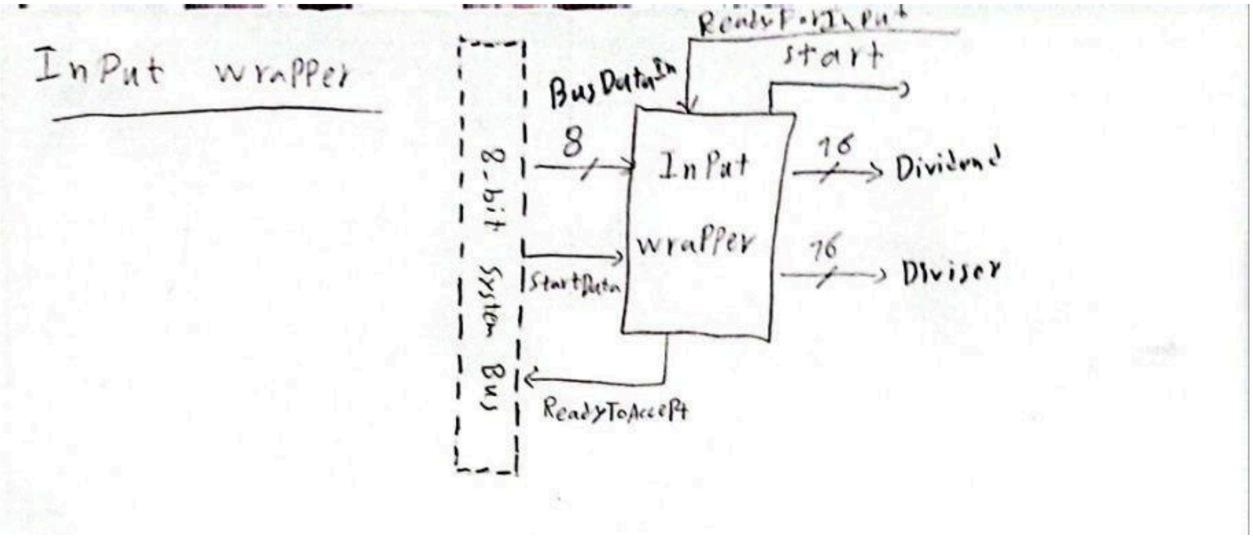
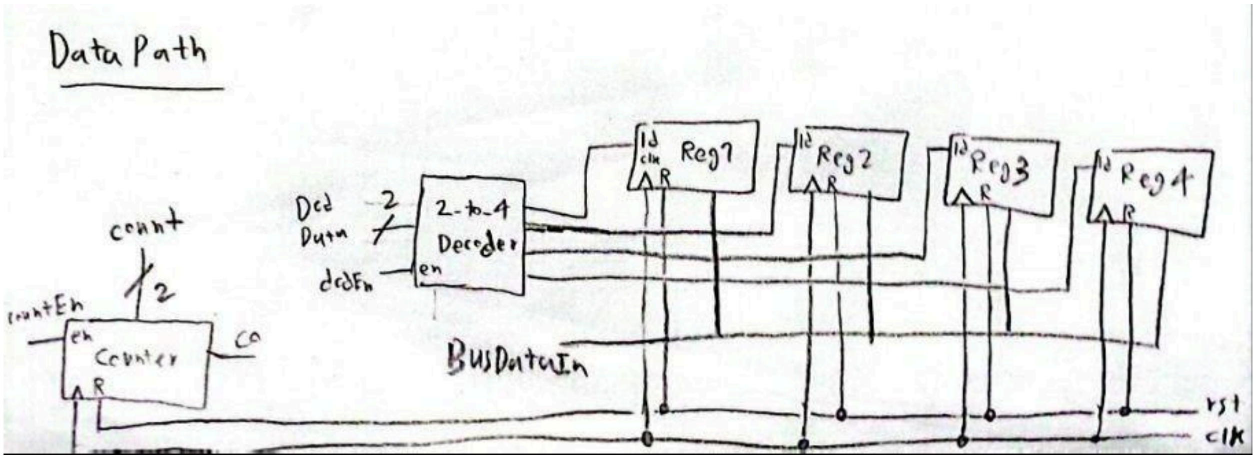
Erfan Falahati SID: 810102491

