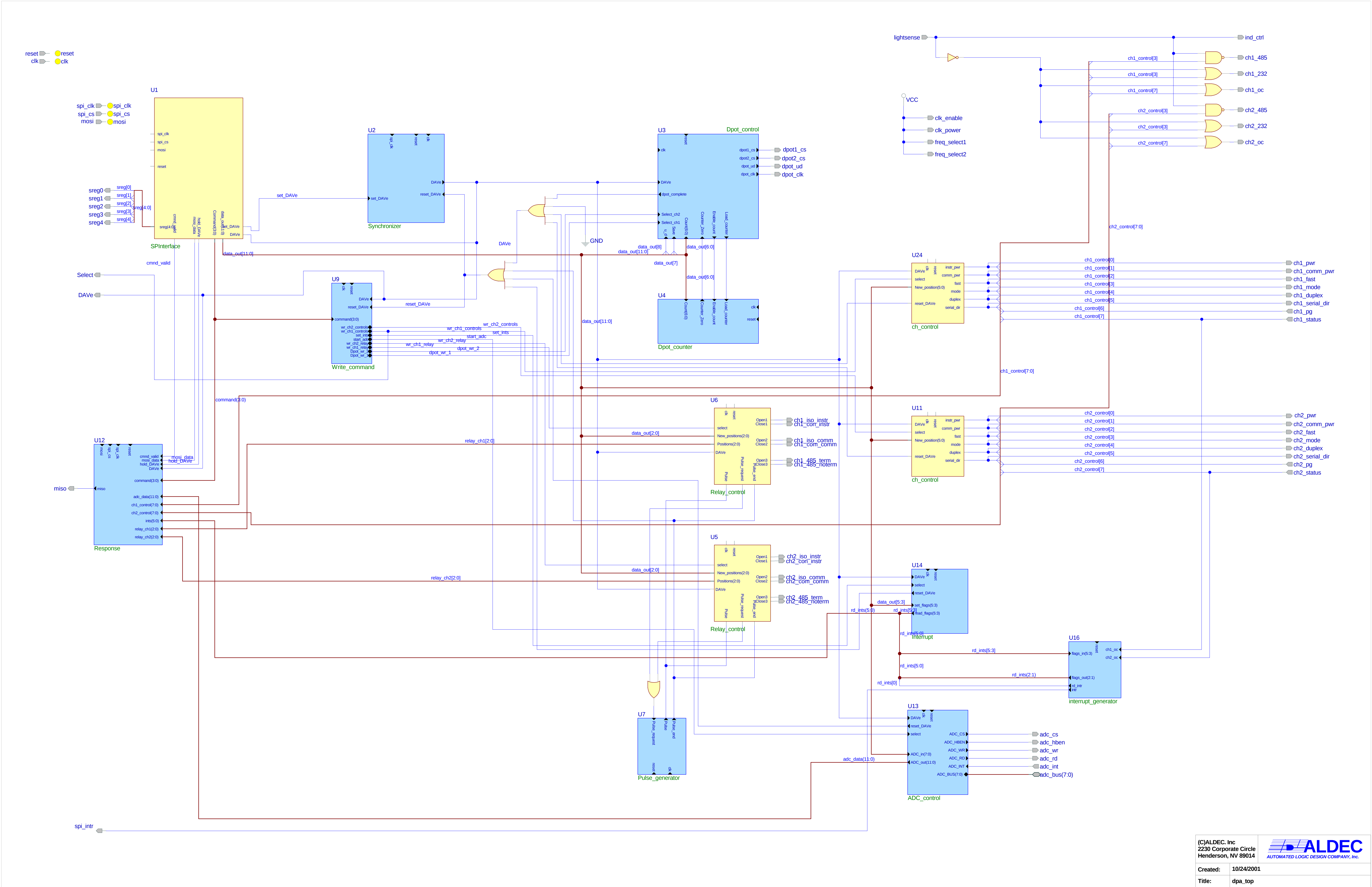




```
//diagram ACTIONS
assign ADC_BUS = (CurrState_Sreg0 == `S2) ? ADC_in : 8'bz;
```

Module: ADC_control

DAVe

clk

reset

ADC_in[7:0]

ADC_INT

select

ADC_BUS[7:0]

ADC_out[11:0]

