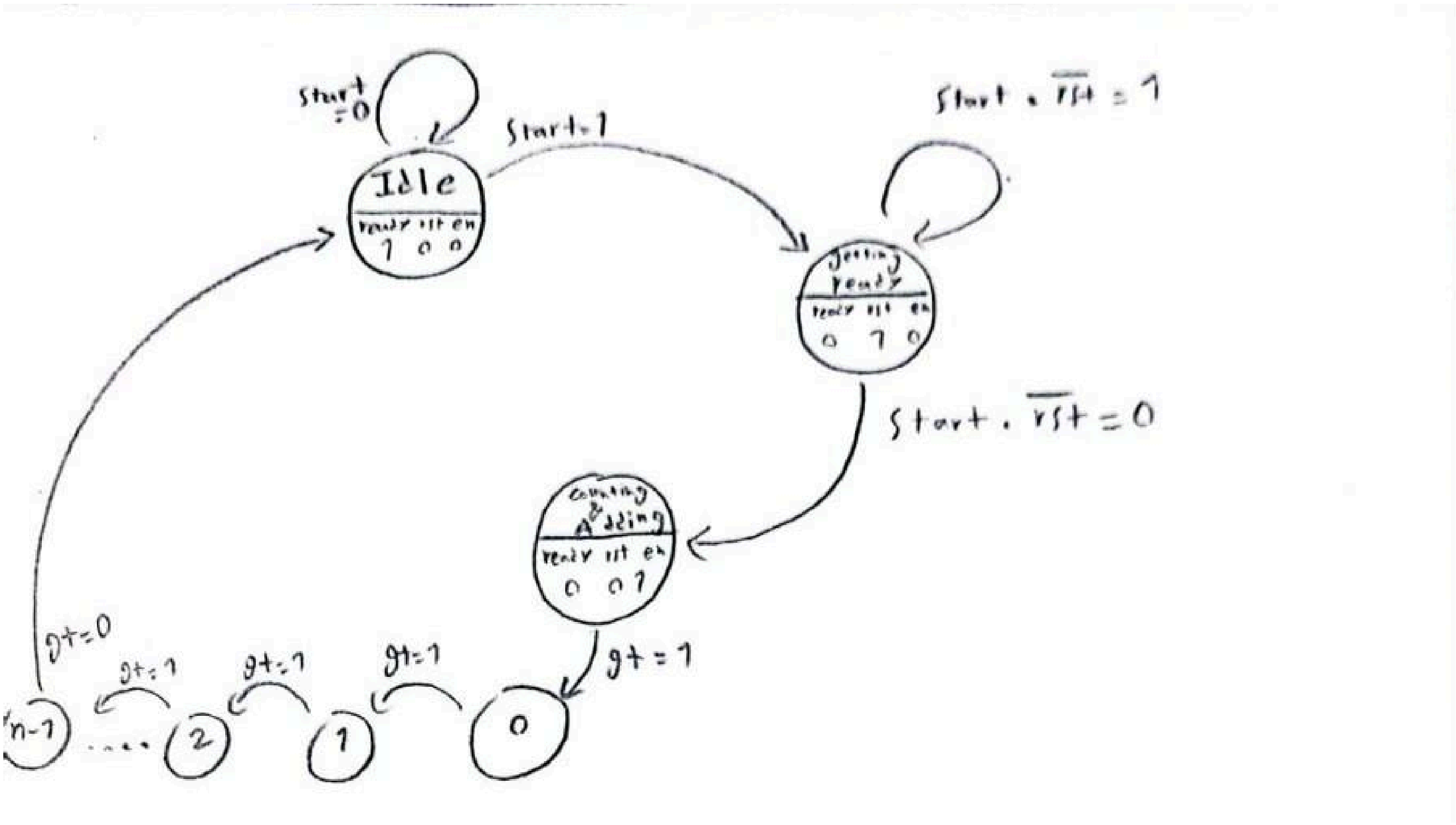
**Erfan Falahati 810102491**

**CA4**

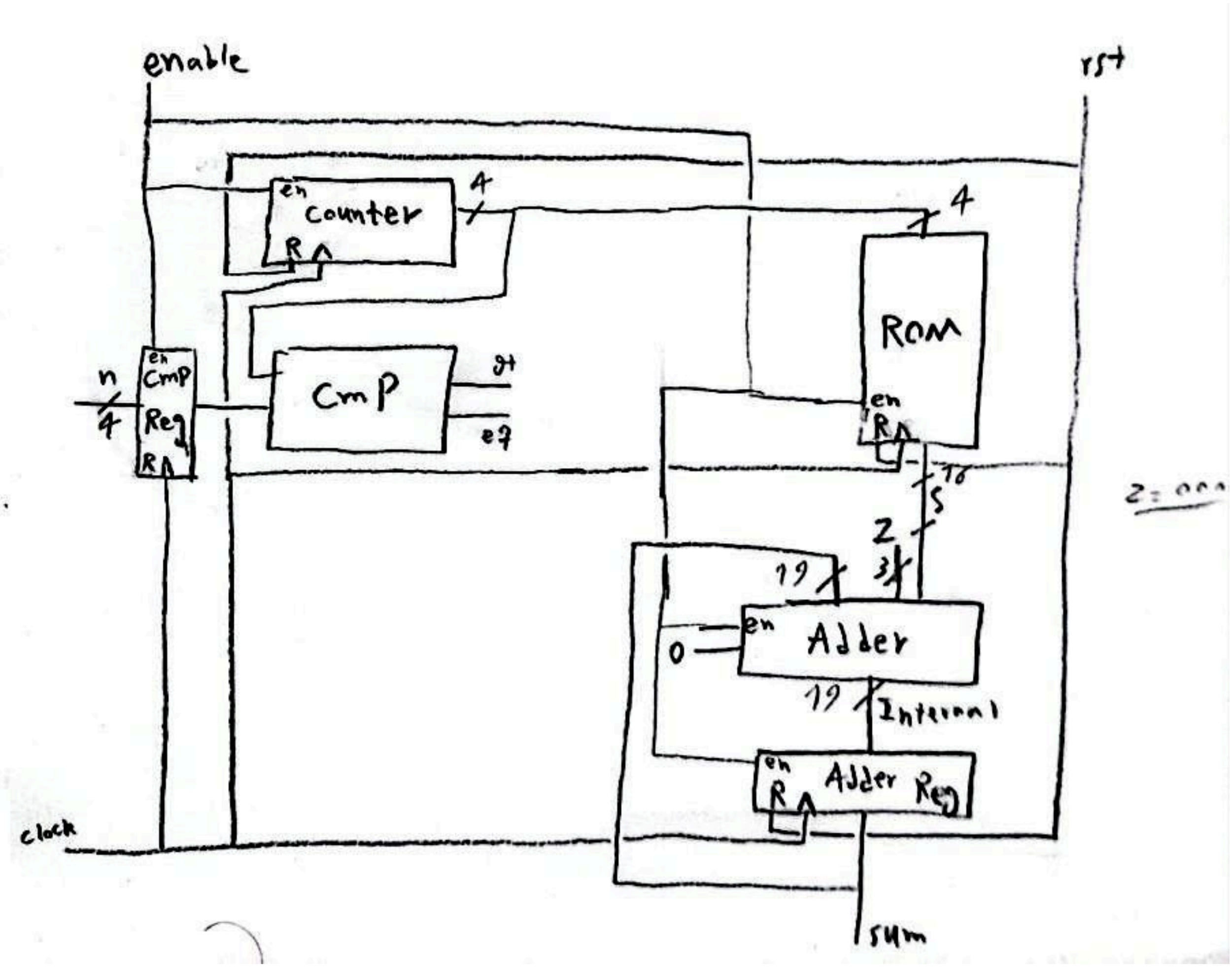