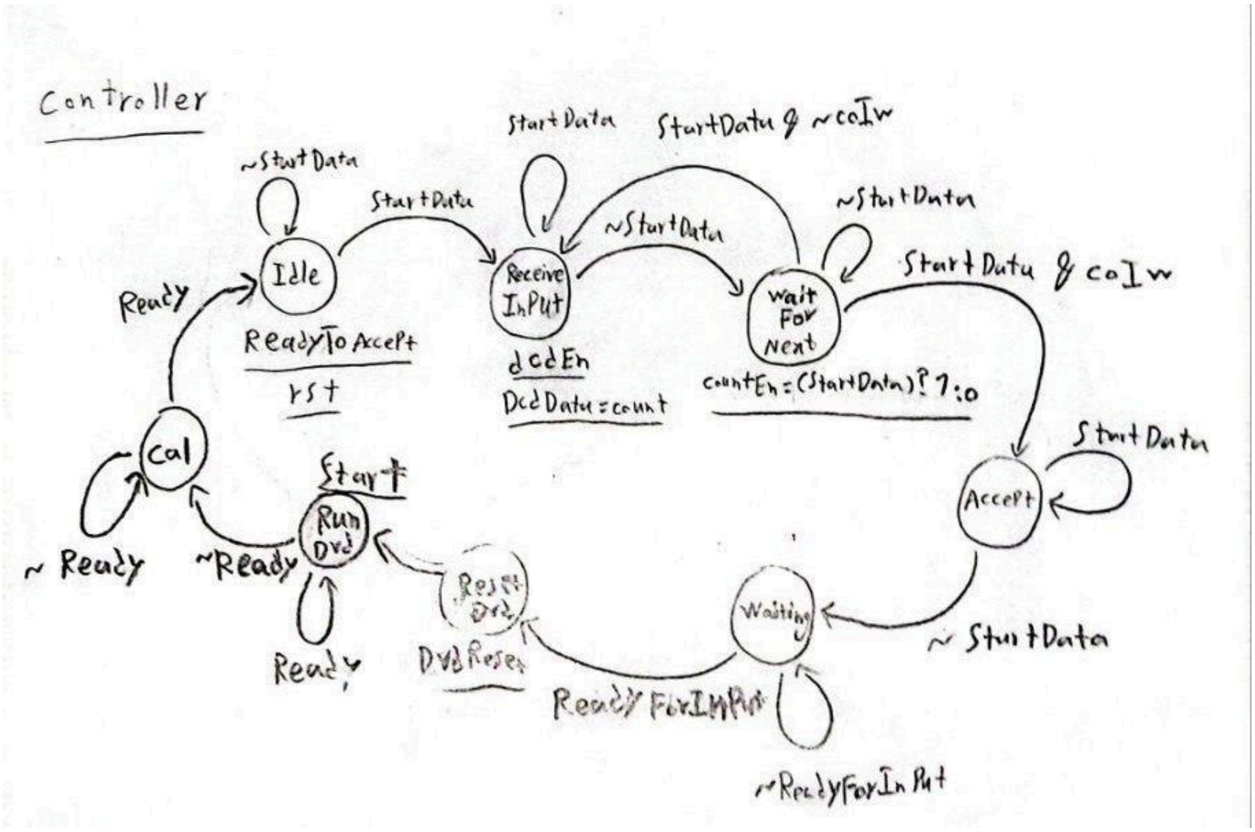
CA#6

RTL Accelerator

Input Wrapper

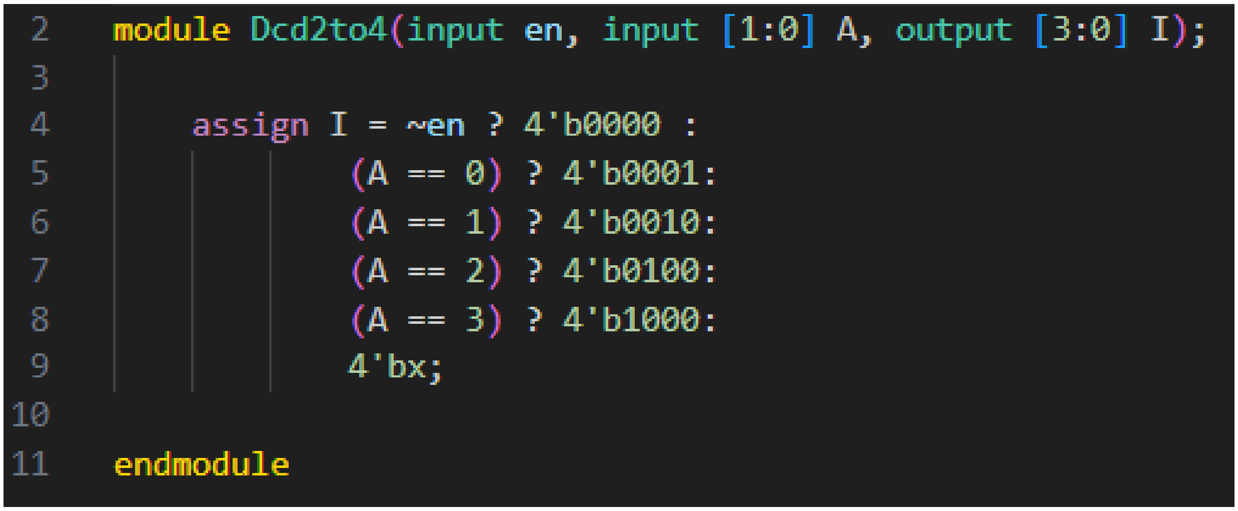






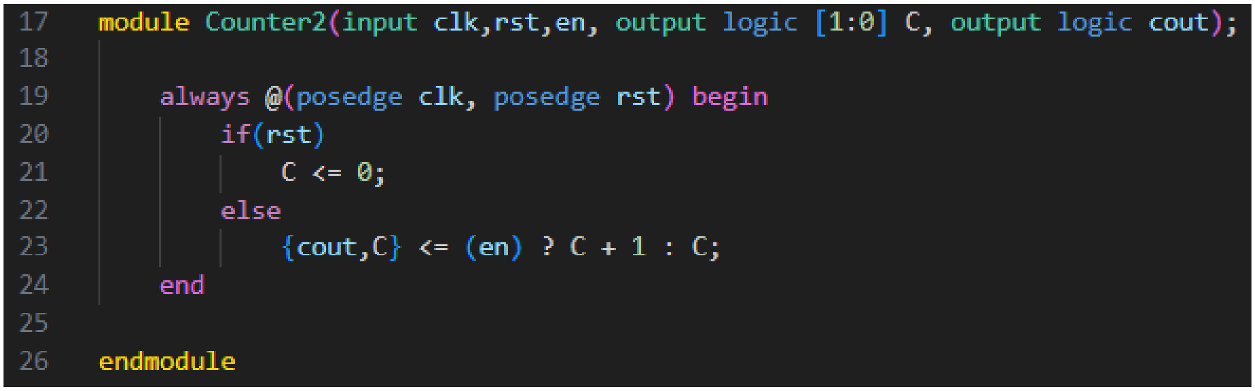
System Verilog

Decoder



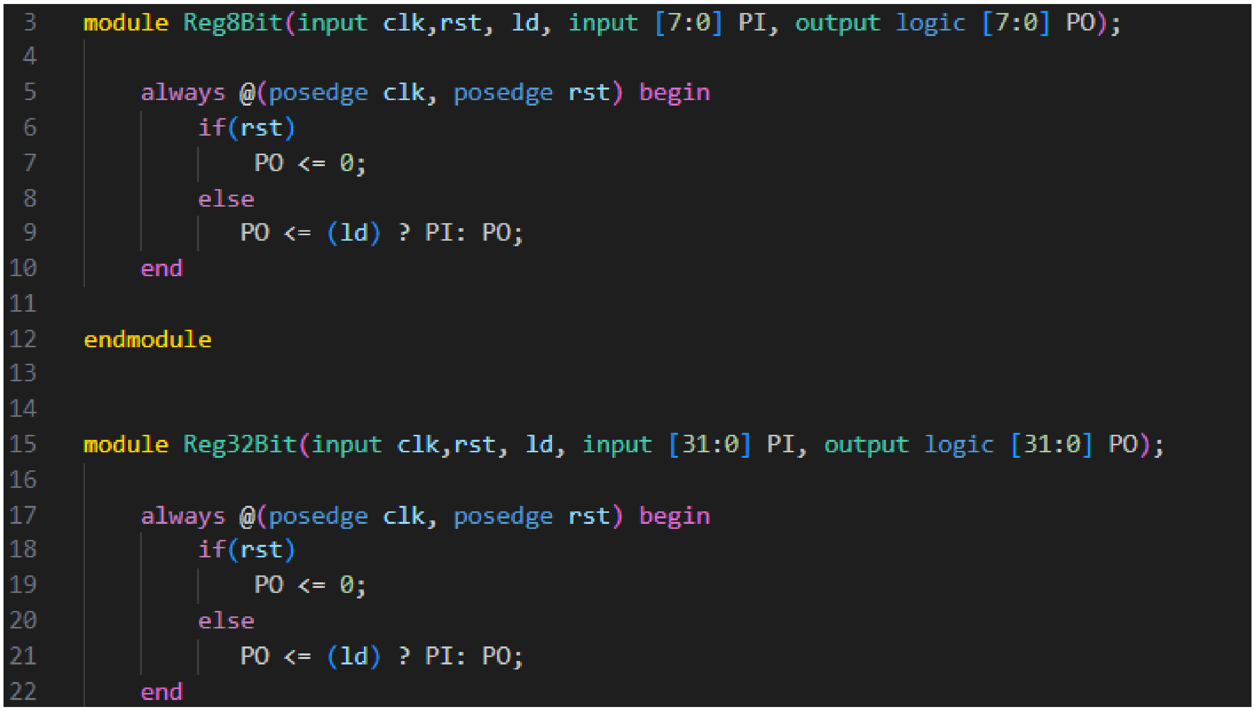
```
module Dcd2to4(input en, input [1:0] A, output [3:0] I);

    assign I = ~en ? 4'b0000 :
            (A == 0) ? 4'b0001:
            (A == 1) ? 4'b0010:
            (A == 2) ? 4'b0100:
            (A == 3) ? 4'b1000:
            4'bx;

endmodule
```