reset_DAVe

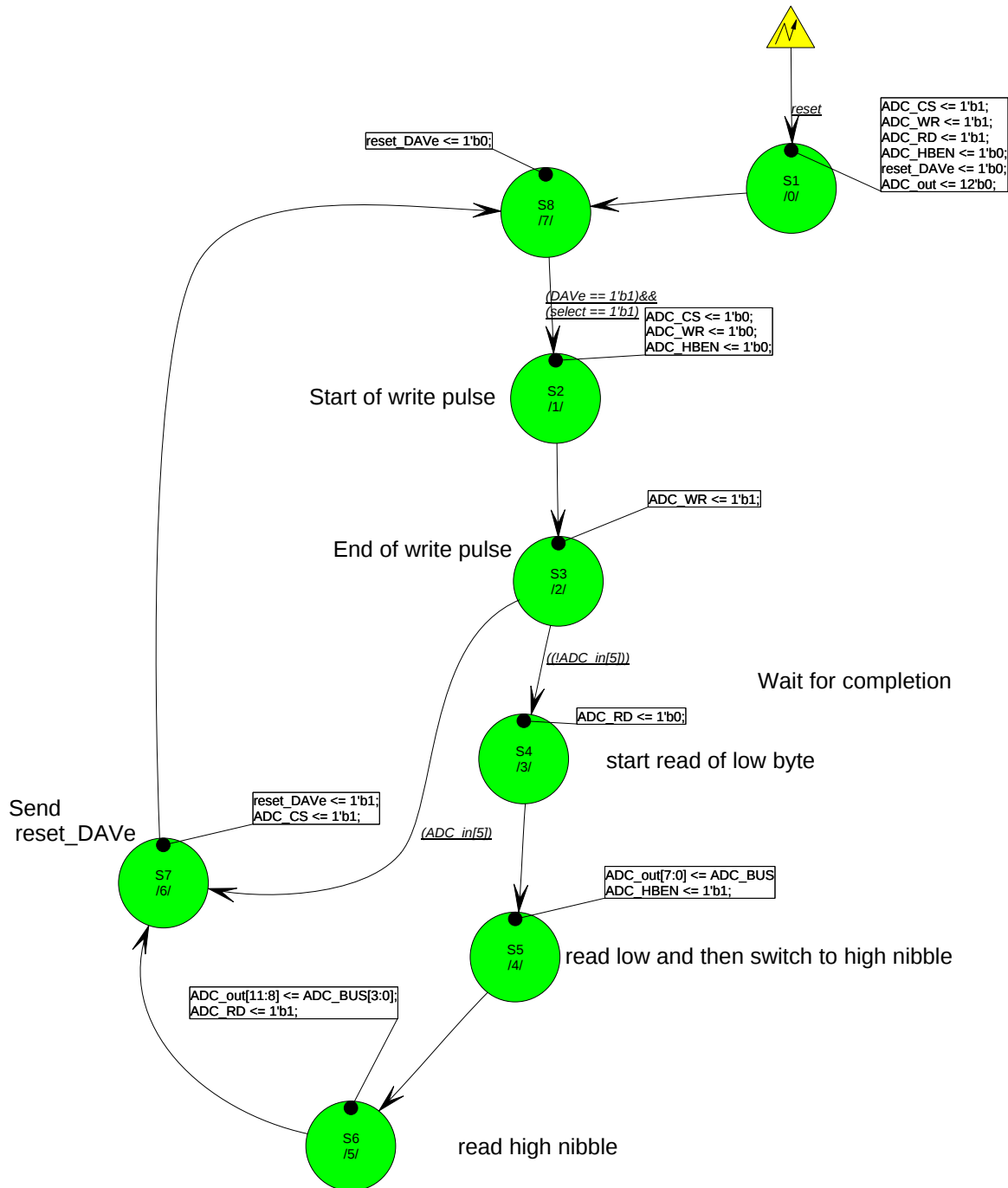
ADC_CS

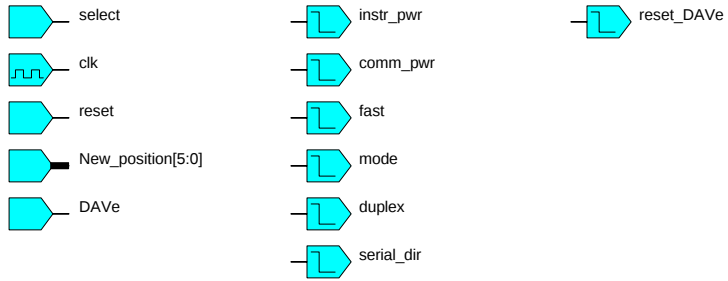
ADC_HBEN

ADC_WR

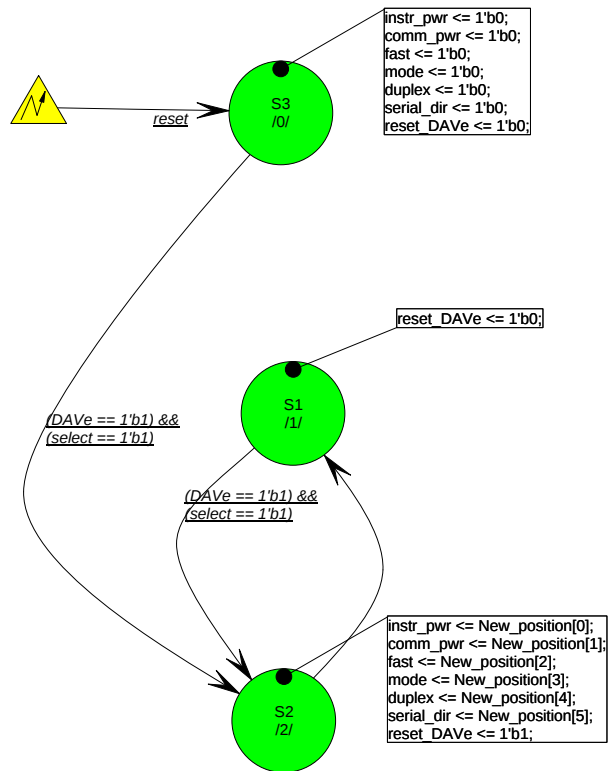
ADC_RD

Sreg0

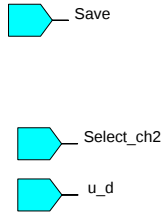
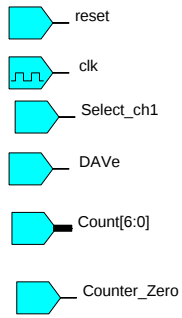