**ECE-Fall403**

**Course id: 4031810189401**

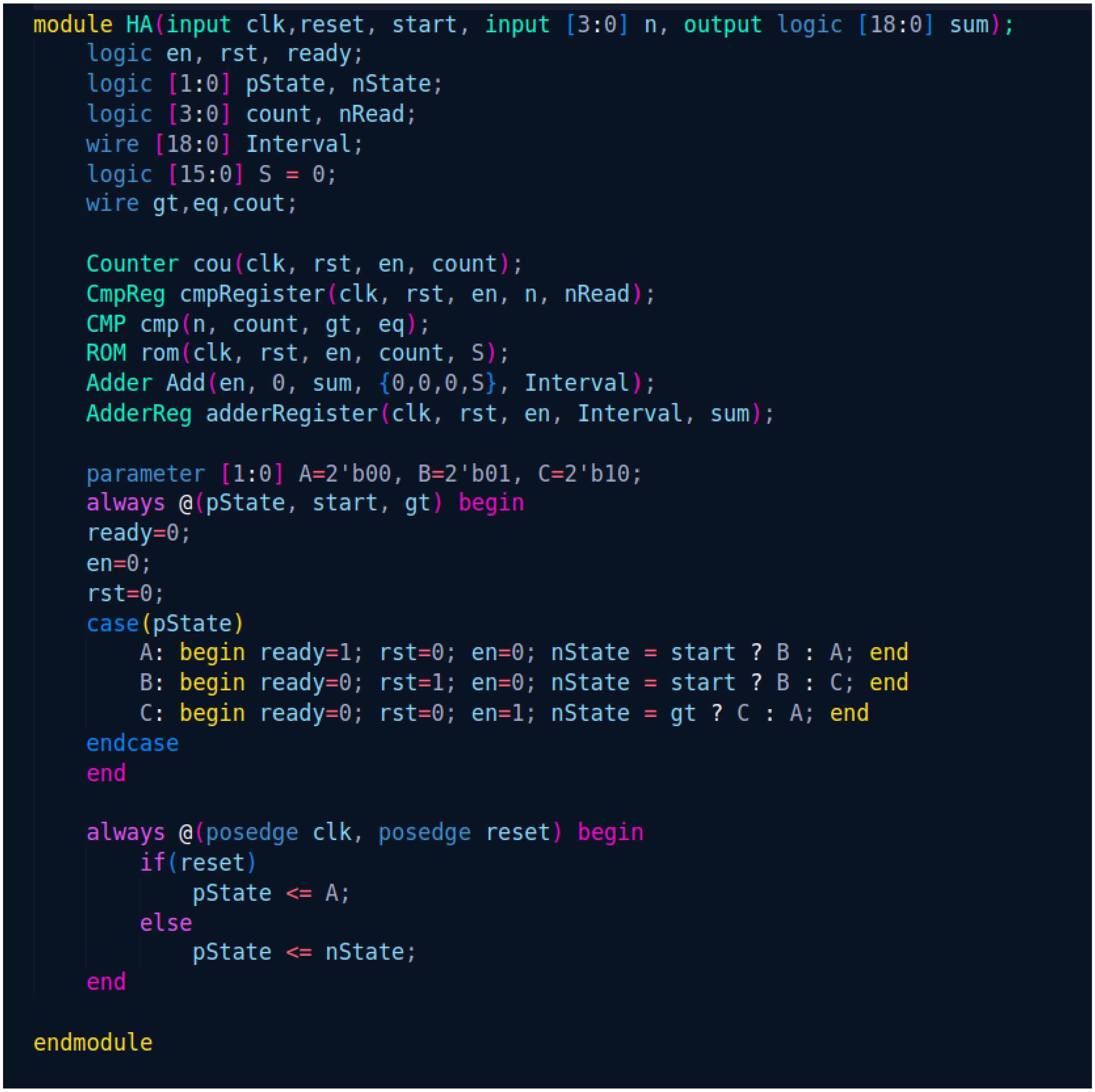
## State Machine:



