0] in, input select, output [7:0] out);
24     assign out = select ? in : 8'bzzzzzzzz;
25 endmodule
26
27 module Output_Wrapper_Datapath(input [15:0] Quotient, Remainder, input PutOnBus, enCnt, izcnt, reset, clk, output [7:0] out, output Co);
28     wire [1:0] cnt;
29     Counter2bit counter(.enCnt(enCnt), .reset(reset), .izcnt(izcnt), .clk(clk), .cnt(cnt), .Co(Co));
30     wire [7:0] multiplexer_output;
31     Multiplexer4to1 multiplexer(.select(cnt), .in({Quotient, Remainder}), .out(multiplexer_output));
32     Buffif1 buffer(.in(multiplexer_output), .select(PutOnBus), .out(out));
33 endmodule
34
```

```

35 module Output_Wrapper_Controller(input Ready,ReceiveData,Co,clk,reset,
36     output logic ReadyForInput, OutBuffFull, encnt,izcnt,output PutOnBus);
37     typedef enum {A,B,C,D} states;
38     states state,next_state;
39     always @(posedge clk or posedge reset) begin
40         if(reset) state <= A;
41         else state <= next_state;
42     end
43     assign PutOnBus = (Co || (state == C)) ? 1'b1 : 1'b0;
44     always @(state, Ready, ReceiveData, Co) begin
45         {ReadyForInput, OutBuffFull, encnt, izcnt} = 0;
46         case (state)
47             A: begin
48                 ReadyForInput = 1;
49                 izcnt = 1;
50                 next_state = Ready ? A : B;
51             end
52             B: begin
53                 next_state = Ready ? C : B;
54             end
55             C: begin
56                 OutBuffFull = 1;
57                 next_state = ReceiveData ? D : C;
58             end
59             D: begin
60                 encnt = 1;
61                 next_state = Co ? A : C;
62             end
63         endcase
64     end
65 endmodule
66
