به نام خدا

پروژه کامپیوتری ۶

سیستم های دیجیتال ۱

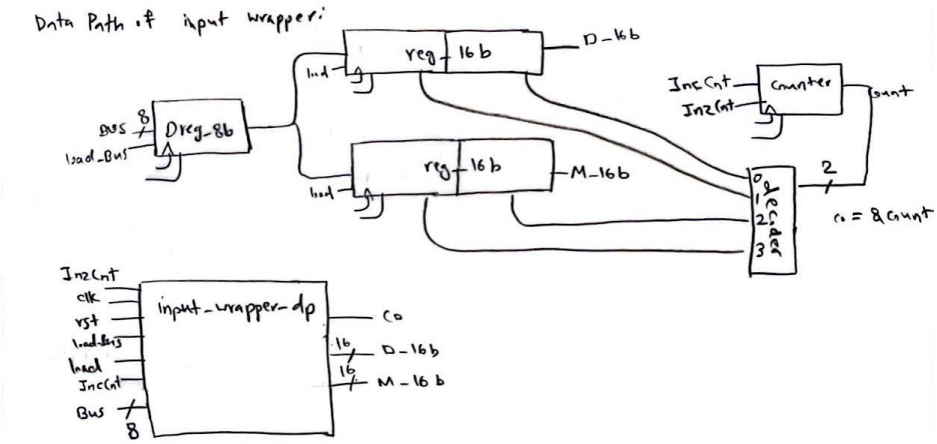
دکتر نوابی

پریسا محمدی ۸۱۰۱۰۱۵۰۹

صورت پروژه

در این پروژه می بایست Input wrapper و output wrapper را برای یک ماژول تقسیم کننده که در پروژه گذشته انجام دادیم طراحی می کردیم. هر کدام از این wrapper ها عملا بخشی از bussing system ما هستند که ولی ما می خواهیم برای هر تعریف هر کدام، یک data path , controller به دست اورده و ان را در زبان سخت افزاری verilog شبیه سازی کرده و این کار را یک بار با نرم افزار Modelsim و بار دیگر پس از compile کردن فایل های پروژه در نرم افزار Quartus و سپس به دست اوردن شبیه سازی با Modelsim کار خود را خاتمه می دهیم.

طراحی data path , controller برای input , output wrappers



طراحی data path برای Input wrapper

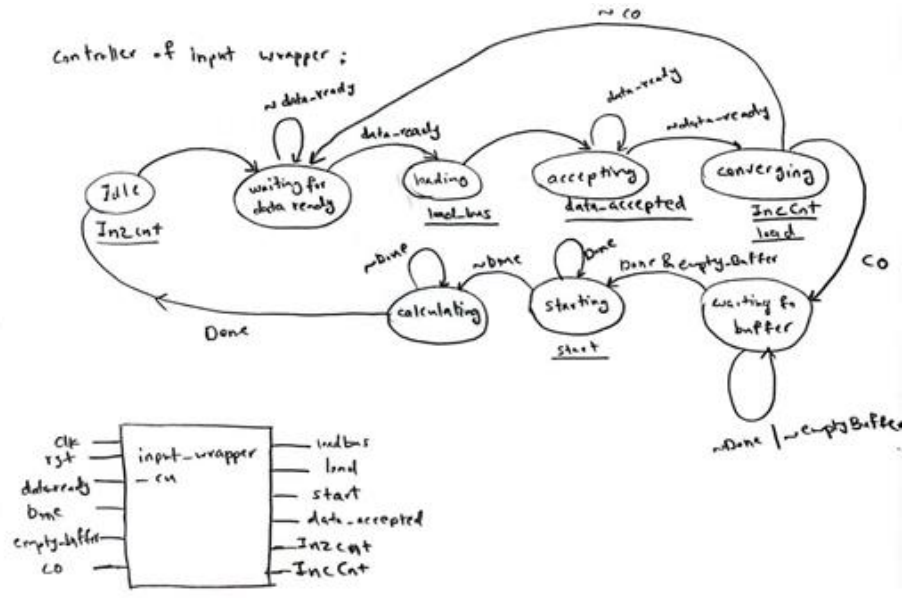
```verilog
module input_wrapper_dp (input clk, rst, load_bus, load, InzCnt, IncCnt, input [7:0] Bus,
                         output co, output reg [15:0] D_16b, M_16b );

    reg [7:0] Dreg_8b;
    reg [1:0] Count;
    always @(posedge clk or posedge rst) begin
        if (rst)
            Dreg_8b <= 8'b0;
        else if (load_bus)
            Dreg_8b <= Bus;
    end
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            D_16b <= 16'b0;
            M_16b <= 16'b0;
        end else if(load) begin
            case(Count)
                2'd0: D_16b[7:0] <= Dreg_8b;
                2'd1: D_16b[15:8] <= Dreg_8b;
                2'd2: M_16b[7:0] <= Dreg_8b;
                2'd3: M_16b[15:8] <= Dreg_8b;
            endcase
        end
    end
    always @(posedge clk or posedge rst) begin
        if (rst)
            Count <= 2'b0;
        else if (InzCnt)
            Count <= 2'b0;
        else if (IncCnt)
            Count <= Count + 1;
    end
    assign co = &Count;

endmodule