# RTL Level (Design)

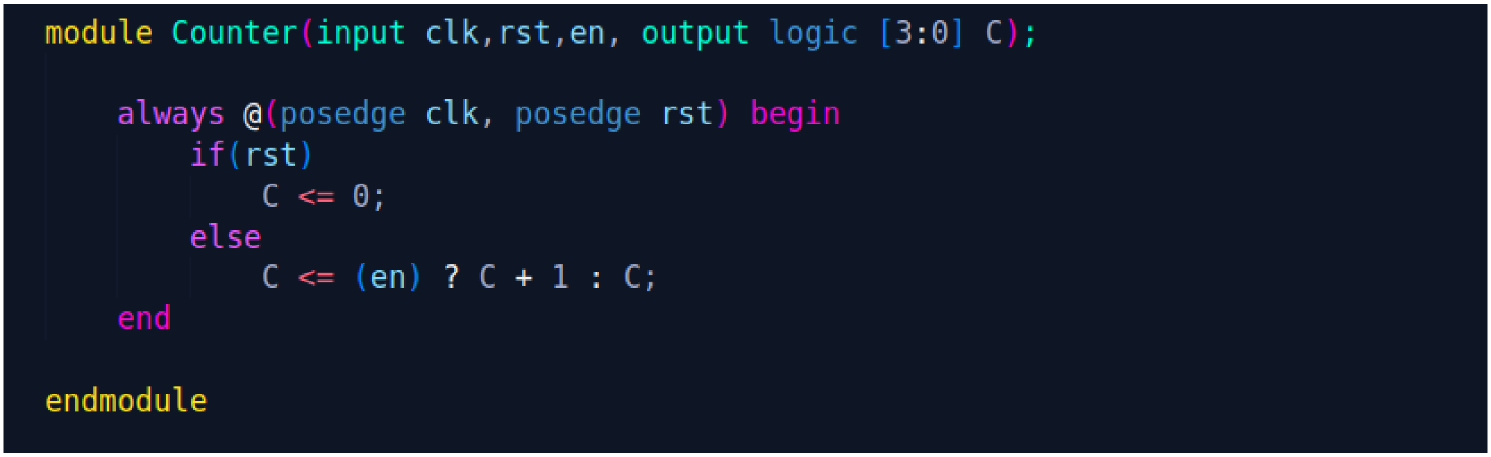


# System Verilog



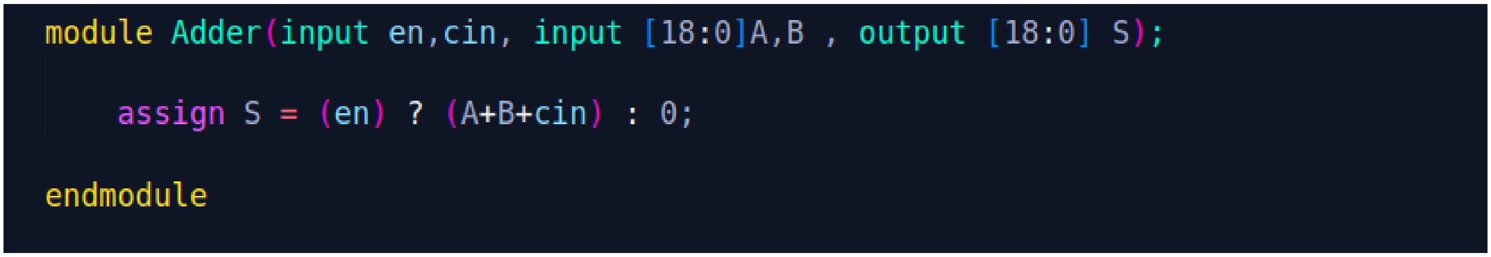
```
module HA(input clk,reset, start, input [3:0] n, output logic [18:0] sum);
    logic en, rst, ready;
    logic [1:0] pState, nState;
    logic [3:0] count, nRead;
    wire [18:0] Interval;
    logic [15:0] S = 0;
    wire gt,eq,cout;

    Counter cou(clk, rst, en, count);
    CmpReg cmpRegister(clk, rst, en, n, nRead);
    CMP cmp(n, count, gt, eq);
    ROM rom(clk, rst, en, count, S);
    Adder Add(en, 0, sum, {0,0,0,S}, Interval);
    AdderReg adderRegister(clk, rst, en, Interval, sum);

    parameter [1:0] A=2'b00, B=2'b01, C=2'b10;
    always @(pState, start, gt) begin
    ready=0;
    en=0;
    rst=0;
    case(pState)
        A: begin ready=1; rst=0; en=0; nState = start ? B : A; end
        B: begin ready=0; rst=1; en=0; nState = start ? B : C; end
        C: begin ready=0; rst=0; en=1; nState = gt ? C : A; end
    endcase
    end

    always @(posedge clk, posedge reset) begin
        if(reset)
            pState <= A;
        else
            pState <= nState;
    end

endmodule
```

# Counter:



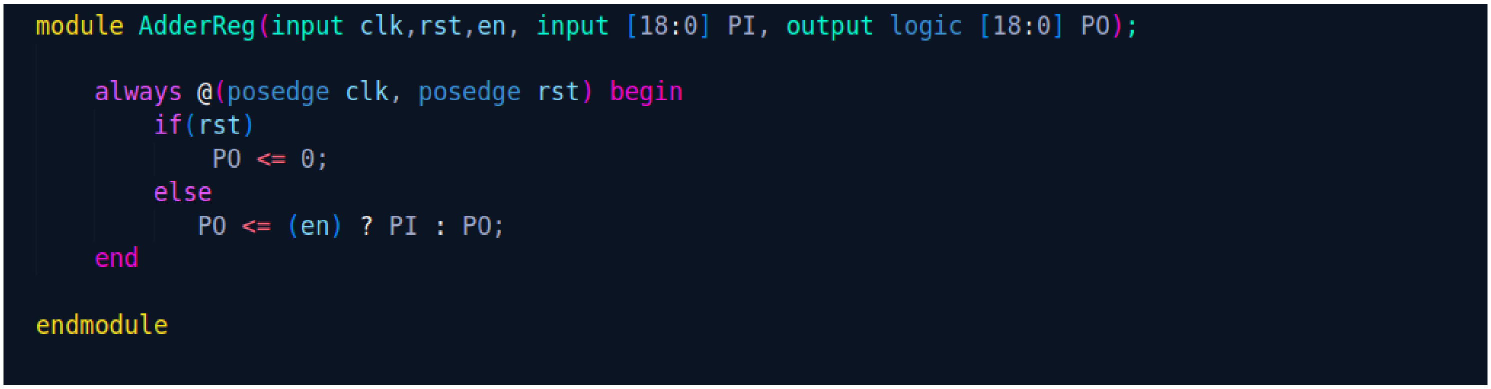
```
module Counter(input clk,rst,en, output logic [3:0] C);

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

endmodule
```