67 module Output_Wrapper(input Ready, ReceiveData, reset,clk,input [15:0] Quotient,Remainder,
68     output ReadyForInput,OutBuffFull, output [7:0]DataOut);
69     wire encnt, izcnt, PutOnBus, Co;
70     Output_Wrapper_Datapath Dp(.Quotient(Quotient), .Remainder(Remainder),.encnt(encnt),
71         .izcnt(izcnt),.reset(reset),.clk(clk), .PutOnBus(PutOnBus),.out(DataOut), .Co(Co));
72     Output_Wrapper_Controller controller(.Ready(Ready), .ReceiveData(ReceiveData), .Co(Co), .clk(clk),
73         .reset(reset), .ReadyForInput(ReadyForInput), .OutBuffFull(OutBuffFull),
74         .encnt(encnt), .izcnt(izcnt), .PutOnBus(PutOnBus));
75 endmodule

```

## Testbench:

```
1  `timescale 1ps/1ps
2  module Divider_With_Interface_tb;
3      reg StartData, reset, clk, ReceiveData;
4      reg [7:0] DataIn;
5      wire ReadyToAccept1, Valid1, OutBuffFull1;
6      wire [7:0] DataOut1;
7      wire ReadyToAccept, Valid, OutBuffFull;
8      wire [7:0] DataOut;
9
10     Divider_With_Interface_post DUT (
11         .StartData(StartData),
12         .reset(reset),
13         .clk(clk),
14         .ReceiveData(ReceiveData),
15         .DataIn(DataIn),
16         .ReadyToAccept(ReadyToAccept1),
17         .Valid(Valid1),
18         .OutBuffFull(OutBuffFull1),
19         .DataOut(DataOut1)
20     );
21
22     Divider_With_Interface_pre DUT_pre (
23         .StartData(StartData),
24         .reset(reset),
25         .clk(clk),
26         .ReceiveData(ReceiveData),
27         .DataIn(DataIn),
28         .ReadyToAccept(ReadyToAccept),
29         .Valid(Valid),
30         .OutBuffFull(OutBuffFull),
31         .DataOut(DataOut)
32     );
33     initial begin
34         clk = 0;
35         forever #5 clk = ~clk;