Counter



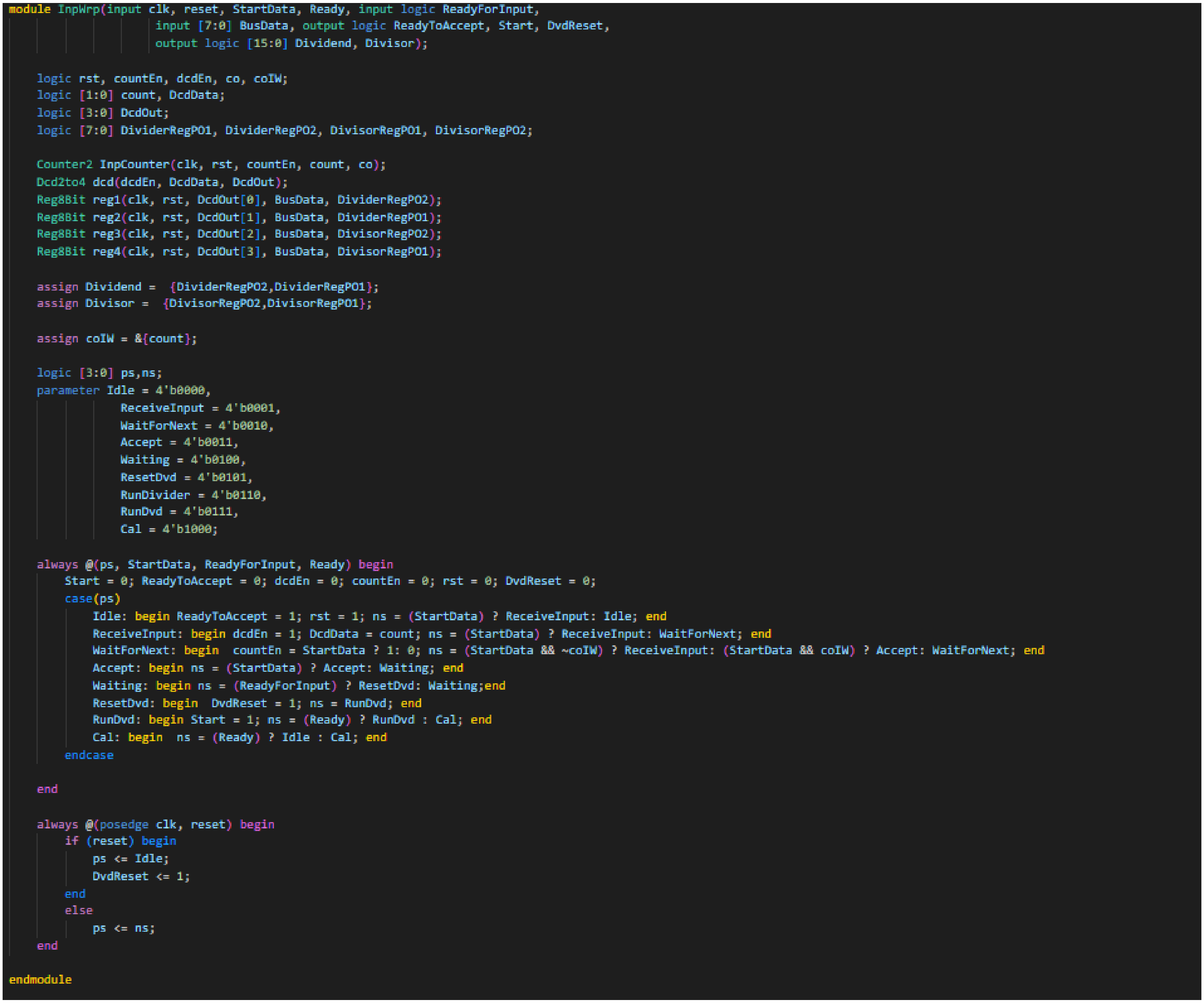
```
module Counter2(input clk,rst,en, output logic [1:0] C, output logic cout);

    always @(posedge clk, posedge rst) begin
        if(rst)
            C <= 0;
        else
            {cout,C} <= (en) ? C + 1 : C;
    end

endmodule
```

Registers



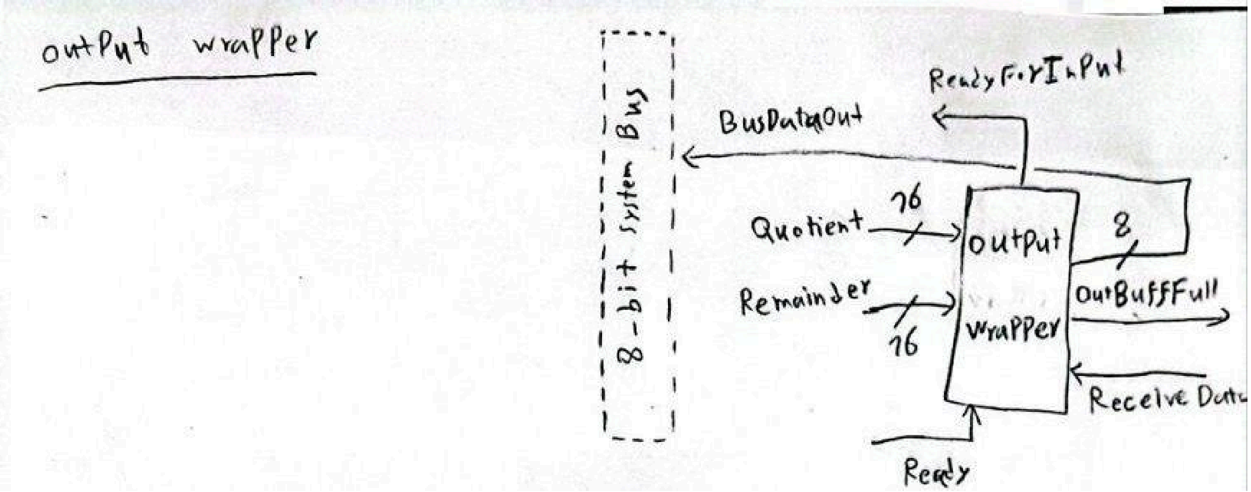
```
module Reg8Bit(input clk,rst, ld, input [7:0] PI, output logic [7:0] PO);

    always @(posedge clk, posedge rst) begin
        if(rst)
            PO <= 0;
        else
           PO <= (ld) ? PI: PO;
    end

endmodule


module Reg32Bit(input clk,rst, ld, input [31:0] PI, output logic [31:0] PO);

    always @(posedge clk, posedge rst) begin
        if(rst)
            PO <= 0;
        else
           PO <= (ld) ? PI: PO;
    end
```