# Adder:



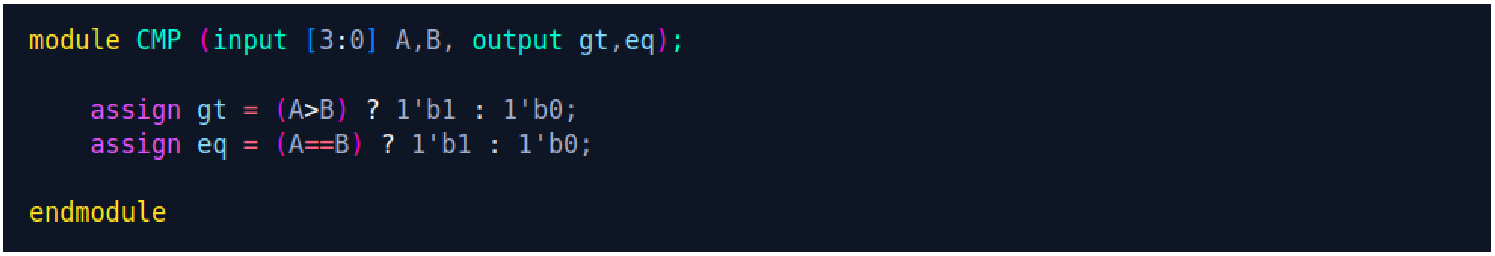
```
module Adder(input en,cin, input [18:0]A,B , output [18:0] S);

    assign S = (en) ? (A+B+cin) : 0;

endmodule
```

# Adder Register:



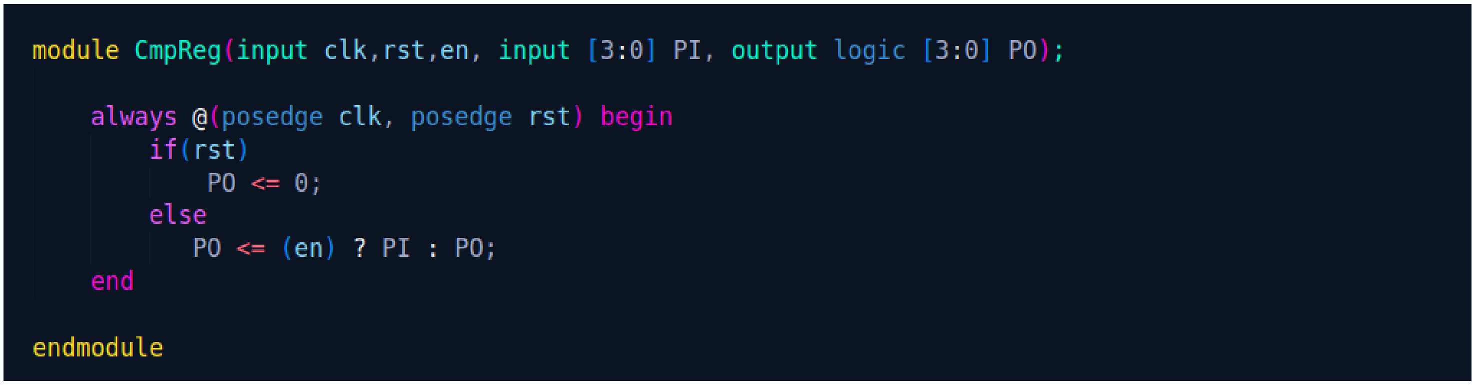
```
module AdderReg(input clk,rst,en, input [18:0] PI, output logic [18:0] PO);

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

endmodule
```

# Comparator



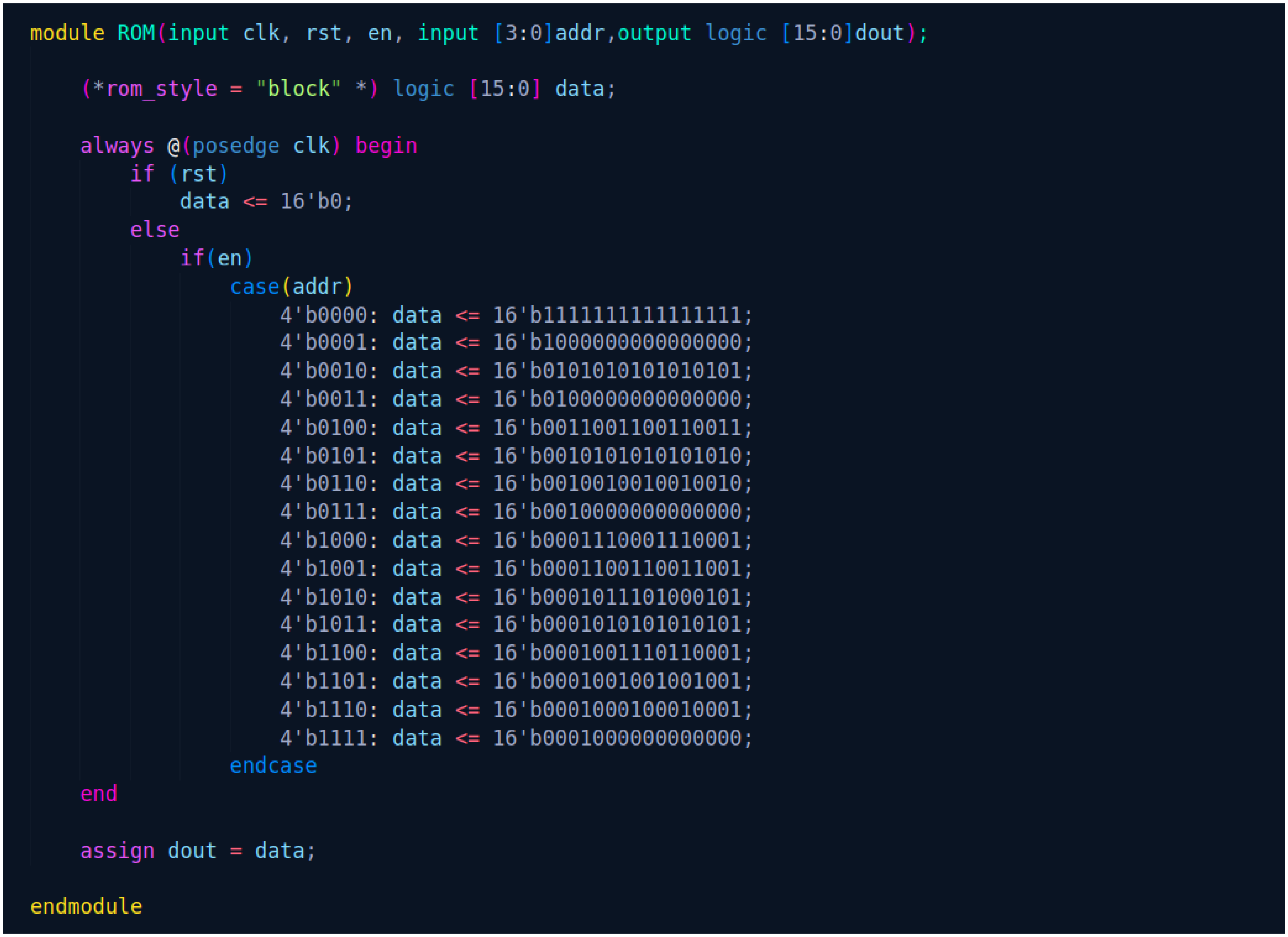
```
module CMP (input [3:0] A,B, output gt,eq);

    assign gt = (A>B) ? 1'b1 : 1'b0;
    assign eq = (A==B) ? 1'b1 : 1'b0;

endmodule
```