36     end
37
```



```

2  module Divider_With_Interface_tb;
37
38      initial begin
39          StartData = 0; reset = 1; DataIn = 8'b0;
40          #10 reset = 0;
41          ReceiveData = 0;
42          #20
43
44          StartData = 1; DataIn = 8'b01010110;
45          #10;
46          wait (ReadyToAccept == 1);
47          StartData = 0; #20;
48
49          StartData = 1; DataIn = 8'b10011101;
50          #10;
51          wait (ReadyToAccept == 1);
52          StartData = 0; #20;
53
54          StartData = 1; DataIn = 8'b00000101;
55          #10;
56          wait (ReadyToAccept == 1);
57          StartData = 0; #20;
58
59          StartData = 1; DataIn = 8'b10000101;
60          #10;
61          wait (ReadyToAccept == 1);
62          StartData = 0;
63          #50;
64
65          repeat (4) begin
66              wait(OutBuffFull);
67              ReceiveData = 1;
68              #20;
69              ReceiveData = 0;
70              #20;
71          end
72
73          #50;
74          $stop;

```

# DIVIDER WITH RTL INTERACTION:

Suppose we want to divide 22173 by 1413:

01010110 10011101 = 22173

00000101 10000101 = 1413

Q = 15 = 00000000 00001111

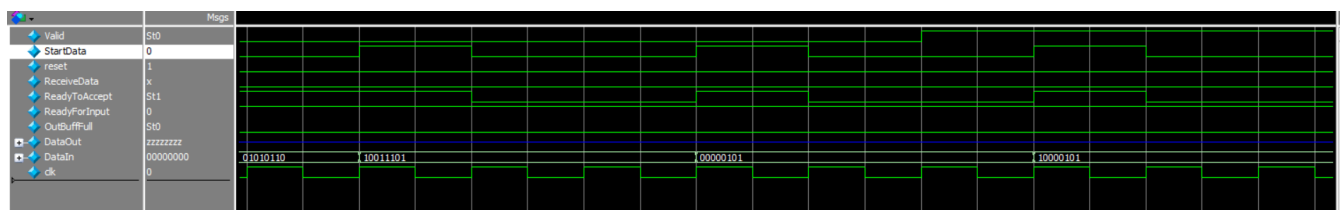
R = 978 = 00000011 11010010

DataIn: 01010110, 10011101, 00000101, 10000101

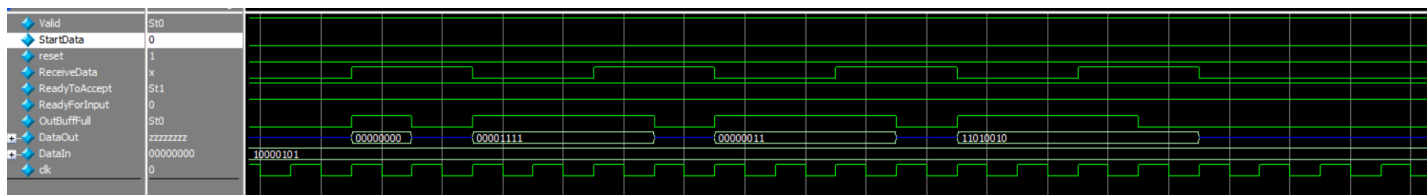
DataOut: 00000000, 00001111, 00000011, 11010010 (and z between them)

## RESULTS

### Sending Data:



### Receiving Data:



## Code:

```
1 module Divider_With_Interface(input StartData, reset, clk, ReceiveData, input [7:0]DataIn,
2   output ReadyToAccept, Valid, OutBuffFull, output [7:0]DataOut);
3   wire ReadyForInput, Start, Ready;
4   wire [15:0]Dividend, Divisor, Quotient, Remainder;