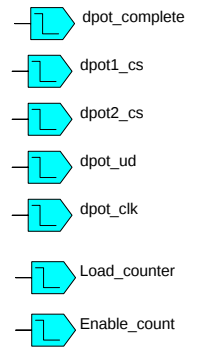
Sreg0



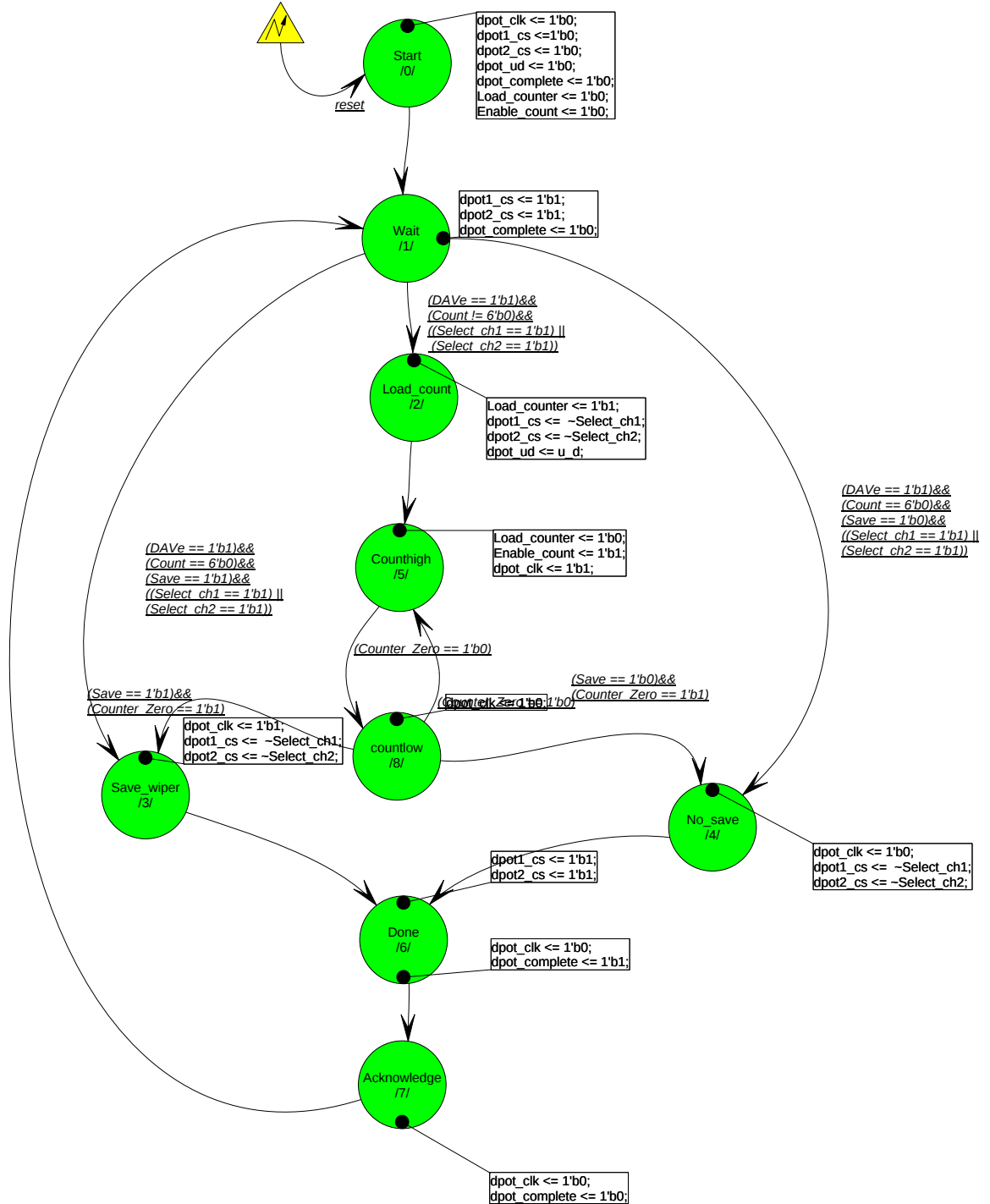
Module: Dpot_control



```
//diagram ACTIONS
//assign dpot_clk = (CurrState_Sreg0 = 3'd3) ? Save :
//                  (CurrState_Sreg0 = 3'd5) ? inc_clk ;
```