# Comparator Register



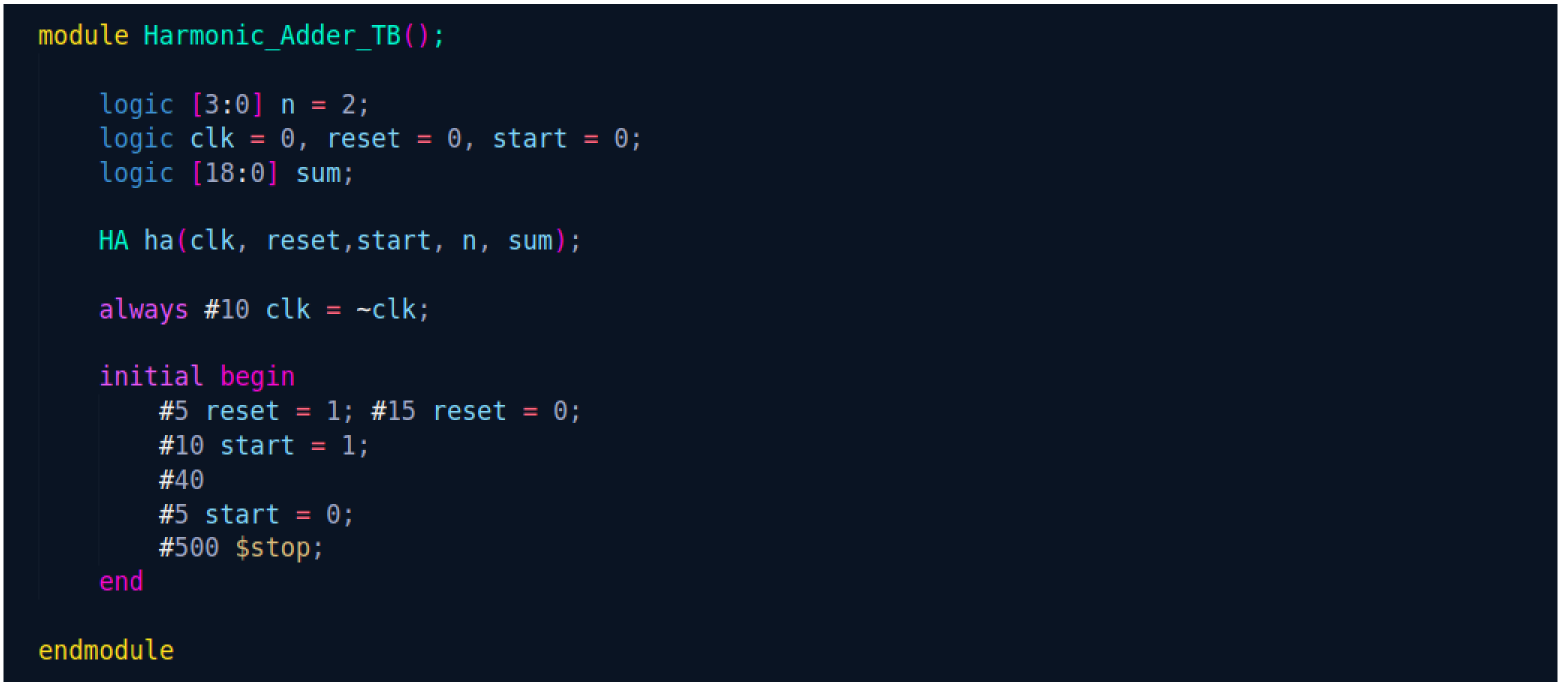
```
module CmpReg(input clk,rst,en, input [3:0] PI, output logic [3:0] PO);

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

endmodule
```