```

کد verilog برای شبیه سازی data path و بررسی عملکرد طراحی



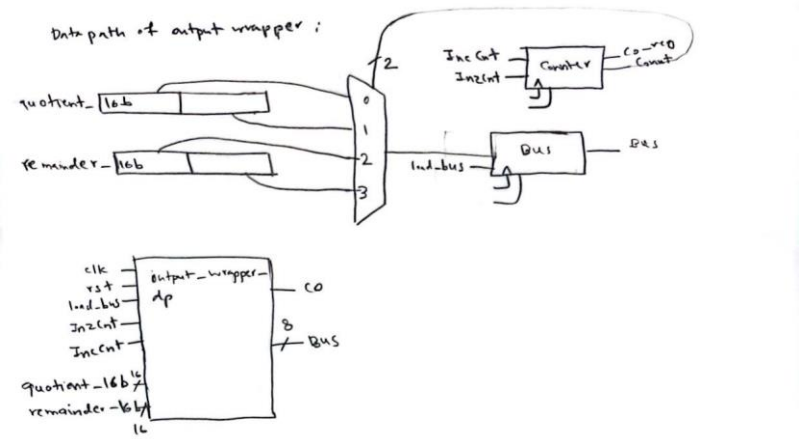
طراحی controller برای input wrapper

```
Ln#
 1  module input_wrapper_cu(input clk, rst, data_ready, Done, empty_buffer, co,
 2                output reg load_bus, load, start, data_accepted, InzCnt, IncCnt);
 3
 4      reg [2:0] pstate, nstate;
 5
 6      parameter [2:0]
 7          Idle = 0,
 8          wating_for_data_ready = 1,
 9          loading = 2,
10          accepting = 3,
11          converging = 4,
12          wating_for_buffer = 5,
13          starting = 6,
14          calculating = 7;
15
16      always @(pstate, data_ready, Done, empty_buffer, co) begin
17          nstate = 0;
18          {load_bus, load, start, data_accepted, InzCnt, IncCnt} = 6'b0;
19
20          case (pstate)
21              Idle: begin
22                  nstate = wating_for_data_ready;
23                  InzCnt = 1'b1;
24              end
25              wating_for_data_ready: begin
26                  nstate = data_ready ? loading : wating_for_data_ready;
27              end
28              loading: begin
29                  nstate = accepting;
30                  load_bus = 1'b1;
31              end
32              accepting: begin
33                  nstate = data_ready ? accepting : converging;
34                  data_accepted = 1'b1;
35
36              end
```

کد verilog برای شبیه سازی controller و بررسی عملکرد طراحی بخش ۱

```
37              converging: begin
38                  nstate = co ? wating_for_buffer : wating_for_data_ready;
39                  IncCnt = 1'b1;
40                  load = 1'b1;
41              end
42              wating_for_buffer: begin
43                  if (~Done | ~empty_buffer) nstate = wating_for_buffer;
44                  else if (Done & empty_buffer) nstate = starting;
45              end
46              starting: begin
47                  nstate = Done ? starting : calculating;
48                  start = 1'b1;
49              end
50              calculating: begin
51                  nstate = Done ? Idle : calculating;
52              end
53
54              default: nstate = Idle;
55          endcase
56      end
57      always @(posedge clk, posedge rst) begin
58          if (rst)
59              pstate <= Idle;
60          else
61              pstate <= nstate;
62      end
63
64
65  endmodule
66
```

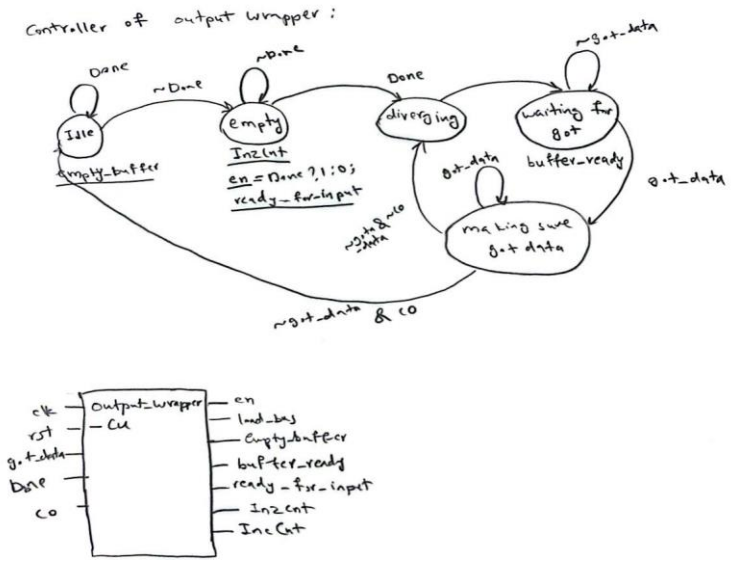
کد verilog برای شبیه سازی controller و بررسی عملکرد طراحی بخش ۲



طراحی data path برای output wrapper

```
Ln#
 1  module output_wrapper_dp (input clk, rst, en, load_bus, InzCnt, IncCnt, input [15:0] quotient_16b, remainder_16b,
 2                            output co, output reg [7:0] Bus );
 3
 4      reg [1:0] Count;
 5      reg co_reg;
 6
 7      always @(posedge clk, posedge rst) begin
 8          if (rst)
 9              {co_reg,Count} <= 3'b0;
10          else if (InzCnt)
11              {co_reg,Count} <= 3'b0;
12          else if (IncCnt)
13              {co_reg,Count} <= Count + 1;
14      end
15
16      always @(posedge clk or posedge rst) begin
17          if (rst) begin
18              Bus <= 8'b0;
19          end else if(load_bus) begin
20              case(Count)
21                  2'd0: Bus <= quotient_16b[15:8];
22                  2'd1: Bus <= quotient_16b[7:0];
23                  2'd2: Bus <= remainder_16b[15:8];
24                  2'd3: Bus <= remainder_16b[7:0];
25              endcase
26          end
27      end
28
29      assign co = co_reg;
30
31  endmodule
```