Sreg0

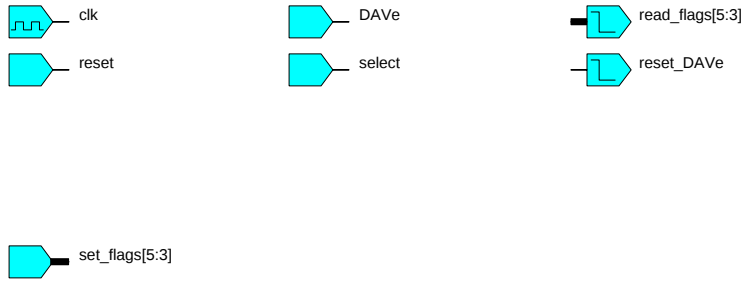


File: g:\VSS_Projects\work\sjensen\DPA\HDL_project\src\Dpot_counter.v

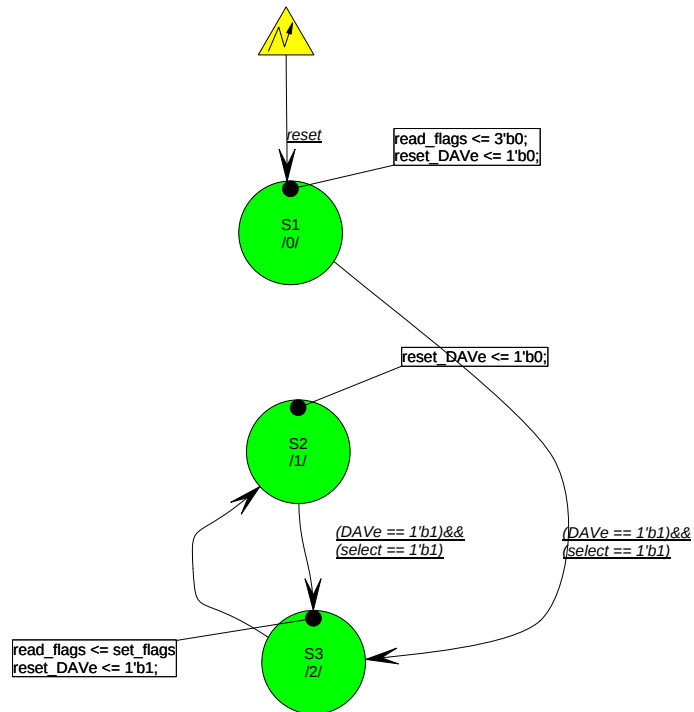
```
1  *****
   ****//
2  // file : c:\projects\nmc\DPA\hdl\DPainterface\src\Dpot_counter.v
3  // created Tue Oct 30 06:46:09 2001
4  // by ITF2V generator version 1.0
5  *****
   ****//
6
7  `timescale 1ps / 1ps
8
9  //{{ Section below this comment is automatically maintained
10 // and may be overwritten
11 //{module {Dpot_counter}}
12 module Dpot_counter ( Count ,Counter_Zero ,Enable_count ,
   Load_counter ,clk ,reset );
13
14 input [6:0] Count ;
15 wire [6:0] Count ;
16 input Enable_count ;
17 wire Enable_count ;
18 input Load_counter ;
19 wire Load_counter ;
20 input clk ;
21 wire clk ;
22 input reset ;
23 wire reset ;
24
25 output Counter_Zero ;
26 wire Counter_Zero ;
27
28 //}} End of automatically maintained section
29
30 // -- Enter your statements here -- //
31
32 reg [7:0] counter;
33
34 assign Counter_Zero = (counter <= 7'd1); //this flags end of
   counting to dpot control
35
36 always @ (posedge reset or posedge clk)
37     begin
38         if (reset) counter <= 7'd0; //reset counter
39         else if (Load_counter) counter <= {Count, 1'b0}; //if it is
   load time then load with 2n+1
40         else if (Enable_count) //if counter is enabled count to zero
   and hold
41             if (counter != 7'd0) counter <= counter - 7'b1;
42             else counter <= 7'd0;
43         else counter <= counter; //just hold till enable counter
44     end
45
46 endmodule
47
```