# ROM

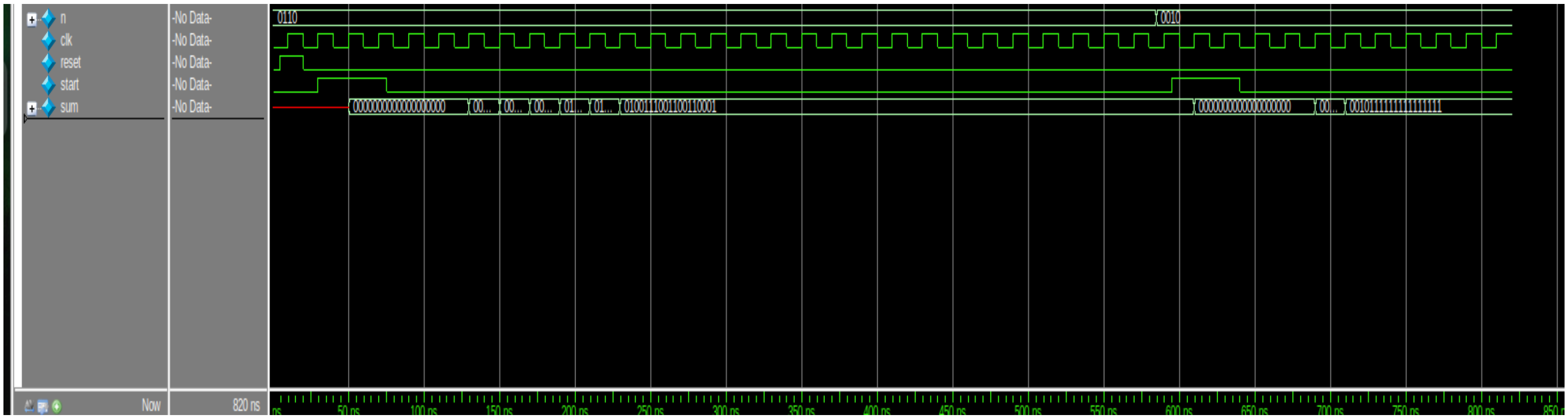


```
module ROM(input clk, rst, en, input [3:0]addr,output logic [15:0]dout);

    (*rom_style = "block" *) logic [15:0] data;

    always @(posedge clk) begin
        if (rst)
            data <= 16'b0;
        else
            if(en)
                case(addr)
                    4'b0000: data <= 16'b1111111111111111;
                    4'b0001: data <= 16'b1000000000000000;
                    4'b0010: data <= 16'b0101010101010101;
                    4'b0011: data <= 16'b0100000000000000;
                    4'b0100: data <= 16'b0011001100110011;
                    4'b0101: data <= 16'b0010101010101010;
                    4'b0110: data <= 16'b0010010010010010;
                    4'b0111: data <= 16'b0010000000000000;
                    4'b1000: data <= 16'b0001110001110001;
                    4'b1001: data <= 16'b0001100110011001;
                    4'b1010: data <= 16'b0001011101000101;
                    4'b1011: data <= 16'b0001010101010101;
                    4'b1100: data <= 16'b0001001110110001;
                    4'b1101: data <= 16'b0001001001001001;
                    4'b1110: data <= 16'b0001000100010001;
                    4'b1111: data <= 16'b0001000000000000;
                endcase
    end

    assign dout = data;

endmodule
```