کد verilog برای شبیه سازی data path و بررسی عملکرد طراحی



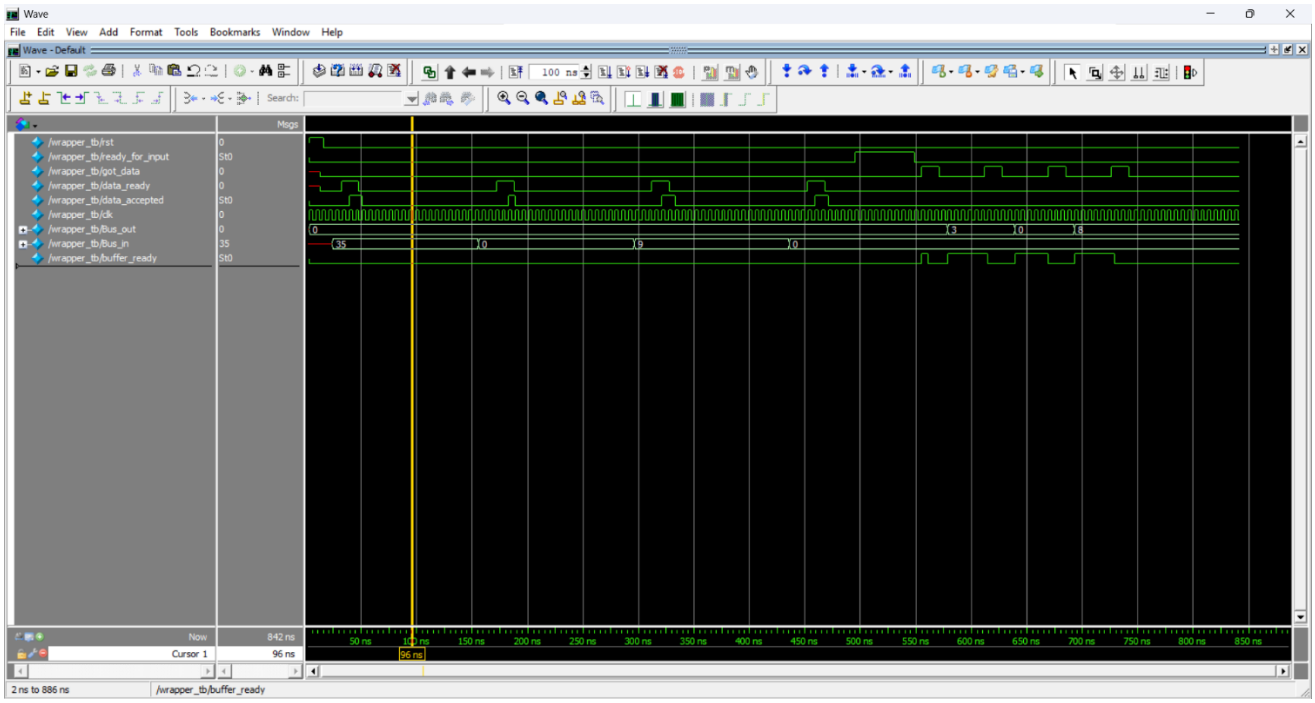
طراحی controller برای output wrapper

```
Ln#
 1  module output_wrapper_cu(input clk, rst, got_data, Done, co,
 2                           output reg en, load_bus, empty_buffer, buffer_ready, ready_for_input, InzCnt, IncCnt);
 3
 4      reg [2:0] pstate, nstate;
 5
 6      parameter [2:0]
 7          Idle = 0,
 8          empty = 1,
 9          diverging = 2,
10          waiting_for_got = 3,
11          making_sure_got_data = 4;
12
13      always @(pstate, got_data, Done, co) begin
14          nstate = 0;
15          {en, load_bus, empty_buffer, ready_for_input, buffer_ready, InzCnt, IncCnt} = 7'b0;
16
17          case (pstate)
18              Idle:begin
19                  nstate = Done ? Idle : empty;
20                  empty_buffer = 1'b1;
21              end
22              empty: begin
23                  nstate = Done ? diverging : empty;
24                  InzCnt = 1'b1;
25                  en = Done ? 1 : 0;
26                  ready_for_input = 1'b1;
27              end
28              diverging: begin
29                  nstate = waiting_for_got;
30                  load_bus = 1'b1;
31                  IncCnt = 1'b1;
32              end
```

کد verilog برای شبیه سازی controller و بررسی عملکرد طراحی بخش ۱

```
33              waiting_for_got: begin
34                  nstate = got_data ? making_sure_got_data : waiting_for_got;
35
36                  buffer_ready = 1'b1;
37              end
38              making_sure_got_data: begin
39                  if (got_data) nstate = making_sure_got_data;
40                  else if (~got_data & ~co) nstate = diverging;
41                  else if (~got_data & co) nstate = Idle;
42
43              end
44
45              default: nstate = Idle;
46          endcase
47      end
48      always @(posedge clk, posedge rst) begin
49          if (rst)
50              pstate <= Idle;
51          else
52              pstate <= nstate;
53      end
54
55  endmodule
```

کد verilog برای شبیه سازی controller و بررسی عملکرد طراحی بخش 2



خروجی simulation به دست امده برای بررسی عملکرد top module

همان طور که در خروجی مشخص است در تست بنچ به ترتیب ۴ تا ۸ بیت را که برای مقدار مقسوم و مقسوم علیه ۱۶ بیتی امان لازم داریم را می دهیم و باید دقت داشته باشیم که در تست بنچ بعد از مقدار دهی به مقسوم ومقسوم علیه بایستی به عنوان ورودی سیگنال data ready را اعلام کنیم تا که گیرنده متوجه شود که فرستنده اماده است تا هر دو تا ۱۶ بیت اش را ارسال