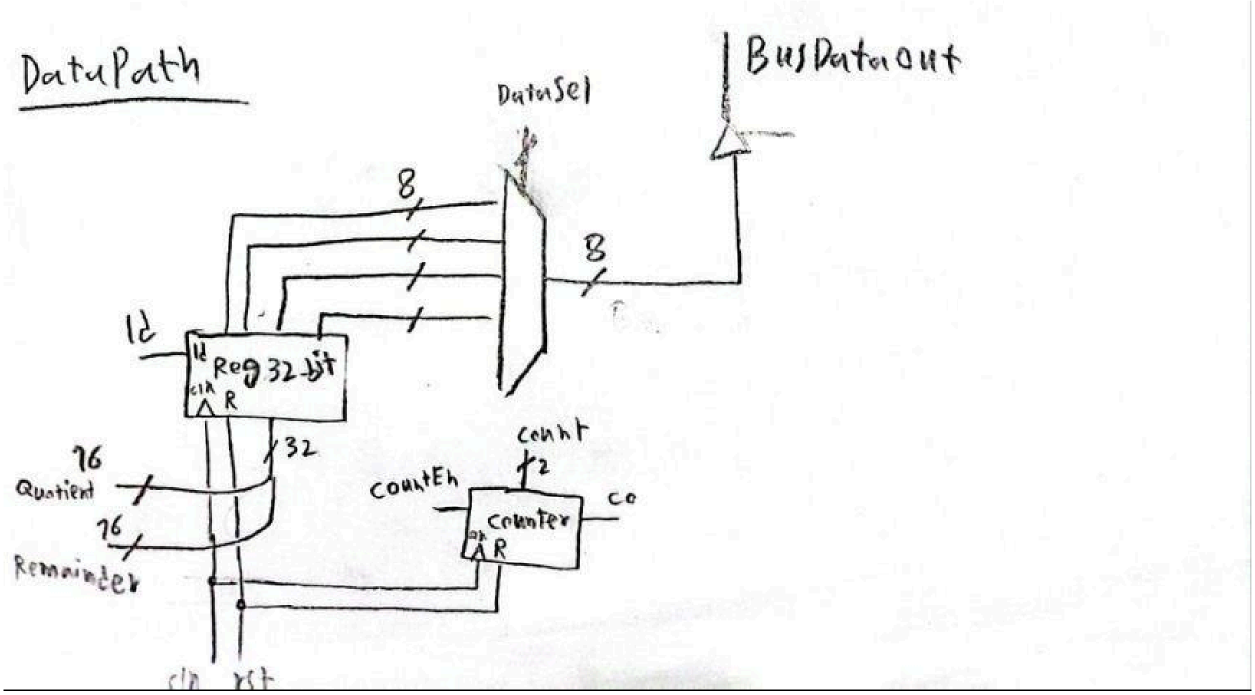
Input Wrapper

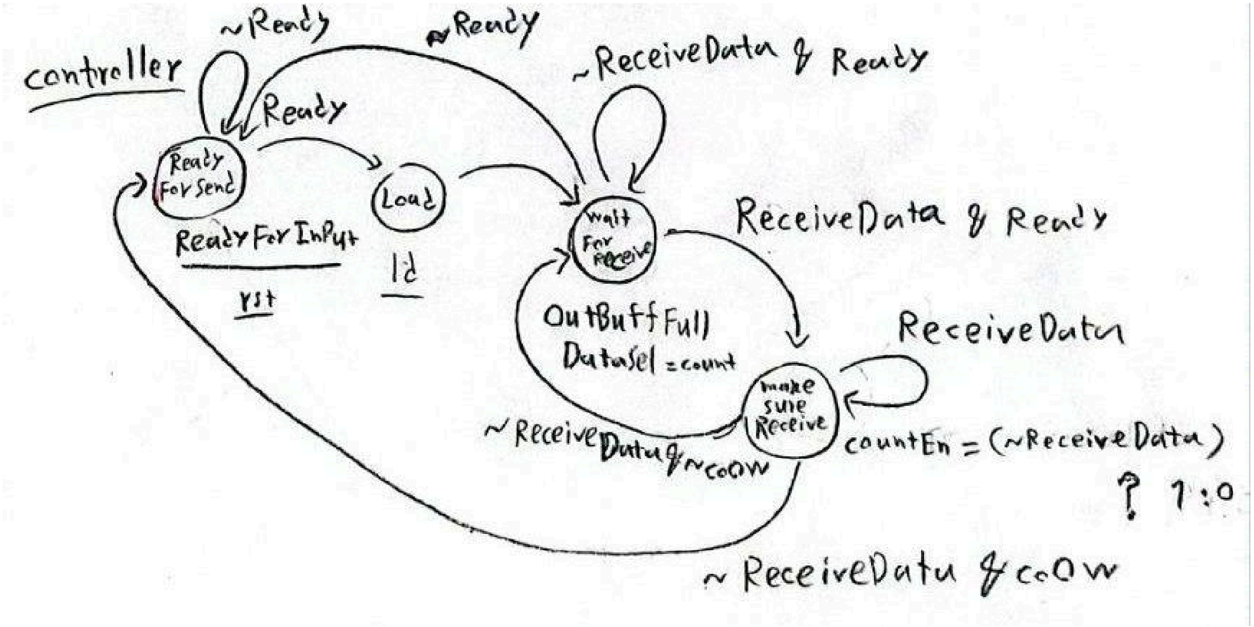


```
module InpWrp(input clk, reset, StartData, Ready, input logic ReadyForInput,
                    input [7:0] BusData, output logic ReadyToAccept, Start, DvdReset,
                    output logic [15:0] Dividend, Divisor);

    logic rst, countEn, dcdEn, co, coIW;
    logic [1:0] count, DcdData;
    logic [3:0] DcdOut;
    logic [7:0] DividerRegPO1, DividerRegPO2, DivisorRegPO1, DivisorRegPO2;

    Counter2 InpCounter(clk, rst, countEn, count, co);
    Dcd2to4 dcd(dcdEn, DcdData, DcdOut);
    Reg8Bit reg1(clk, rst, DcdOut[0], BusData, DividerRegPO2);
    Reg8Bit reg2(clk, rst, DcdOut[1], BusData, DividerRegPO1);
    Reg8Bit reg3(clk, rst, DcdOut[2], BusData, DivisorRegPO2);
    Reg8Bit reg4(clk, rst, DcdOut[3], BusData, DivisorRegPO1);

    assign Dividend =  {DividerRegPO2,DividerRegPO1};
    assign Divisor =  {DivisorRegPO2,DivisorRegPO1};

    assign coIW = &{count};

    logic [3:0] ps,ns;
    parameter Idle = 4'b0000,
                ReceiveInput = 4'b0001,
                WaitForNext = 4'b0010,
                Accept = 4'b0011,
                Waiting = 4'b0100,
                ResetDvd = 4'b0101,
                RunDivider = 4'b0110,
                RunDvd = 4'b0111,
                Cal = 4'b1000;

    always @(ps, StartData, ReadyForInput, Ready) begin
        Start = 0; ReadyToAccept = 0; dcdEn = 0; countEn = 0; rst = 0; DvdReset = 0;
        case(ps)
            Idle: begin ReadyToAccept = 1; rst = 1; ns = (StartData) ? ReceiveInput: Idle; end
            ReceiveInput: begin dcdEn = 1; DcdData = count; ns = (StartData) ? ReceiveInput: WaitForNext; end
            WaitForNext: begin  countEn = StartData ? 1: 0; ns = (StartData && ~coIW) ? ReceiveInput: (StartData && coIW) ? Accept: WaitForNext; end
            Accept: begin ns = (StartData) ? Accept: Waiting; end
            Waiting: begin ns = (ReadyForInput) ? ResetDvd: Waiting;end
            ResetDvd: begin  DvdReset = 1; ns = RunDvd; end
            RunDvd: begin Start = 1; ns = (Ready) ? RunDvd : Cal; end
            Cal: begin  ns = (Ready) ? Idle : Cal; end
        endcase

    end

    always @(posedge clk, reset) begin
        if (reset) begin
            ps <= Idle;
            DvdReset <= 1;
        end
        else
            ps <= ns;
    end

endmodule
```