5   Input_Wrapper input_wrapper(.ReadyForInput(ReadyForInput), .StartData(StartData), .clk(clk), .reset(reset),
6   .DataIn(DataIn), .Dividend(Dividend), .Divisor(Divisor), .ReadyToAccept(ReadyToAccept), .Start(Start));
7   Divider divider(.Dividend(Dividend), .Divisor(Divisor), .clk(clk), .start(Start),
8   .rst(reset), .R(Remainder), .Q(Quotient), .ready(Ready), .valid(Valid));
9   Output_Wrapper output_wrapper(.Ready(Ready), .ReceiveData(ReceiveData), .reset(reset), .clk(clk),
10  .Quotient(Quotient), .Remainder(Remainder), .ReadyForInput(ReadyForInput), .OutBuffFull(OutBuffFull), .DataOut(DataOut));
11 endmodule
```

# Testbench:

```
1 module Divider_With_Interface_tb;
2   wire ReadyToAccept1, Valid1, OutBuffFull1;
3   wire [7:0] DataOut1;
4   wire ReadyToAccept, Valid, OutBuffFull;
5   wire [7:0] DataOut;
6
7   Divider_With_Interface_post DUT (
8     .StartData(StartData),
9     .reset(reset),
10    .clk(clk),
11    .ReceiveData(ReceiveData),
12    .DataIn(DataIn),
13    .ReadyToAccept(ReadyToAccept1),
14    .Valid(Valid1),
15    .OutBuffFull(OutBuffFull1),
16    .DataOut(DataOut1)
17  );
18
19  Divider_With_Interface_pre DUT_pre (
20    .StartData(StartData),
21    .reset(reset),
22    .clk(clk),
23    .ReceiveData(ReceiveData),
24    .DataIn(DataIn),
25    .ReadyToAccept(ReadyToAccept),
26    .Valid(Valid),
27    .OutBuffFull(OutBuffFull),
28    .DataOut(DataOut)
29  );
30
31  initial begin
32    clk = 0;
33    forever #5 clk = ~clk;
34  end
35
36  initial begin
37    StartData = 0; reset = 1; DataIn = 8'b0;
38    #10 reset = 0;
39    ReceiveData = 0;
40    #20
41
42    StartData = 1; DataIn = 8'b01010110;
43    #10;
44    wait (ReadyToAccept == 1);