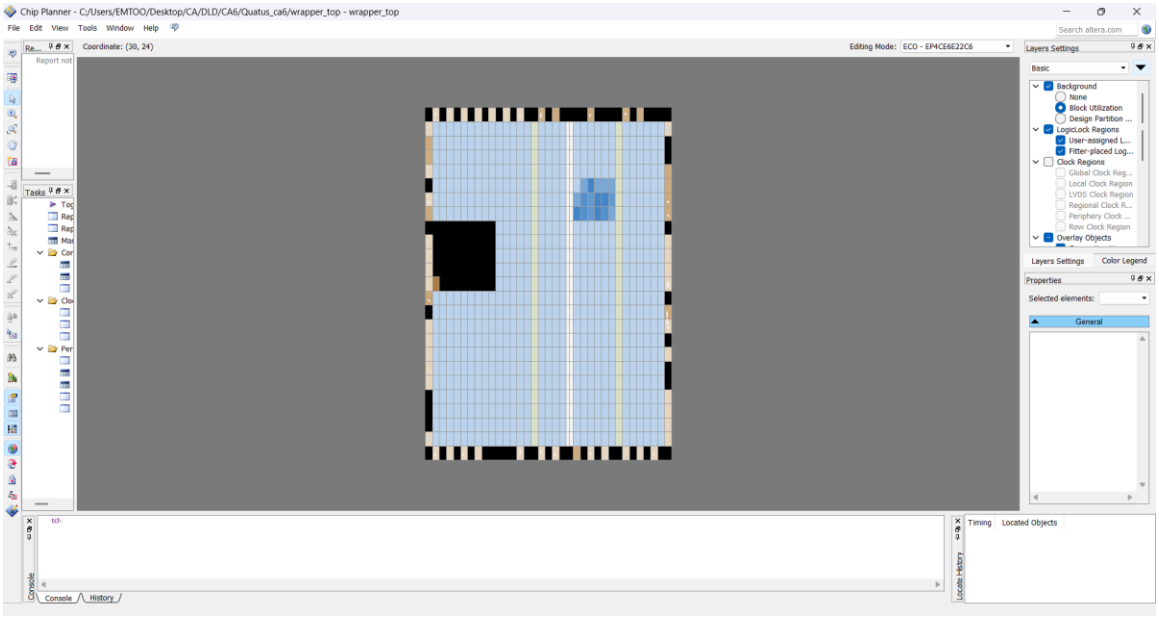
کند. و در خروجی نیز به ترتیب ۴ تا ۸ بیت برای مقادیر خارج قسمت و باقی مانده داریم که حاصل ماژول تقسیم کننده ای هستند که در پروژه قبل زدیم.

برای بررسی سیگنال ها داخلی input , output wrapper می توان سیگنال های زیر را در wave بررسی کرد تا به صحت عملکرد طراحی خود در بخش های مختلف اطمینان حاصل کنیم.

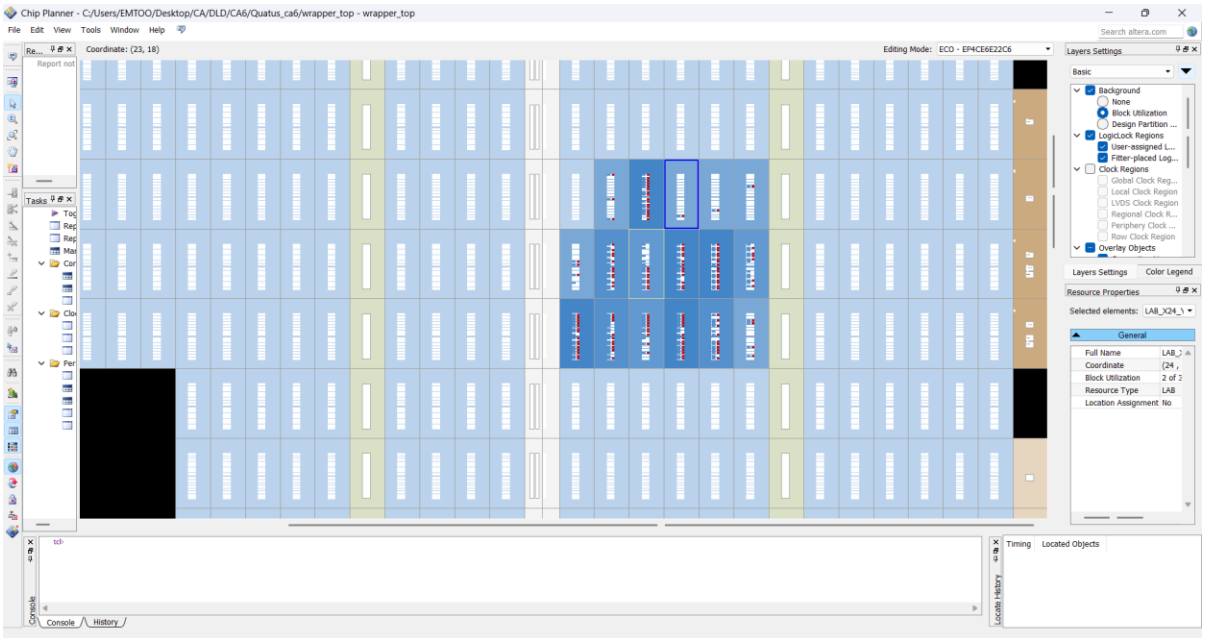
sim:/wrapper_tb/rst sim:/wrapper_tb/ready_for_input sim:/wrapper_tb/got_data
sim:/wrapper_tb/data_ready sim:/wrapper_tb/data_accepted sim:/wrapper_tb/clk
sim:/wrapper_tb/Bus_out sim:/wrapper_tb/Bus_in sim:/wrapper_tb/buffer_ready
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/INPUT_WRAPPER_DP/D_16b
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/INPUT_WRAPPER_DP/M_16b
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/INPUT_WRAPPER_DP/co
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/INPUT_WRAPPER_CU/pstate
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/load
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/INPUT_WRAPPER_CU/empty_buffer
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/INPUT_WRAPPER_CU/Done
sim:/wrapper_tb/UUT/INPUT_WRAPPER_TOP/INPUT_WRAPPER_CU/start
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/OUTPUT_WRAPPER_CU/pstate
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/OUTPUT_WRAPPER_CU/empty_buffer
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/OUTPUT_WRAPPER_CU/Done
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/quotient_16b
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/remainder_16b
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/OUTPUT_WRAPPER_DP/load_bus
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/OUTPUT_WRAPPER_DP/co
sim:/wrapper_tb/UUT/OUTPUT_WRAPPER_TOP/OUTPUT_WRAPPER_DP/Count