Output Wrapper

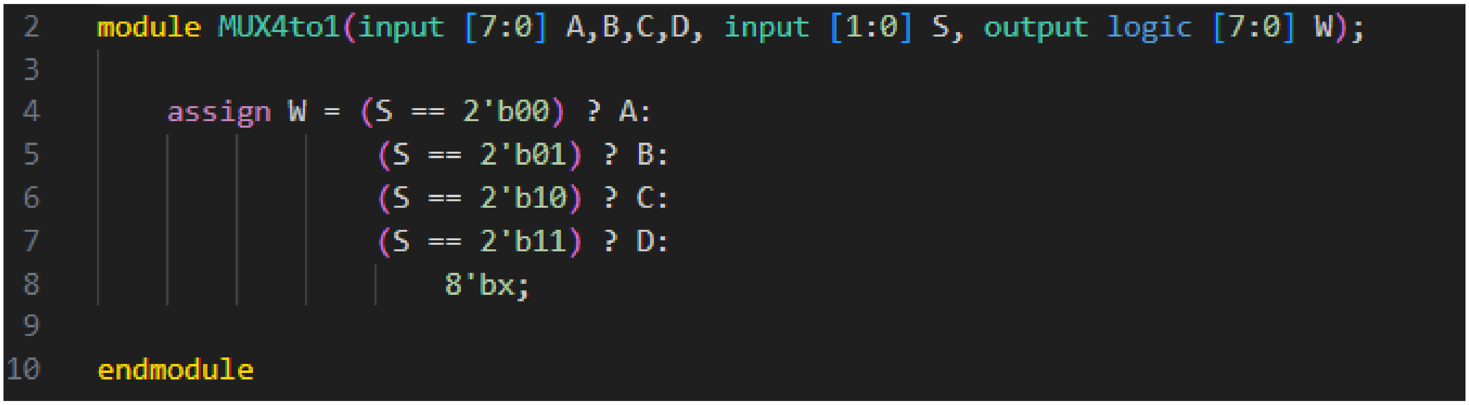






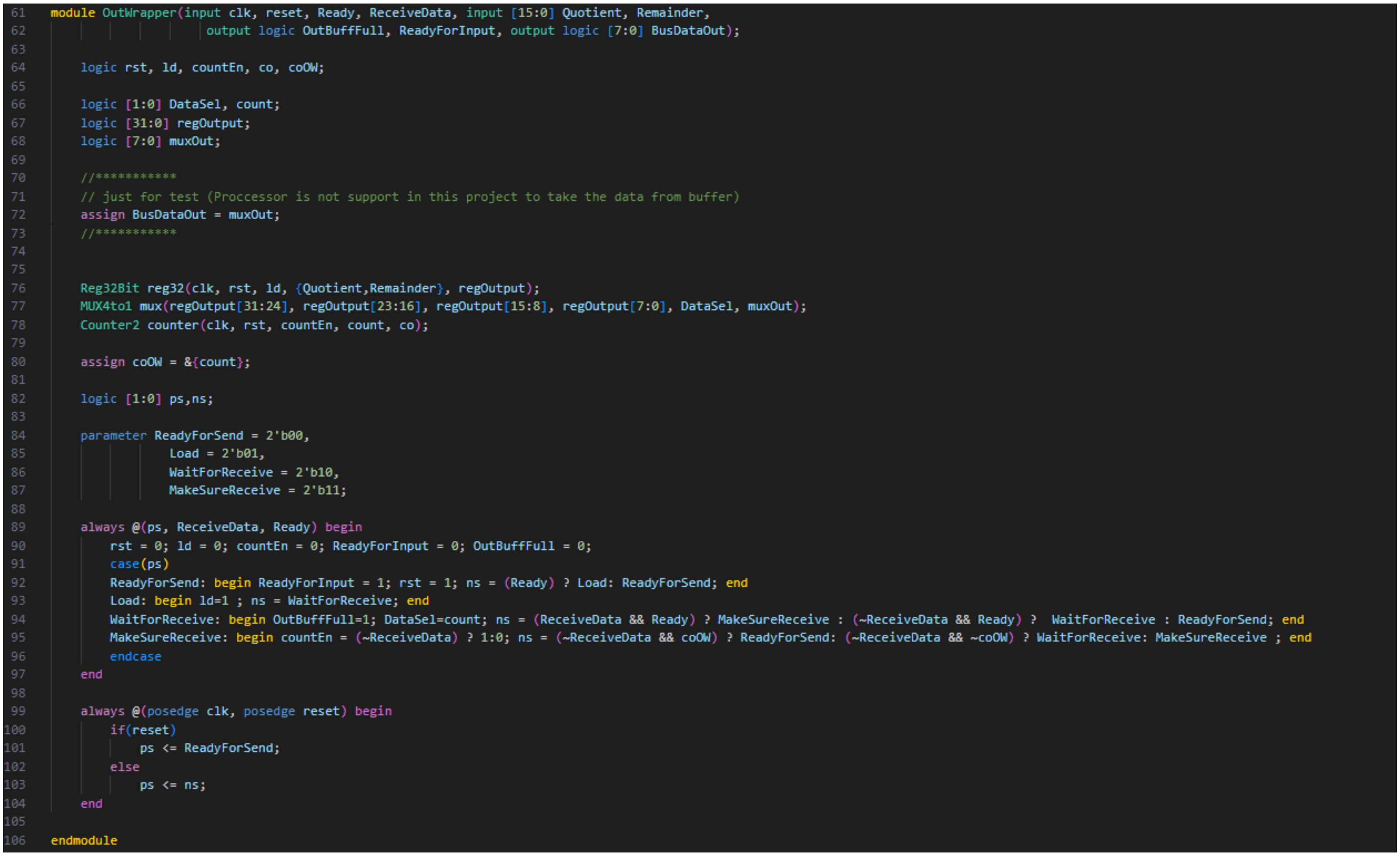
System Verilog

Multiplexer



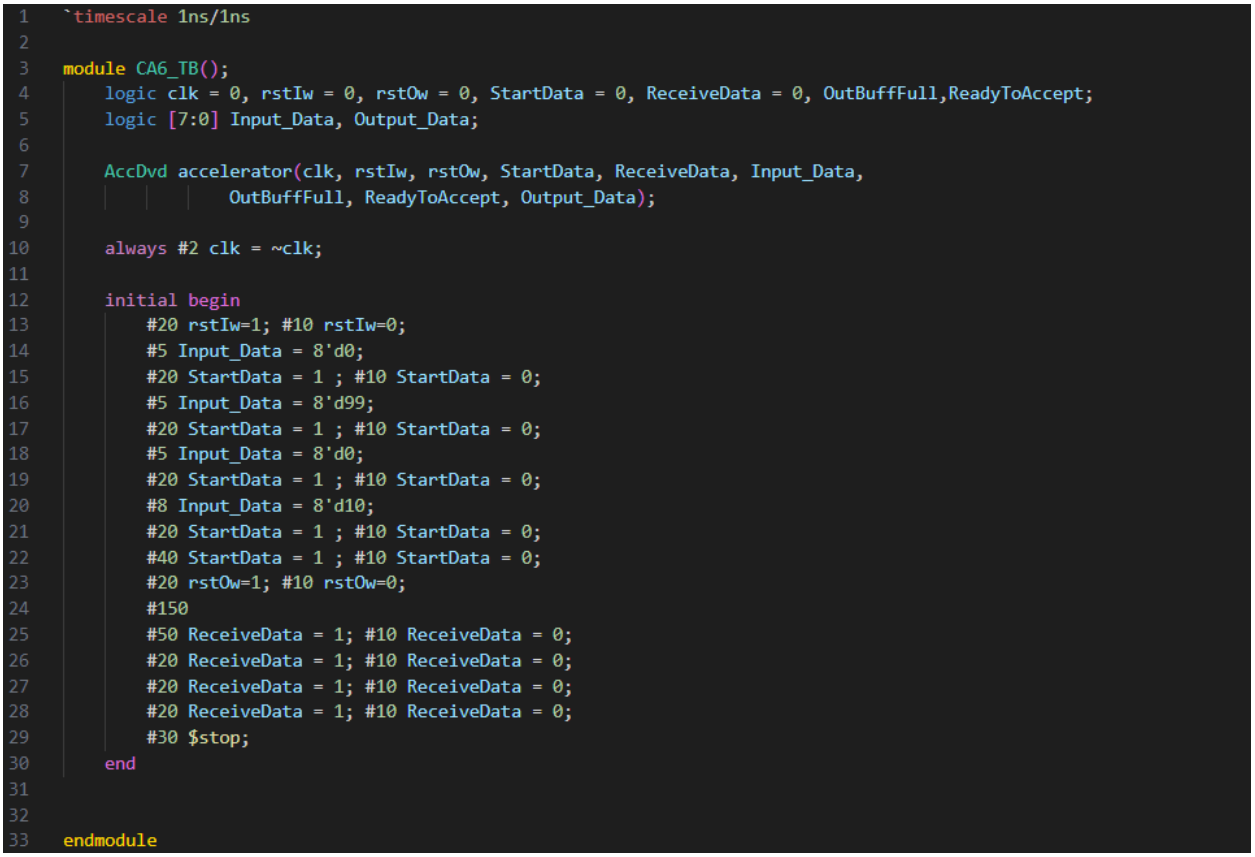
```
module MUX4to1(input [7:0] A,B,C,D, input [1:0] S, output logic [7:0] W);

    assign W = (S == 2'b00) ? A:
                 (S == 2'b01) ? B:
                 (S == 2'b10) ? C:
                 (S == 2'b11) ? D:
                     8'bx;

endmodule
```