//diagram ACTIONS

Module: Interrupt



Sreg0



File: g:\VSS_Projects\work\sjensen\DPA\HDL_project\src\interrupt_generator.v

```
1  //*****
   ****//
2  // file : G:\VSS_Projects\work\sjensen\DPA\hdl\DPainterface\src\interrupt_gene
   rator.v
3  // created Wed Oct 31 13:17:43 2001
4  // by ITF2V generator version 1.0
5  //*****
   ****//
6
7  `timescale 1ps / 1ps
8
9  //{{ Section below this comment is automatically maintained
10 // and may be overwritten
11 //{module {interrupt_generator}}
12 module interrupt_generator ( ch1_oc ,ch2_oc ,flags_in ,flags_out ,
   intr ,rd_intr ,reset );
13
14 input ch1_oc ;
15 wire ch1_oc ;
16 input ch2_oc ;
17 wire ch2_oc ;
18 input [5:3] flags_in ;
19 wire [5:3] flags_in ;
20 input reset ;
21 wire reset ;
22
23 output [2:1] flags_out ;
24 wire [2:1] flags_out ;
25 output intr ;
26 wire intr ;
27 output rd_intr ;
28 wire rd_intr ;
29
30 //}} End of automatically maintained section
31
32 // -- Enter your statements here -- //
33
34 wire interim;
35
36 assign flags_out[1] = (!ch1_oc);
37 assign flags_out[2] = (!ch2_oc);
38 assign rd_intr = (flags_out[1] | flags_out[2]);
39 assign intr = ((flags_in[3]==1'b1) && ((flags_out[1] & flags_in[4])
   || (flags_out[2] & flags_in[5]))) ? 1'b0 : 1'bz;
40
41
42 endmodule
43
```

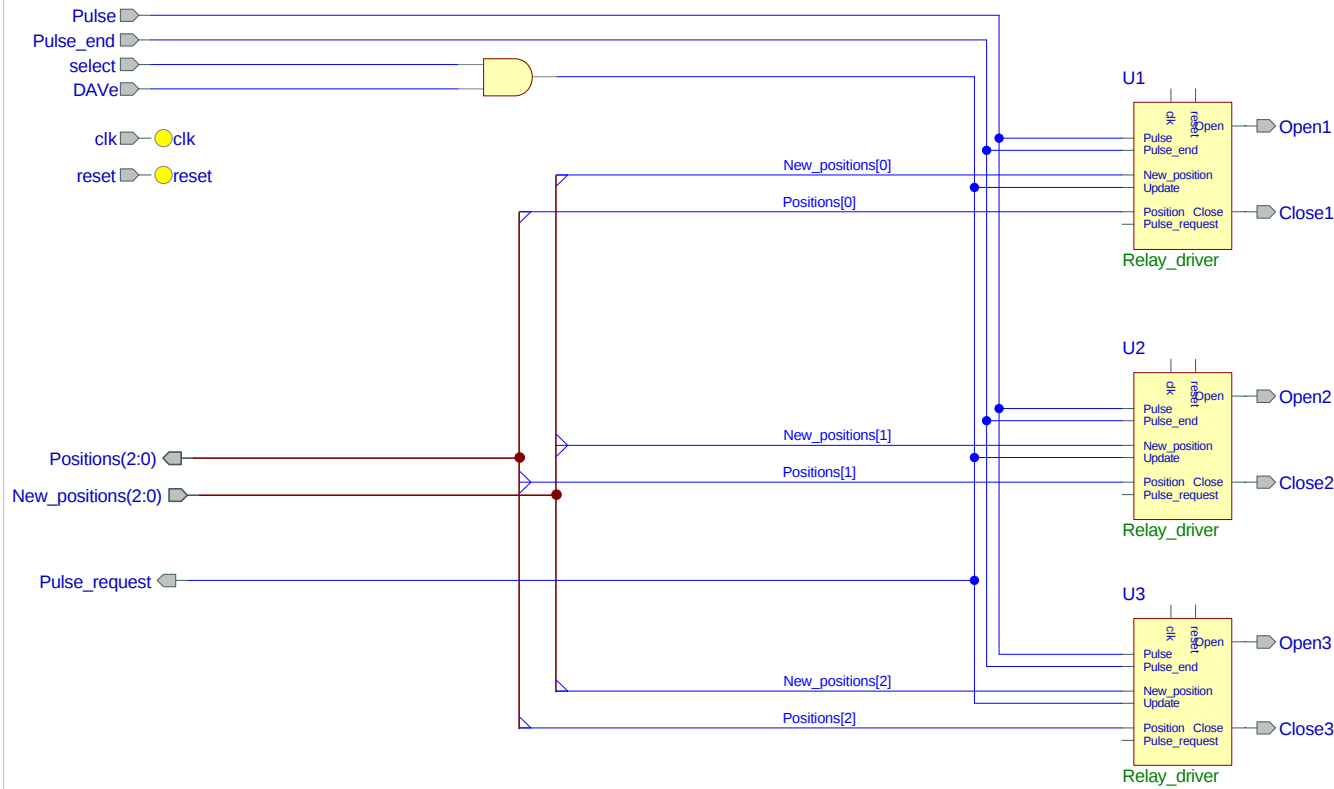
File: g:\VSS_Projects\work\sjensen\DPA\HDL_project\src\Pulse_generator.v

```
1  *****
   ****//
2  // file : c:\projects\nmc\DPA\hdl\DPAinterface\src\Pulse_generator.v
3  // created Tue Oct 30 08:18:00 2001
4  // by ITF2V generator version 1.0
5  *****
   ****//
6
7  `timescale 1ps / 1ps
8
9  //{{ Section below this comment is automatically maintained
10 // and may be overwritten
11 //{module {Pulse_generator}}
12 module Pulse_generator ( Pulse ,Pulse_end ,Pulse_request ,reset ,clk
   );
13
14 input Pulse_request ;
15 wire Pulse_request ;
16 input reset ;
17 wire reset ;
18 input clk ;
19 wire clk ;
20
21 output Pulse ;
22 reg Pulse ;
23 output Pulse_end ;
24 reg Pulse_end ;
25
26 //}} End of automatically maintained section
27
28 // -- Enter your statements here -- //
29 parameter Pulse_start = 11'd1638;
30
31 reg [10:0] count;
32
33 //Whe count == 0 and Pulse_request is true then load counter with
Pulse_start
34 //and start counting down to 0 and then hold.
35 always @ (posedge reset or posedge clk)
36     begin
37         if (reset) count <= 11'b0;
38         else if ((count == 11'b0) && (Pulse_request)) count <=
Pulse_start;
39         else if (count != 11'b0) count <= count - 1'b1;
40         else count <= 11'b0;
41     end
42
43 //as long as count does not equal zero then pulse is true
44 always @ (posedge reset or posedge clk)
45     begin
46         if (reset) Pulse <= 1'b0;
47         else if (count > 11'b1) Pulse <= 1'b1;
48         else Pulse <= 1'b0;
49     end
50
51 //Pulse end is one clock long and is used to update the relay
position
```