سیگنال های داخلی wrapper ها برای بررسی عملکرد صحیح ماژول ها

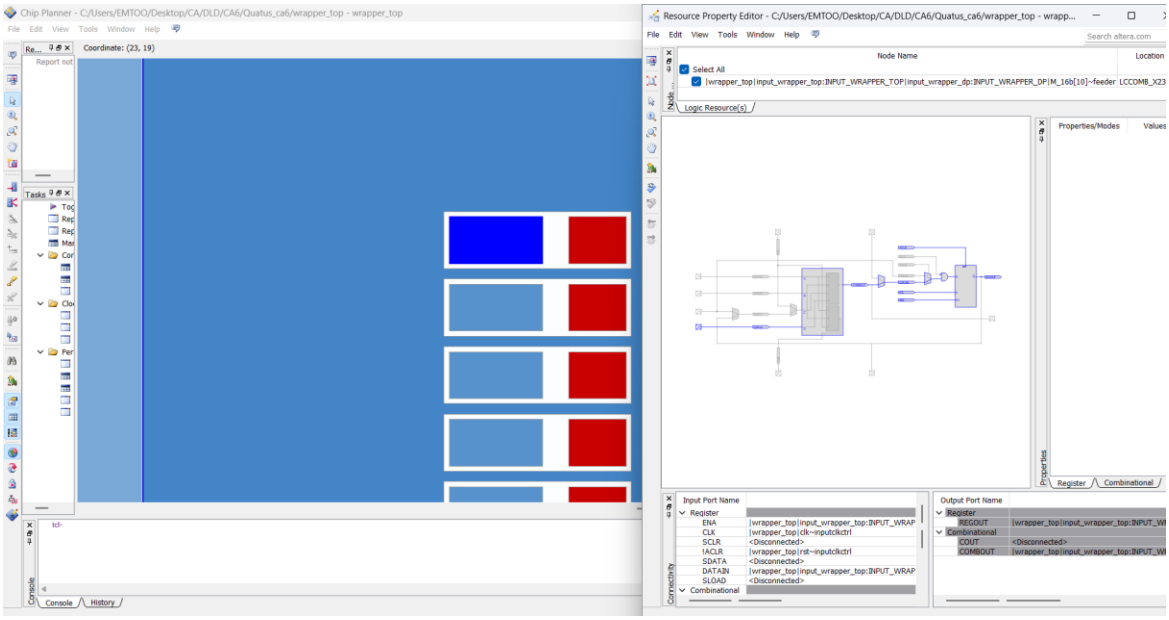
نتایجpost_synthو خروجی های کامپایل کردن در کوارتز :



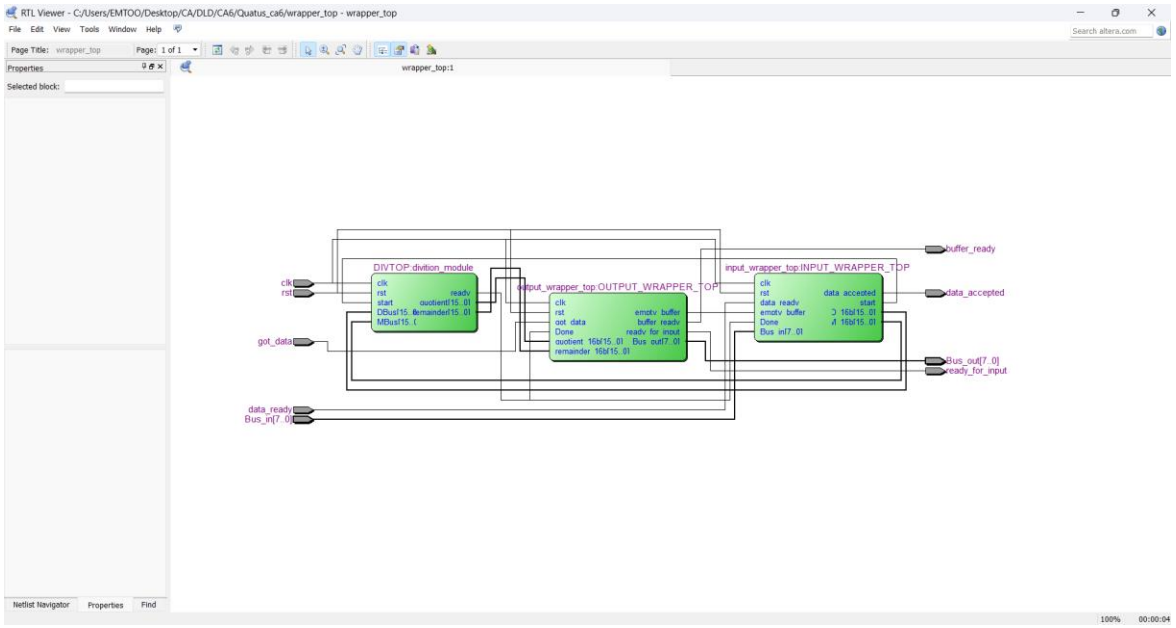
بخش های اشغال شده در fpga انتخابی ما توسط تمامی ماژول ها استفاده شده در این طراحی