Output Wrapper

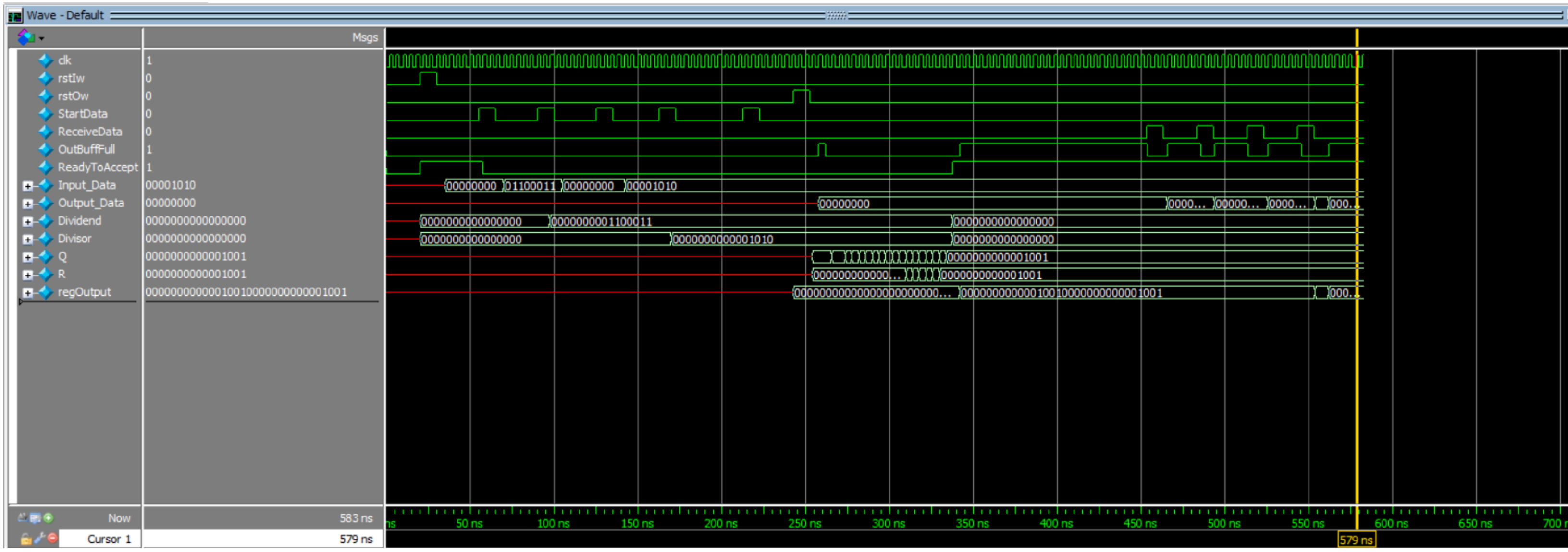


```
module OutWrapper(input clk, reset, Ready, ReceiveData, input [15:0] Quotient, Remainder,
                        output logic OutBuffFull, ReadyForInput, output logic [7:0] BusDataOut);

    logic rst, ld, countEn, co, coOW;

    logic [1:0] DataSel, count;
    logic [31:0] regOutput;
    logic [7:0] muxOut;

    //***********
    // just for test (Proccessor is not support in this project to take the data from buffer)
    assign BusDataOut = muxOut;
    //***********


    Reg32Bit reg32(clk, rst, ld, {Quotient,Remainder}, regOutput);
    MUX4to1 mux(regOutput[31:24], regOutput[23:16], regOutput[15:8], regOutput[7:0], DataSel, muxOut);
    Counter2 counter(clk, rst, countEn, count, co);

    assign coOW = &{count};

    logic [1:0] ps,ns;

    parameter ReadyForSend = 2'b00,
                Load = 2'b01,
                WaitForReceive = 2'b10,
                MakeSureReceive = 2'b11;

    always @(ps, ReceiveData, Ready) begin
        rst = 0; ld = 0; countEn = 0; ReadyForInput = 0; OutBuffFull = 0;
        case(ps)
        ReadyForSend: begin ReadyForInput = 1; rst = 1; ns = (Ready) ? Load: ReadyForSend; end
        Load: begin ld=1 ; ns = WaitForReceive; end
        WaitForReceive: begin OutBuffFull=1; DataSel=count; ns = (ReceiveData && Ready) ? MakeSureReceive : (~ReceiveData && Ready) ?  WaitForReceive : ReadyForSend; end
        MakeSureReceive: begin countEn = (~ReceiveData) ? 1:0; ns = (~ReceiveData && coOW) ? ReadyForSend: (~ReceiveData && ~coOW) ? WaitForReceive: MakeSureReceive ; end
        endcase
    end

    always @(posedge clk, posedge reset) begin
        if(reset)
            ps <= ReadyForSend;
        else
            ps <= ns;
    end

endmodule
```

Pre Synthesis



```
`timescale 1ns/1ns

module CA6_TB();
    logic clk = 0, rstIw = 0, rstOw = 0, StartData = 0, ReceiveData = 0, OutBuffFull,ReadyToAccept;
    logic [7:0] Input_Data, Output_Data;

    AccDvd accelerator(clk, rstIw, rstOw, StartData, ReceiveData, Input_Data,
                OutBuffFull, ReadyToAccept, Output_Data);

    always #2 clk = ~clk;

    initial begin
        #20 rstIw=1; #10 rstIw=0;
        #5 Input_Data = 8'd0;
        #20 StartData = 1 ; #10 StartData = 0;
        #5 Input_Data = 8'd99;
        #20 StartData = 1 ; #10 StartData = 0;
        #5 Input_Data = 8'd0;
        #20 StartData = 1 ; #10 StartData = 0;
        #8 Input_Data = 8'd10;
        #20 StartData = 1 ; #10 StartData = 0;
        #40 StartData = 1 ; #10 StartData = 0;
        #20 rstOw=1; #10 rstOw=0;
        #150
        #50 ReceiveData = 1; #10 ReceiveData = 0;
        #20 ReceiveData = 1; #10 ReceiveData = 0;
        #20 ReceiveData = 1; #10 ReceiveData = 0;
        #20 ReceiveData = 1; #10 ReceiveData = 0;
        #30 $stop;
    end


endmodule
```