File: g:\VSS_Projects\work\sjensen\DPA\HDL_project\src\Pulse_generator.v

```
52  // and clear the DAVE flag
53  always @ (posedge reset or posedge clk)
54      begin
55          if (reset) Pulse_end <= 1'b0;
56          else if (count != 11'd2) Pulse_end <= 1'b0;
57          else Pulse_end <= 1'b1;
58      end
59  endmodule
60
```

Project



(C)ALDEC. Inc 2230 Corporate Circle Henderson, NV 89014		 ALDEC AUTOMATED LOGIC DESIGN COMPANY, Inc.	
Created:	10/30/2001		
Title:	Relay_control		

File: g:\VSS_Projects\work\sjensen\DPA\HDL_project\src\Relay_driver.v

```
1  *****
2  ****//
3  // file : c:\projects\nmc\DPA\hdl\DPAinterface\src\Relay_driver.v
4  // created Tue Oct 30 07:42:01 2001
5  // by ITF2V generator version 1.0
6  *****
7  ****//
8
9  //{{ Section below this comment is automatically maintained
10 // and may be overwritten
11 //{module {Relay_driver}}
12 module Relay_driver ( Close ,New_position ,Open ,Position ,Pulse ,
13   Pulse_end ,Pulse_request ,Update ,clk ,reset );
14   input New_position ;
15   wire New_position ;
16   input Pulse ;
17   wire Pulse ;
18   input Pulse_end ;
19   wire Pulse_end ;
20   input Update ;
21   wire Update ;
22   input clk ;
23   wire clk ;
24   input reset ;
25   wire reset ;
26
27   output Close ;
28   wire Close ;
29   output Open ;
30   wire Open ;
31   output Position ;
32   reg Position ;
33   output Pulse_request ;
34   wire Pulse_request ;
35
36   //}} End of automatically maintained section
37
38   // -- Enter your statements here -- //
39   //when there is a difference in New_position and Position and
40   Update is try then request a pulse;
41   assign Pulse_request = (Position ^ New_position) & Update;
42   //Open/close goes try when a requested pulse is true and the pulse
43   is being generated selected by New_position
44   assign Open = Pulse & Pulse_request & New_position;
45   assign Close = Pulse & Pulse_request & !New_position;
46
47   always @ (posedge reset or posedge clk)
48   begin
49       if (reset) Position <= 1'b0; //at reset set to zero state!
50       else if ((Pulse_end)&&(Update)) Position <= New_position;
51       //at the end of the pulse update the reported position
52       else Position <= Position;
53   end
```