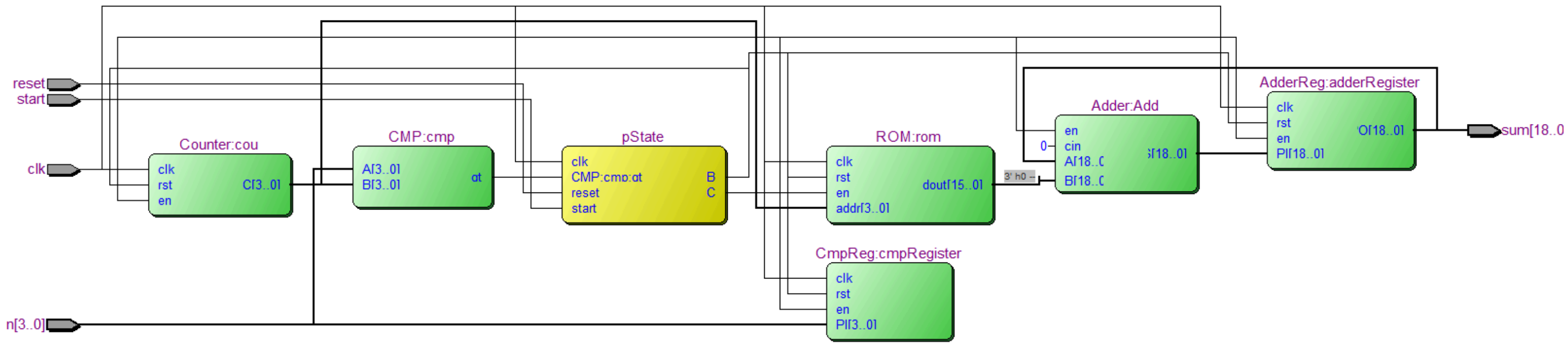
# Test Bench

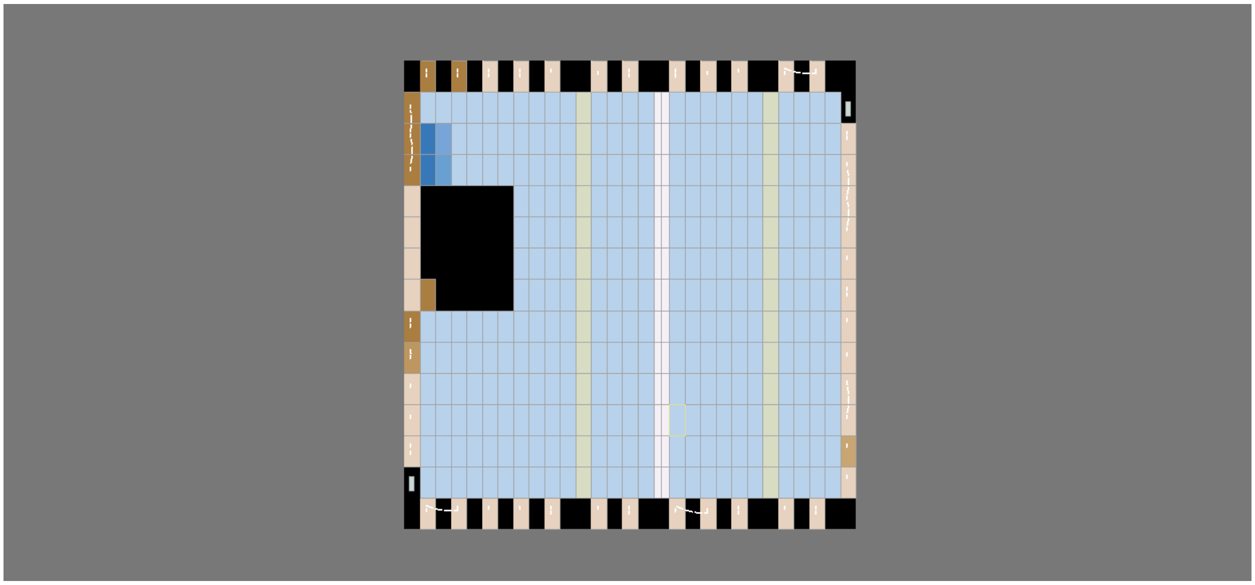


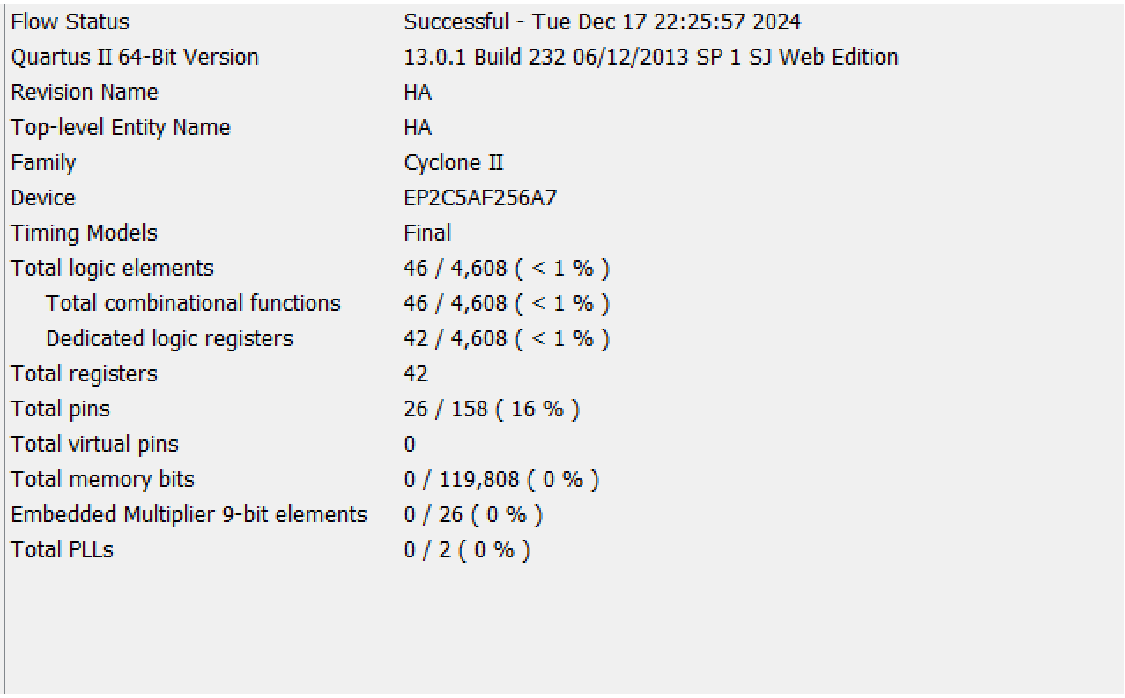
```
module Harmonic_Adder_TB();

    logic [3:0] n = 2;
    logic clk = 0, reset = 0, start = 0;
    logic [18:0] sum;

    HA ha(clk, reset,start, n, sum);

    always #10 clk = ~clk;

    initial begin
        #5 reset = 1; #15 reset = 0;
        #10 start = 1;
        #40
        #5 start = 0;
        #500 $stop;
    end

endmodule
```