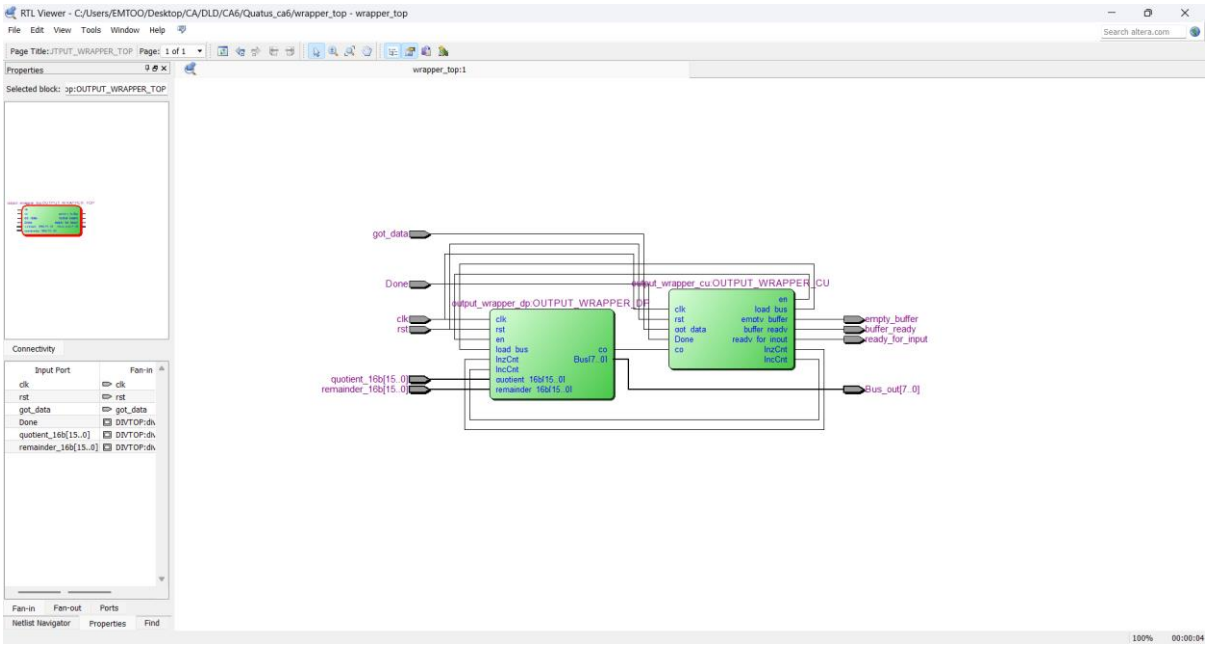
زیر بخش های اشغال شده در fpga انتخابی ما توسط تمامی ماژول ها استفاده شده در این طراحی



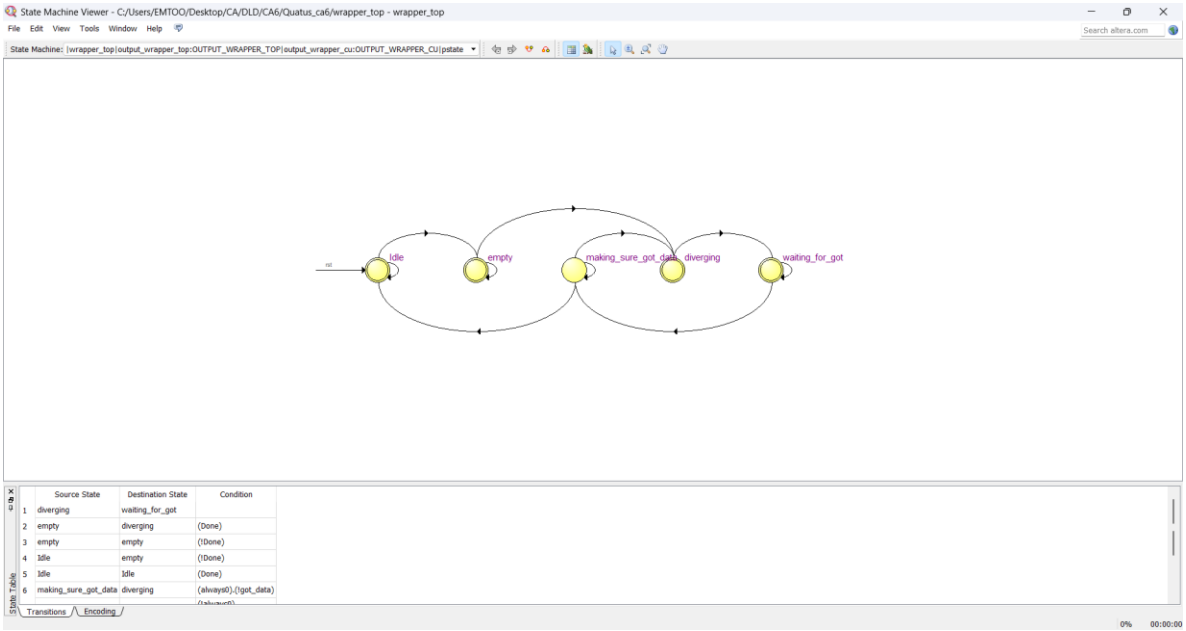
داخل هر بلوک از logic block های بخش های استفاده شده از fpga