File: g:\VSS_Projects\work\sjensen\DPA\HDL_project\src\Relay_driver.v

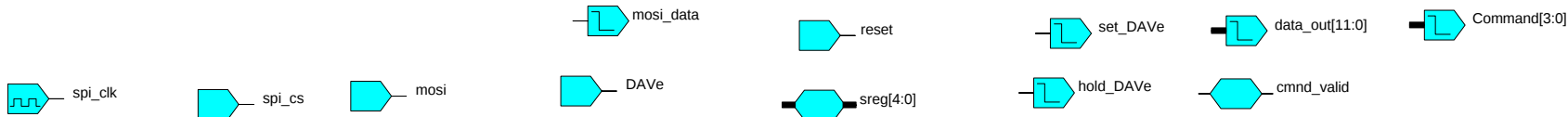
```
52  endmodule
53
```

File: g:\vss_projects\work\sjensen\dpa\hdl_project\src\Response.v

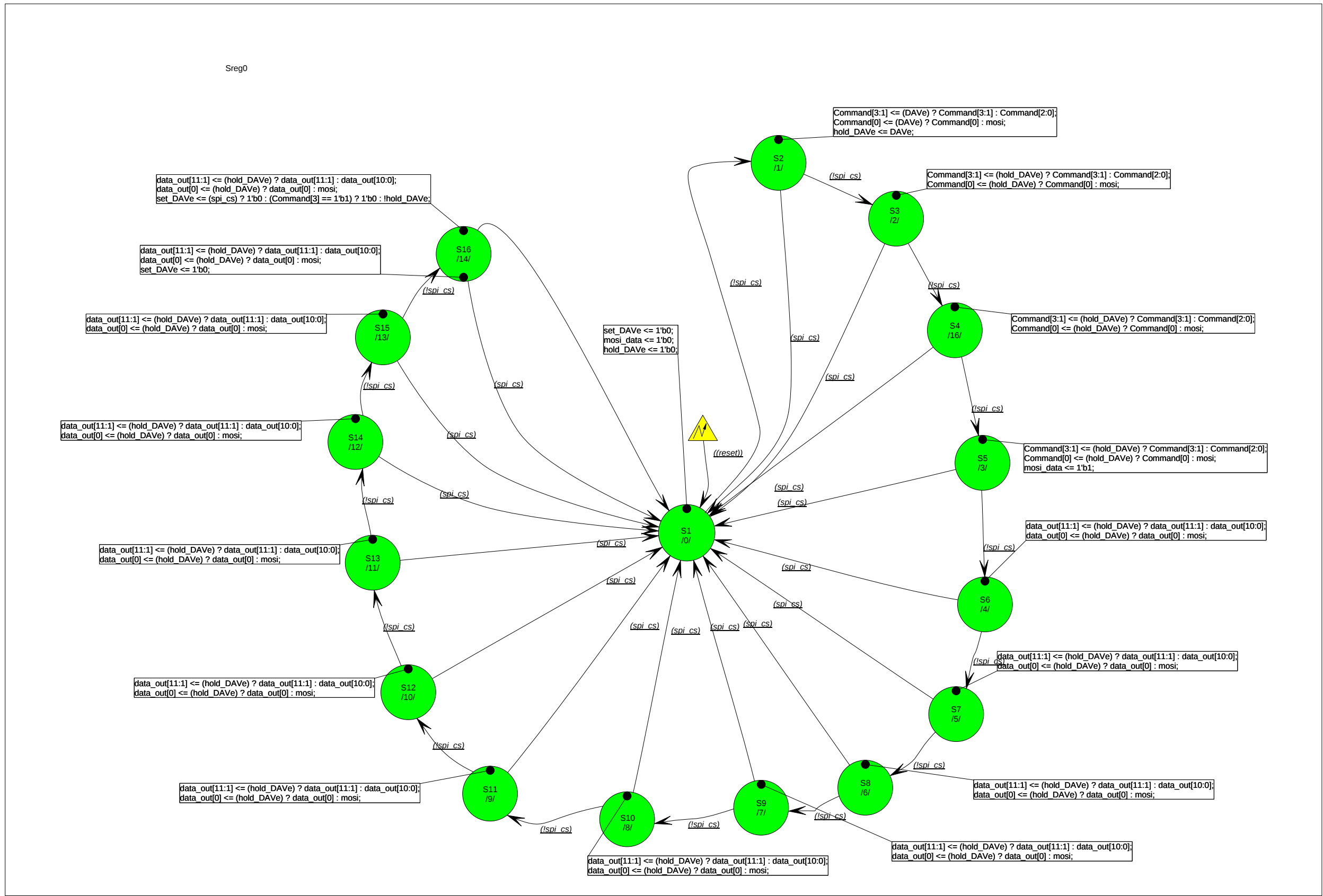
```
1  //*****
   ****//
2  // file : C:\projects\nmc\DPA\hdl\step2\src\Response.v
3  // created Thu Nov 15 06:29:55 2001
4  // by ITF2V generator version 1.0
5  //*****
   ****//
6
7  `timescale 1ps / 1ps
8
9  //{{ Section below this comment is automatically maintained
10 // and may be overwritten
11 //{module {Response}}
12 module Response ( DAVE ,adc_data ,ch1_control ,ch2_control ,
   cmdn_valid ,command ,hold_DAVE ,ints ,miso ,mosi ,mosi_data ,
   relay_ch1 ,relay_ch2 ,reset ,spi_clk ,spi_cs );
13
14   input DAVE ;
15   wire DAVE ;
16   input [11:0] adc_data ;
17   wire [11:0] adc_data ;
18   input [7:0] ch1_control ;
19   wire [7:0] ch1_control ;
20   input [7:0] ch2_control ;
21   wire [7:0] ch2_control ;
22   input cmdn_valid ;
23   wire cmdn_valid ;
24   input [3:0] command ;
25   wire [3:0] command ;
26   input hold_DAVE ;
27   wire hold_DAVE ;
28   input [5:0] ints ;
29   wire [5:0] ints ;
30   input mosi ;
31   wire mosi ;
32   input mosi_data ;
33   wire mosi_data ;
34   input [2:0] relay_ch1 ;
35   wire [2:0] relay_ch1 ;
36   input [2:0] relay_ch2 ;
37   wire [2:0] relay_ch2 ;
38   input reset ;
39   wire reset ;
40   input spi_clk ;
41   wire spi_clk ;
42   input spi_cs ;
43   wire spi_cs ;
44
45   output miso ;
46   wire miso ;
47
48   //}} End of automatically maintained section
49
50   // -- Enter your statements here -- //
51
52   `define DAVE {hold_DAVE,hold_DAVE,hold_DAVE,hold_DAVE,hold_DAVE,
   hold_DAVE,hold_DAVE,hold_DAVE,hold_DAVE,hold_DAVE,hold_DAVE,
   hold_DAVE}
```

```
53
54 reg [11:0] miso_reg;
55 reg miso_pin_reg;
56
57 assign miso = (spi_cs) ? 1'bz : miso_pin_reg;
58
59 always @ (posedge reset or negedge spi_clk) //grab the state of
DAVE!
60     begin
61         if (reset) miso_pin_reg <= 1'b0;
62         else if (mosi_data) miso_pin_reg <= miso_reg[11];
63         else miso_pin_reg <= hold_DAVE;
64     end
65
66 always @ ( posedge reset or posedge spi_clk) //the shift out happens
on negative edges
67     begin
68         if (reset) miso_reg <= 12'b111111111111; //fill the reg with
1's this is the DAVE busy state
69         else if (cmdnd_valid)
70             case ({hold_DAVE, command[2:0], mosi})
71                 5'd0: miso_reg <= 11'b0;
72                 5'd1: miso_reg <= 11'b0;
73                 5'd2: miso_reg <= {9'b0, relay_ch1};
74                 5'd3: miso_reg <= {9'b0, relay_ch2};
75                 5'd4: miso_reg <= 11'b0;
76                 5'd5: miso_reg <= {3'b0, ints};
77                 5'd6: miso_reg <= {6'b0, ch1_control};
78                 5'd7: miso_reg <= {6'b0, ch2_control};
79                 5'd8: miso_reg <= `DAVE;
80
81                 5'd9: miso_reg <= {9'b0, relay_ch1};
82                 5'd10: miso_reg <= {9'b0, relay_ch2};
83                 5'd11: miso_reg <= adc_data;
84                 5'd12: miso_reg <= {6'b0, ints};
85                 5'd13: miso_reg <= {4'b0, ch1_control};
86                 5'd14: miso_reg <= {4'b0, ch2_control};
87                 5'd15: miso_reg <= `DAVE;
88                 default: miso_reg <= 12'b111111111111;
89
90 //interface busy send all 1's
91             endcase
92         else
93             begin
94                 miso_reg[11:1] <= miso_reg[10:0];
95                 miso_reg[0] <= 1'b1;
96             end
97     end
98 endmodule
```