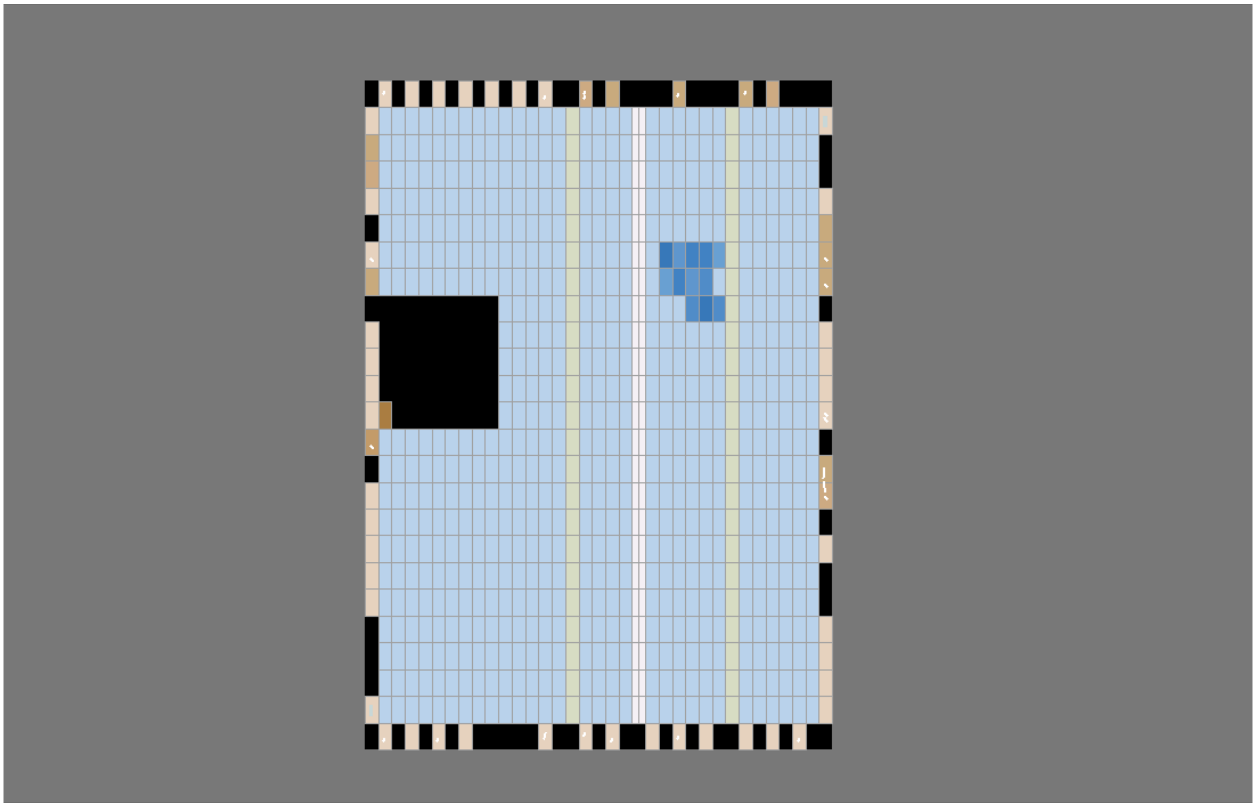


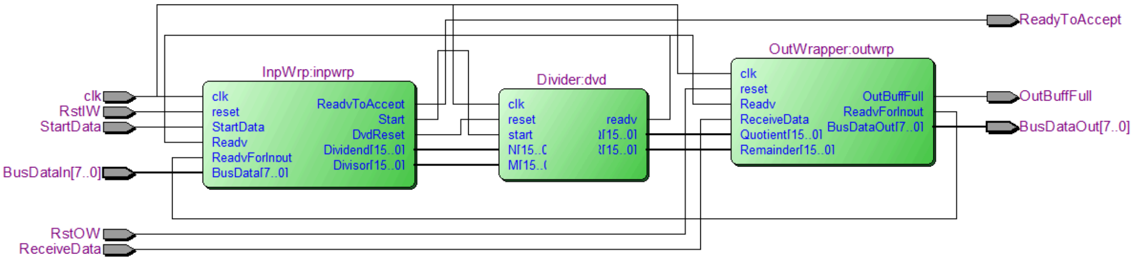
Wave - Default

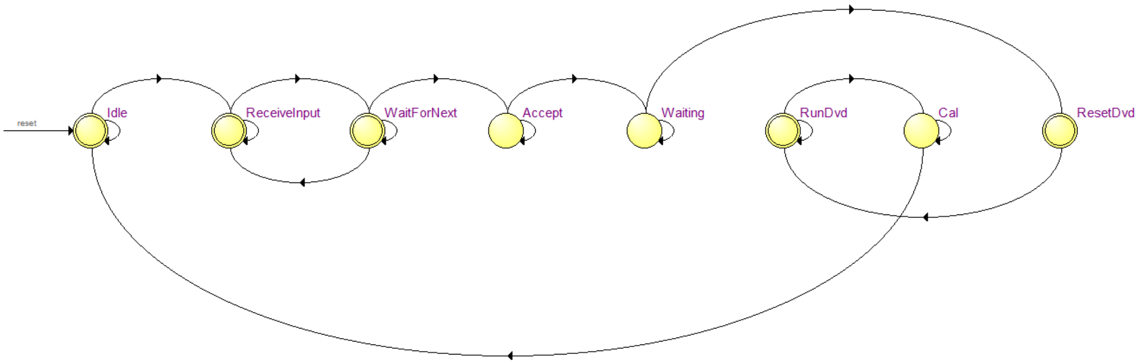
	Msgs
clk	1
rstIw	0
rstOw	0
StartData	0
ReceiveData	0
OutBuffFull	1
ReadyToAccept	1
Input_Data	00001010
Output_Data	00000000
Dividend	0000000000000000
Divisor	0000000000000000
Q	0000000000001001
R	0000000000001001
regOutput	00000000000010010000000000001001

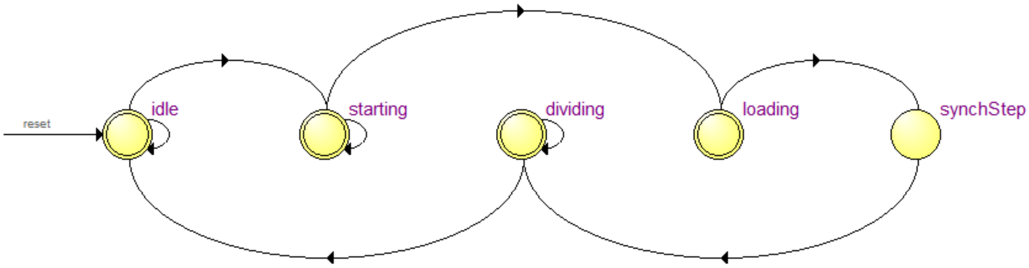
Now	583 ns
Cursor 1	579 ns

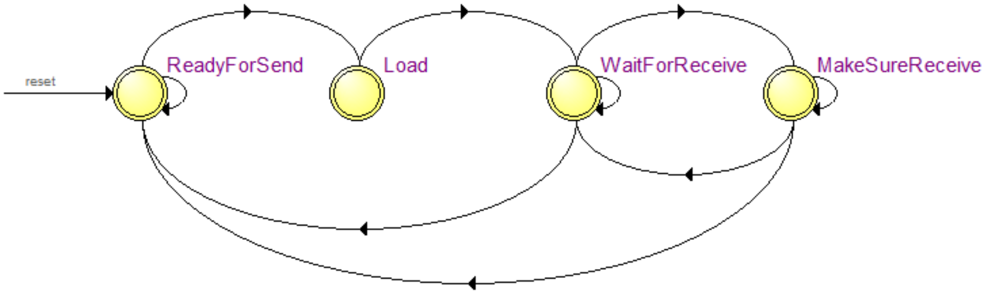
Post Synthesis (Cyclone IV E)