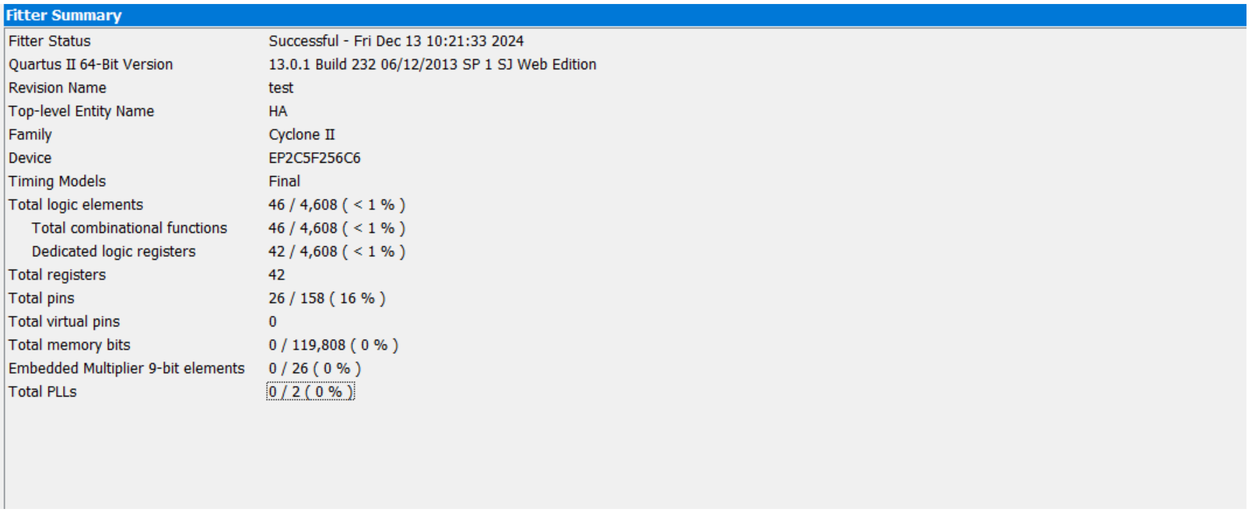
# Post-Synthesis





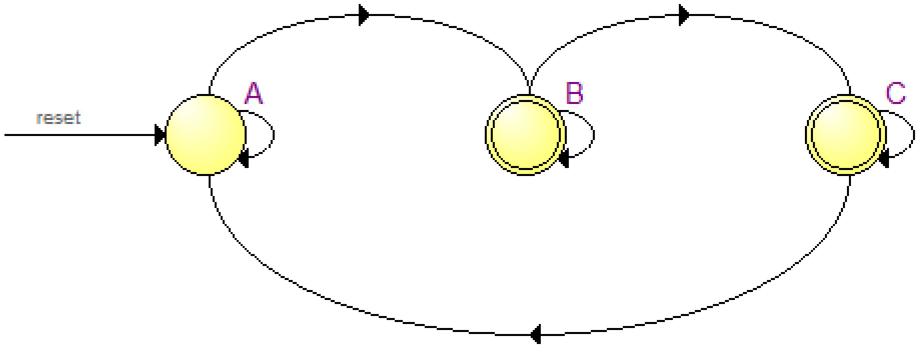


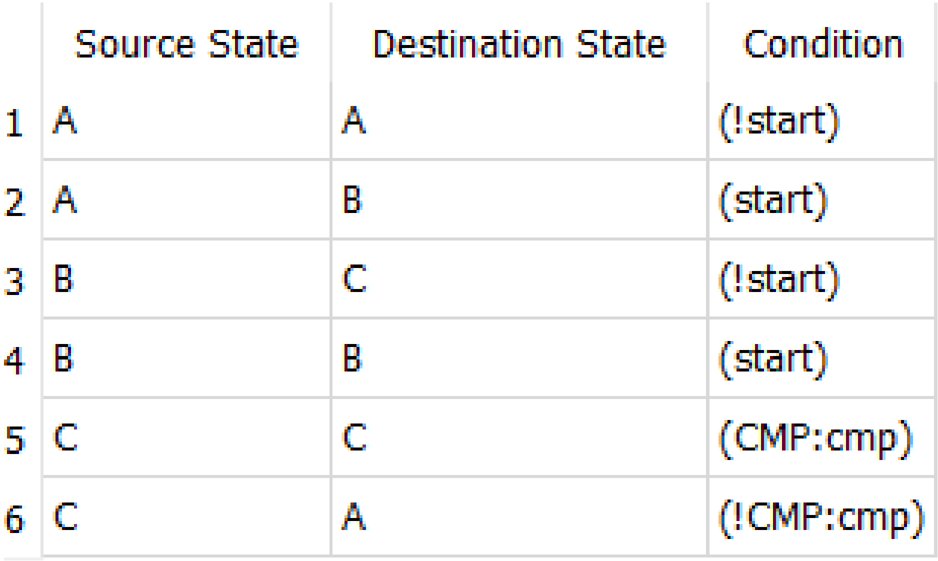
| | |
|---|---|
| Flow Status | Successful - Tue Dec 17 22:25:57 2024 |
| Quartus II 64-Bit Version | 13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition |
| Revision Name | HA |
| Top-level Entity Name | HA |
| Family | Cyclone II |
| Device | EP2C5AF256A7 |
| Timing Models | Final |
| Total logic elements | 46 / 4,608 ( < 1 % ) |
| Total combinational functions | 46 / 4,608 ( < 1 % ) |
| Dedicated logic registers | 42 / 4,608 ( < 1 % ) |
| Total registers | 42 |
| Total pins | 26 / 158 ( 16 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 119,808 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 0 / 26 ( 0 % ) |
| Total PLLs | 0 / 2 ( 0 % ) |



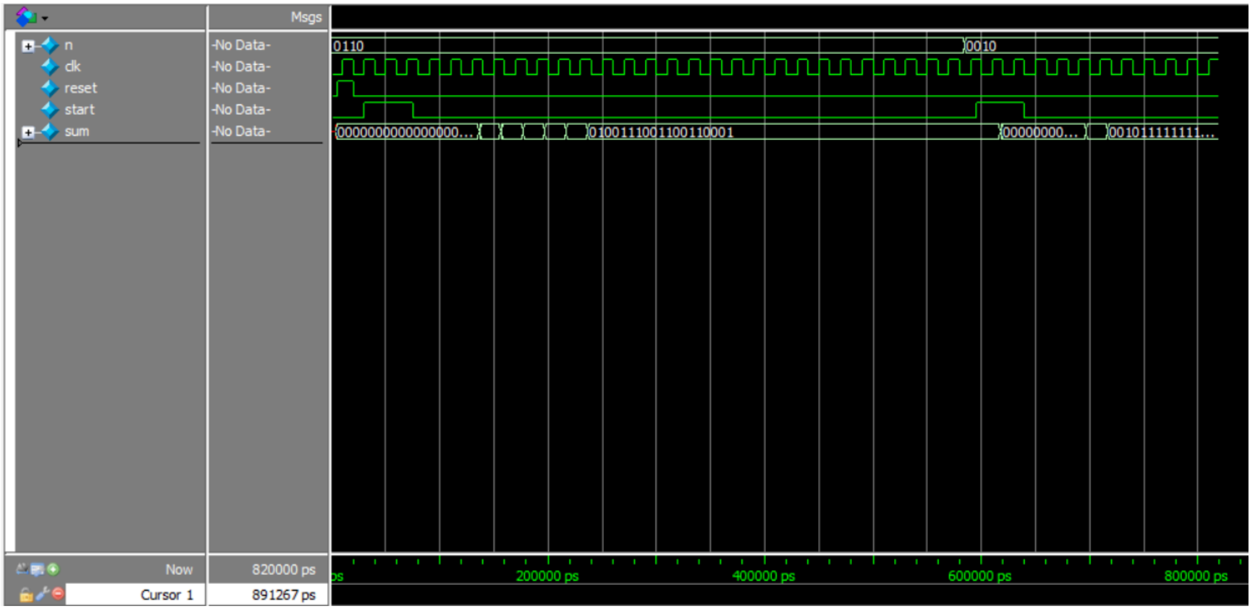
Fitter Summary

| | |
|---|---|
| Fitter Status | Successful - Fri Dec 13 10:21:33 2024 |
| Quartus II 64-Bit Version | 13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition |
| Revision Name | test |
| Top-level Entity Name | HA |
| Family | Cyclone II |
| Device | EP2C5F256C6 |
| Timing Models | Final |
| Total logic elements | 46 / 4,608 ( < 1 % ) |
| Total combinational functions | 46 / 4,608 ( < 1 % ) |
| Dedicated logic registers | 42 / 4,608 ( < 1 % ) |
| Total registers | 42 |
| Total pins | 26 / 158 ( 16 % ) |
| Total virtual pins | 0 |
| Total memory bits | 0 / 119,808 ( 0 % ) |
| Embedded Multiplier 9-bit elements | 0 / 26 ( 0 % ) |
| Total PLLs | 0 / 2 ( 0 % ) |





|   | Source State | Destination State | Condition |
|---|---|---|---|
| 1 | A | A | (!start) |
| 2 | A | B | (start) |
| 3 | B | C | (!start) |
| 4 | B | B | (start) |
| 5 | C | C | (CMP:cmp) |
| 6 | C | A | (!CMP:cmp) |


Msgs
n
clk
reset
start
sum
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
0110
0010
0000000000000000...
0100111001100110001
00000000...
001011111111...
Now
820000 ps
Cursor 1
891267 ps
ps
200000 ps
400000 ps
600000 ps
800000 ps