45    StartData = 0; #20;
46
47    StartData = 1; DataIn = 8'b10011101;
48    #10;
49    wait (ReadyToAccept == 1);
50    StartData = 0; #20;
51
52    StartData = 1; DataIn = 8'b00000101;
53    #10;
54    wait (ReadyToAccept == 1);
55    StartData = 0; #20;
56
57    StartData = 1; DataIn = 8'b10000101;
58    #10;
59    wait (ReadyToAccept == 1);
60    StartData = 0;
61    #50;
62
63    repeat (4) begin
64      wait(OutBuffFull);
65      ReceiveData = 1;
66      #20;
67      ReceiveData = 0;
68      #20;
69    end
70
71    #50;
72  end
```

# SYNTHESIS:

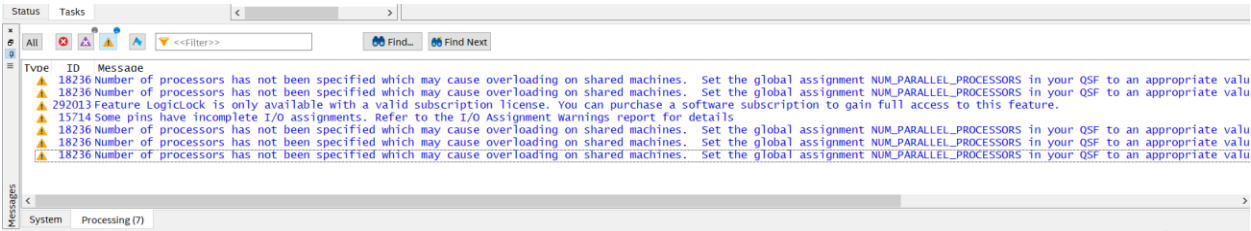
## Total of 95 registers:

Compilation Report - Divider_With_Interface_pre		Divider_With_Interface.sv	
Table of Contents		Flow Summary	
Flow Summary		<<Filter>>	
Flow Settings		Flow Status	
Flow Non-Default Global Settings		Successful - Sat Feb 01 16:29:28 2025	
Flow Elapsed Time		Quartus Prime Version	
Flow OS Summary		20.1.0 Build 711 06/05/2020 SJ Lite Edition	
Flow Log		Revision Name	
Analysis & Synthesis		Divider_With_Interface_pre	
Fitter		Top-level Entity Name	
Assembler		Divider_With_Interface_pre	
Timing Analyzer		Family	
EDA Netlist Writer		Cyclone IV E	
Flow Messages		Device	
Flow Suppressed Messages		EP4CE115F29I8L	
		Timing Models	
		Final	
		Total logic elements	
		151 / 114,480 ( < 1 % )	
		Total registers	
		95	
		Total pins	
		23 / 529 ( 4 % )	
		Total virtual pins	
		0	
		Total memory bits	
		0 / 3,981,312 ( 0 % )	
		Embedded Multiplier 9-bit elements	
		0 / 532 ( 0 % )	
		Total PLLs	
		0 / 4 ( 0 % )	

## Utilization of these registers:

	Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Memory Bits	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtua