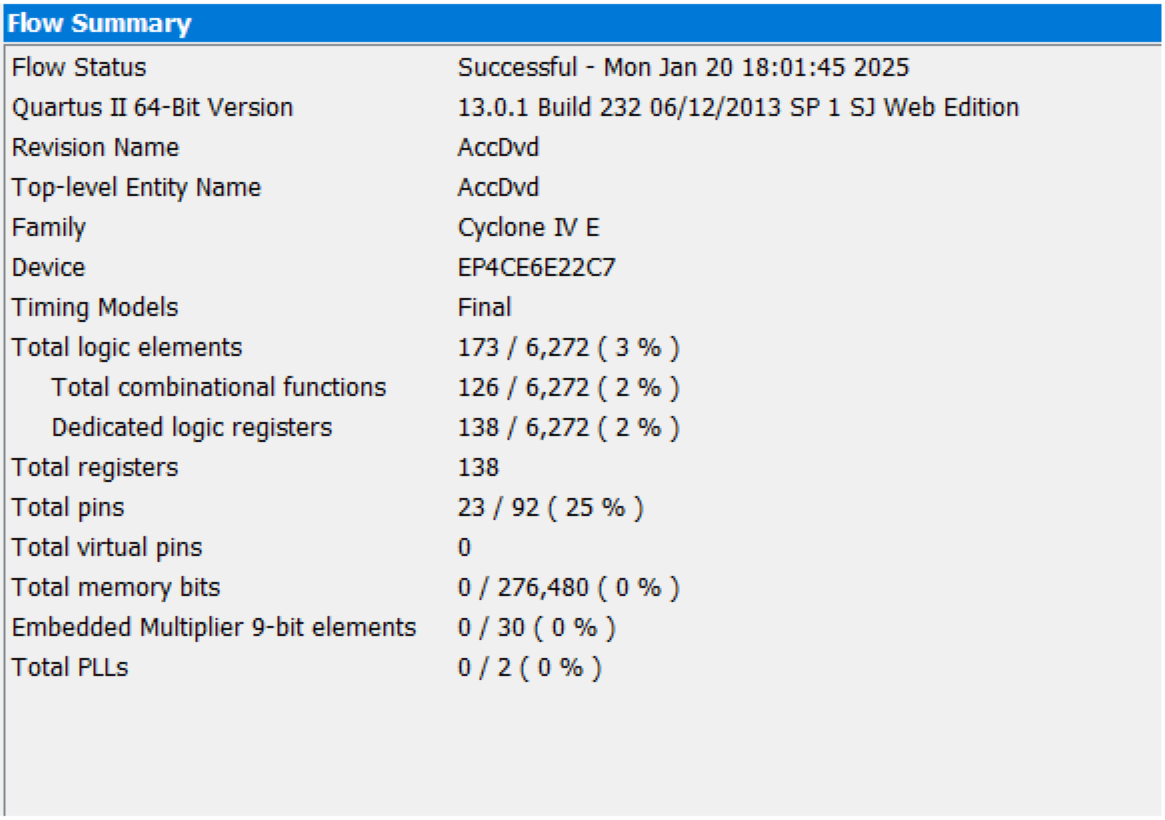





reset
Idle
ReceiveInput
WaitForNext
Accept
Waiting
RunDvd
Cal
ResetDvd

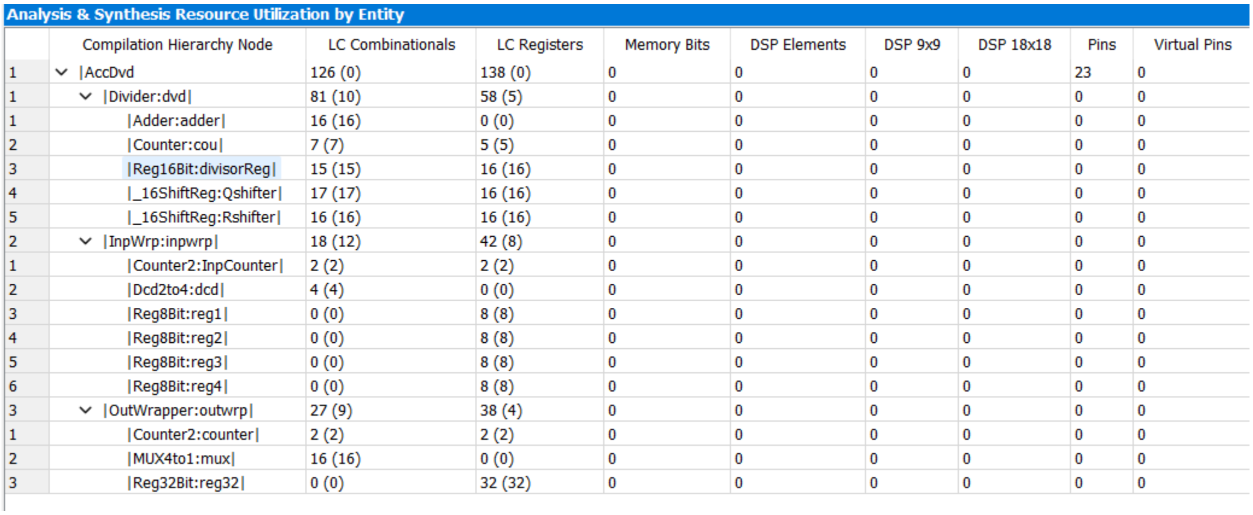

reset
idle
starting
dividing
loading
synchStep


reset
ReadyForSend
Load
WaitForReceive
MakeSureReceive



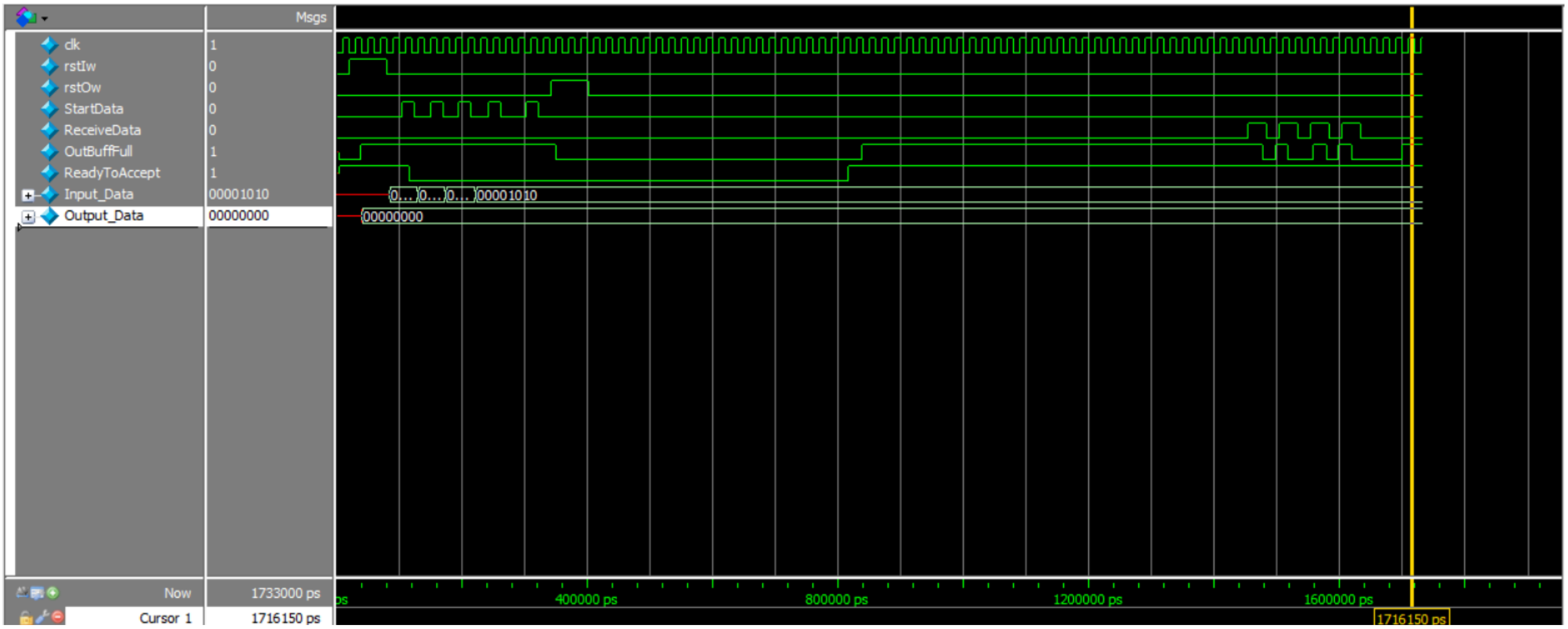
Flow Summary

Flow Status	Successful - Mon Jan 20 18:01:45 2025
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	AccDvd
Top-level Entity Name	AccDvd
Family	Cyclone IV E
Device	EP4CE6E22C7
Timing Models	Final
Total logic elements	173 / 6,272 (3 %)
Total combinational functions	126 / 6,272 (2 %)
Dedicated logic registers	138 / 6,272 (2 %)
Total registers	138
Total pins	23 / 92 (25 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)



Analysis & Synthesis Resource Utilization by Entity

	Compilation Hierarchy Node	LC Combinationals	LC Registers	Memory Bits	DSP Elements	DSP 9x9	DSP 18x18	Pins	Virtual Pins
1	|AccDvd	126 (0)	138 (0)	0	0	0	0	23	0
1	|Divider:dvd|	81 (10)	58 (5)	0	0	0	0	0	0
1	|Adder:adder|	16 (16)	0 (0)	0	0	0	0	0	0
2	|Counter:cou|	7 (7)	5 (5)	0	0	0	0	0	0
3	|Reg16Bit:divisorReg|	15 (15)	16 (16)	0	0	0	0	0	0
4	|_16ShiftReg:Qshifter|	17 (17)	16 (16)	0	0	0	0	0	0
5	|_16ShiftReg:Rshifter|	16 (16)	16 (16)	0	0	0	0	0	0
2	|InpWrp:inpwrp|	18 (12)	42 (8)	0	0	0	0	0	0
1	|Counter2:InpCounter|	2 (2)	2 (2)	0	0	0	0	0	0
2	|Dcd2to4:dcd|	4 (4)	0 (0)	0	0	0	0	0	0
3	|Reg8Bit:reg1|	0 (0)	8 (8)	0	0	0	0	0	0
4	|Reg8Bit:reg2|	0 (0)	8 (8)	0	0	0	0	0	0
5	|Reg8Bit:reg3|	0 (0)	8 (8)	0	0	0	0	0	0
6	|Reg8Bit:reg4|	0 (0)	8 (8)	0	0	0	0	0	0
3	|OutWrapper:outwrp|	27 (9)	38 (4)	0	0	0	0	0	0
1	|Counter2:counter|	2 (2)	2 (2)	0	0	0	0	0	0
2	|MUX4to1:mux|	16 (16)	0 (0)	0	0	0	0	0	0
3	|Reg32Bit:reg32|	0 (0)	32 (32)	0	0	0	0	0	0



```
                    Msgs
clk                 1
rstIw               0
rstOw               0
StartData           0
ReceiveData         0
OutBuffFull         1
ReadyToAccept       1
Input_Data          00001010
Output_Data         00000000

Now                 1733000 ps
Cursor 1            1716150 ps
```