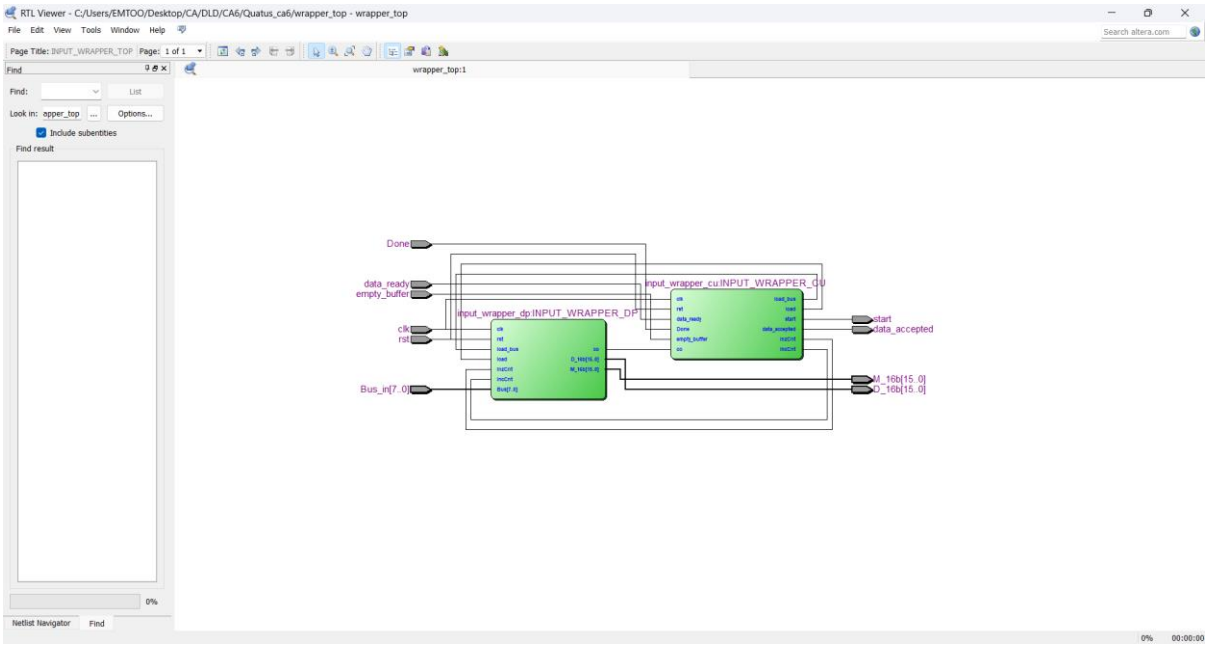
ماژول های استفاده شده در fpga برای ایجاد top module



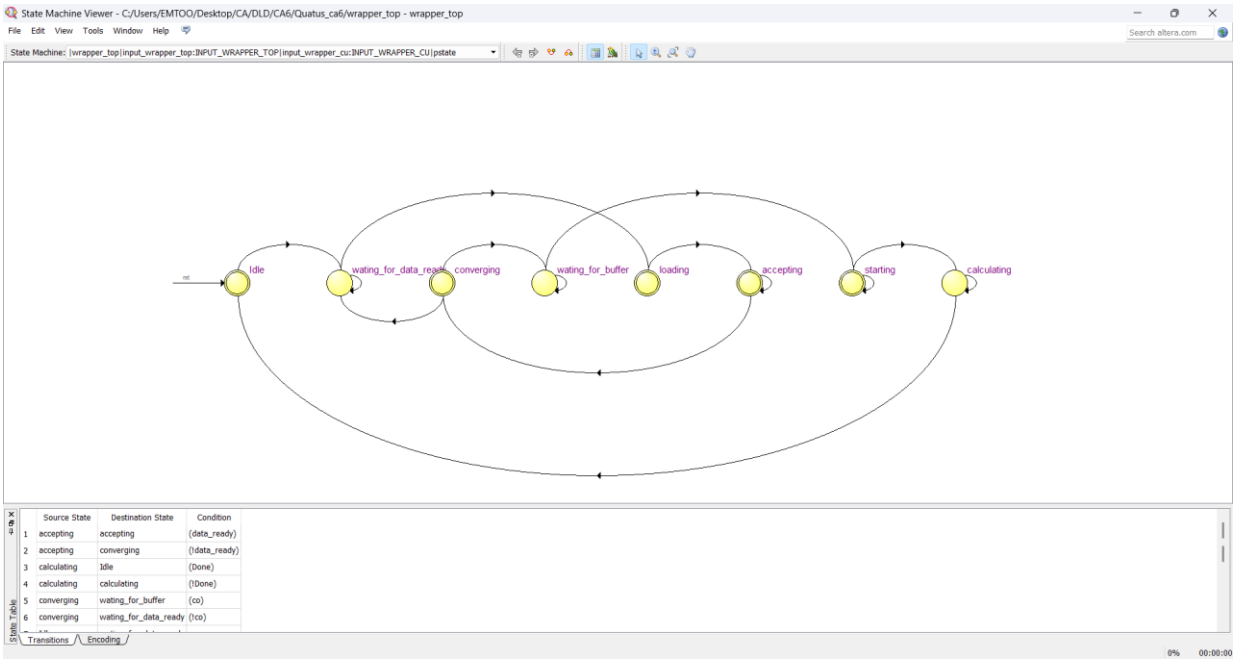
ماژول های تشکیل دهنده input wrapper در fpga