1	▼ [Divider_With_Interface_pre]	138 (1)	95 (0)	0	0	0	0	23	0
1	▼ [Divider:divider]	96 (0)	54 (0)	0	0	0	0	0	0
1	▼ [Divider_Controller:controller_module]	11 (6)	6 (2)	0	0	0	0	0	0
1	[Counter:count_module]	5 (5)	4 (4)	0	0	0	0	0	0
2	▼ [Divider_Datapath:datapath_module]	85 (5)	48 (0)	0	0	0	0	0	0
1	[Absolute:Aabs]	18 (18)	0 (0)	0	0	0	0	0	0
2	[Left_Shift_Register:Areg]	14 (14)	16 (16)	0	0	0	0	0	0
3	[Left_Shift_Register:Temp_reg]	16 (16)	16 (16)	0	0	0	0	0	0
4	[Register:Breg]	16 (16)	16 (16)	0	0	0	0	0	0
5	[Subtract:sub]	16 (16)	0 (0)	0	0	0	0	0	0
2	▼ [Input_Wrapper:input_wrapper]	15 (0)	37 (0)	0	0	0	0	0	0
1	[Input_Wrapper_Controller:controller]	8 (8)	3 (3)	0	0	0	0	0	0
2	▼ [Input_Wrapper_Datapath:Dp]	7 (0)	34 (0)	0	0	0	0	0	0
1	[Counter2bit:counter]	3 (3)	2 (2)	0	0	0	0	0	0
2	[Dcd2to4:dcd]	4 (4)	0 (0)	0	0	0	0	0	0
3	[Register8bitrg1]	0 (0)	8 (8)	0	0	0	0	0	0
4	[Register8bitrg2]	0 (0)	8 (8)	0	0	0	0	0	0
5	[Register8bitrg3]	0 (0)	8 (8)	0	0	0	0	0	0
6	[Register8bitrg4]	0 (0)	8 (8)	0	0	0	0	0	0
3	▼ [Output_Wrapper:output_wrapper]	26 (0)	4 (0)	0	0	0	0	0	0
1	[Output_Wrapper_Controller:controller]	7 (7)	2 (2)	0	0	0	0	0	0

# Zero important warnings

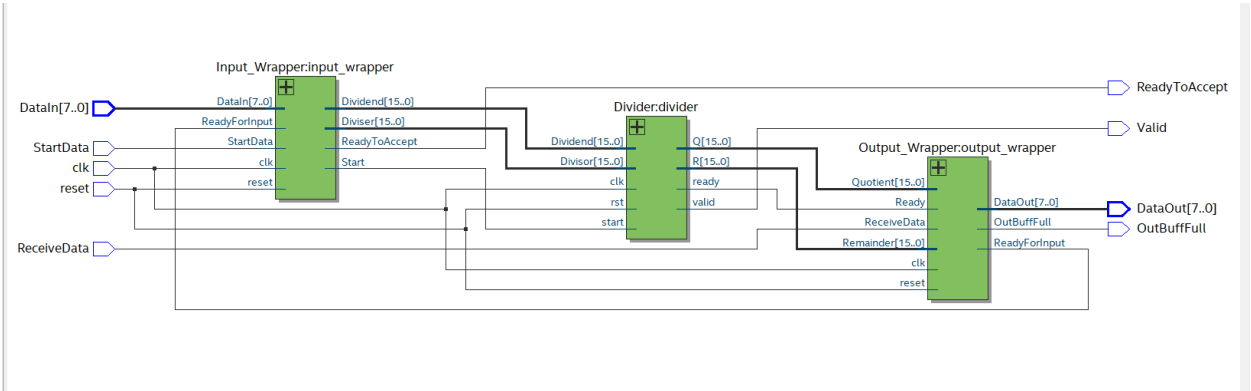


# Maximum Frequency of clock:

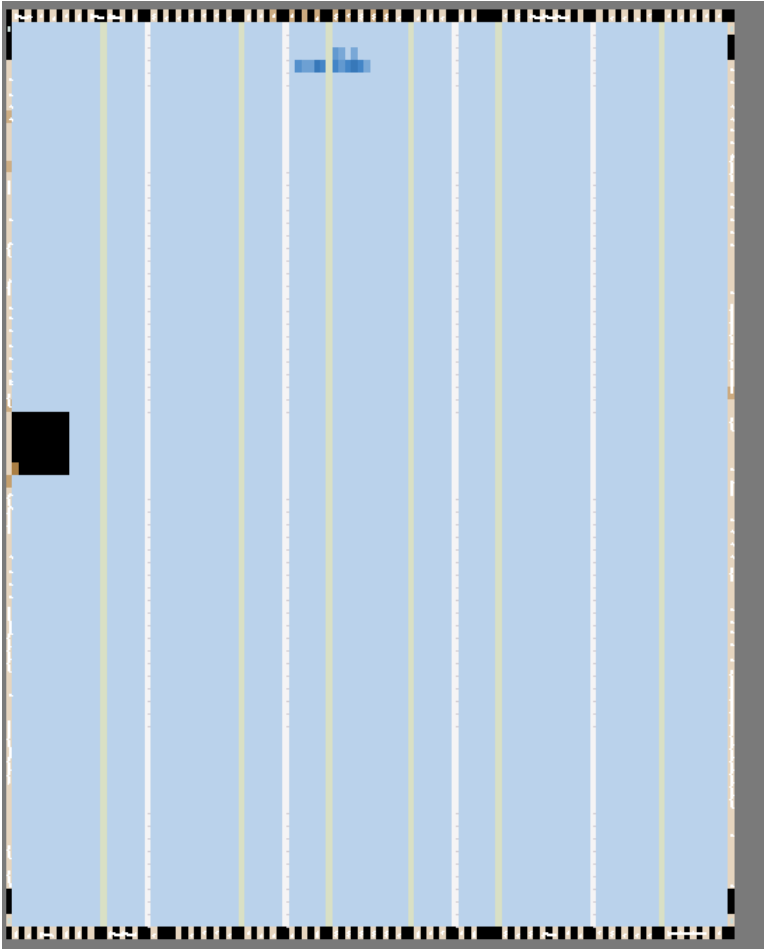
<<Filter>>

	Fmax	Restricted Fmax	Clock Name
1	227.48 MHz	227.48 MHz	clk

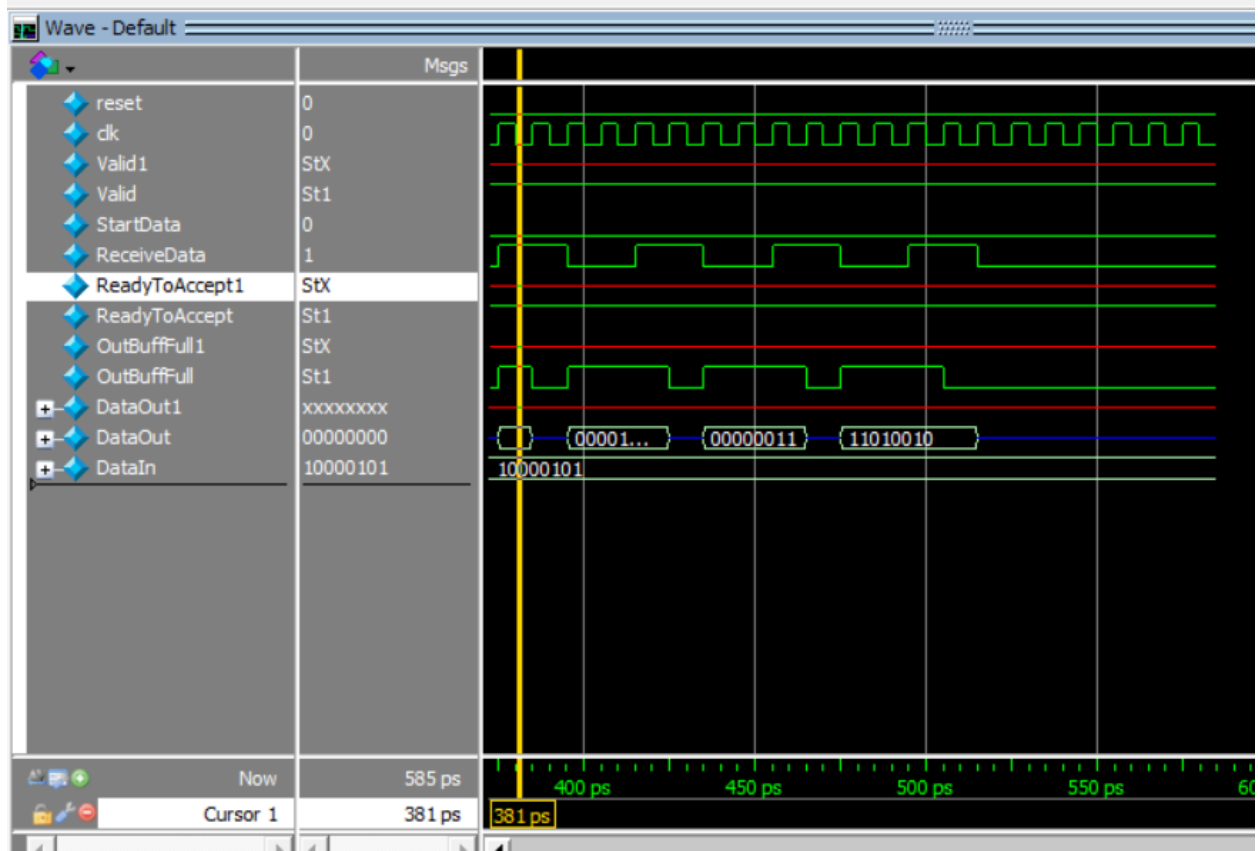
# RTL OVERVIEW:



**CHIP UTILIZATION:**



## POST SYNTHESIS SIMULATION:



For whatever reason I can't seem to find why the output is x, presynthesis is correct and no warnings in the syntheses part, Ms.Zahra Mahdavi and I spent 5 hours debugging it and couldn't find out why 😊

Synthesis and post synthesis simulation of the divider(CA5) was correct.