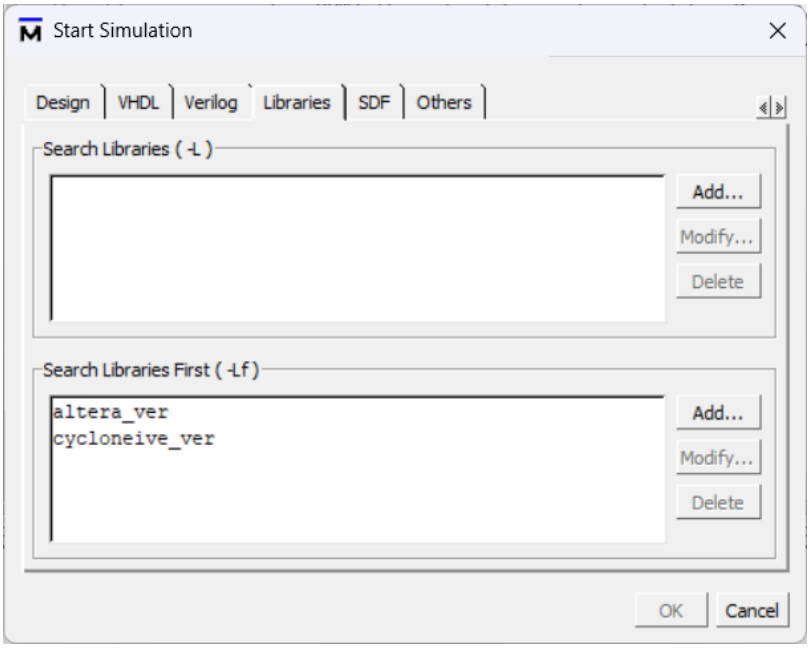
ساختار کنترلر موجود در input wrapper



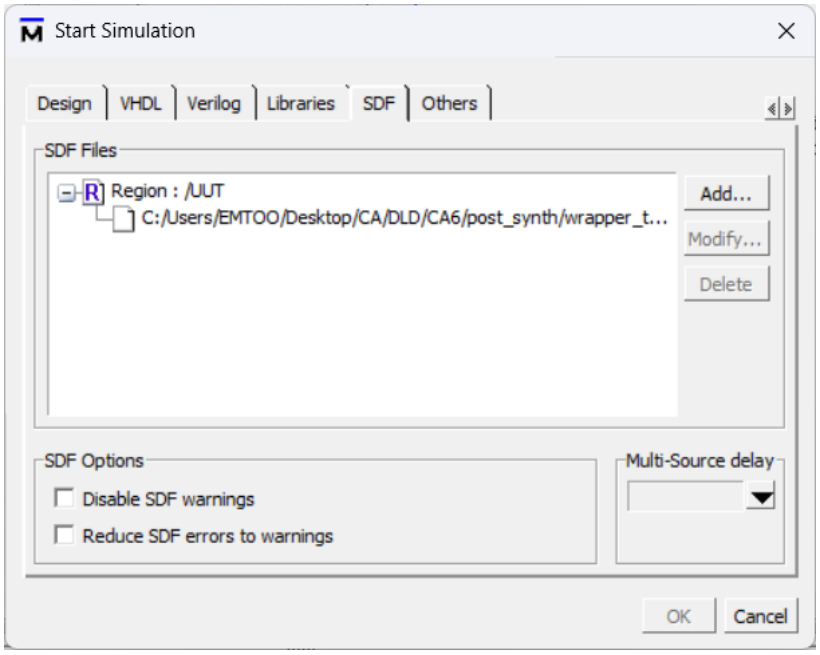
ماژول های تشکیل دهنده output wrapper در fpga



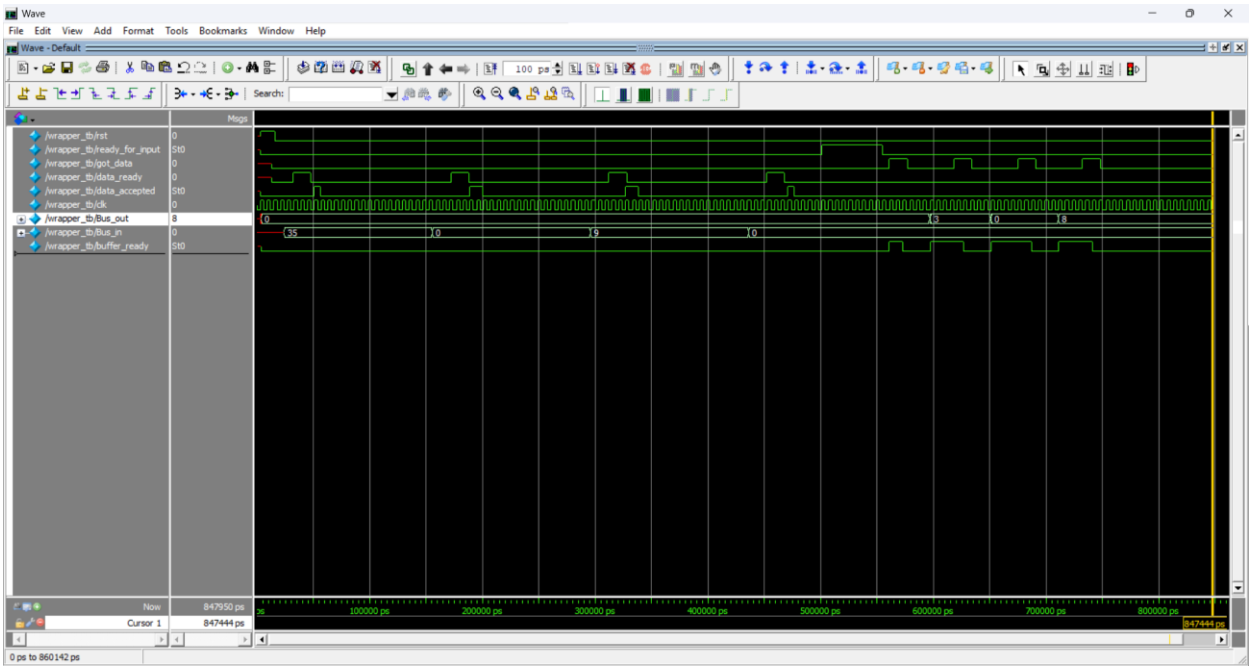
ساختار کنترلر موجود در output wrapper



کتابخانه های مورد استفاده برای simulation فایل های خروجی Quartus



اضافه کردن standard delay output(sdo) برای simulatio



خروجی simulation حاصل از Post synth

تفاوت این خروجی با خروجی شبیه سازی که بدون کمک quartus انجام شده بود این است که پس از هر کلاک برای دیدن نتیجه اتفاق ای که قبلا بر روی posedge کلاک انجام می شد الان با یک مقداری delay روبه رو هستیم. ولی خروجی همچنان به ازای ورودی یکسان از دفعه قبل همچنان ثابت و صحیح است و تغییری نکرده است.

پایان