Module: SPInterface



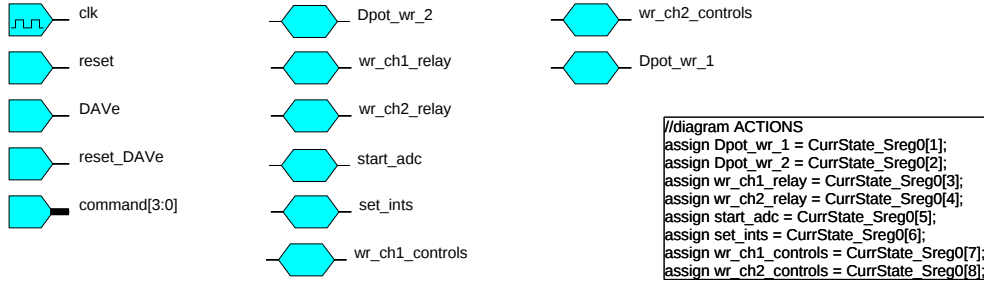
```
// diagram ACTION
assign cmdnd_valid = CurrState_Sreg0[4];
assign sreg = CurrState_Sreg0;
```



File: g:\VSS_Projects\work\sjensen\DPA\hdl_project\src\Synchronizer.v

```
1  //****************************************************************
   ****//
2  // file : c:\projects\nmc\DPA\hdl\DPAinterface\src\Synchronizer.v
3  // created Tue Oct 30 07:13:30 2001
4  // by ITF2V generator version 1.0
5  //****************************************************************
   ****//
6
7  `timescale 1ps / 1ps
8
9  //{{ Section below this comment is automatically maintained
10 // and may be overwritten
11 //{module {Synchronizer}}
12 module Synchronizer ( DAVE ,clk ,reset_DAVE ,reset ,set_DAVE ,
   spi_clk );
13
14 input clk ;
15 wire clk ;
16 input reset_DAVE ;
17 wire reset_DAVE ;
18 input reset ;
19 wire reset ;
20 input set_DAVE ;
21 wire set_DAVE ;
22 input spi_clk ;
23 wire spi_clk ;
24
25 output DAVE ;
26 wire DAVE ;
27
28 //}} End of automatically maintained section
29
30 // -- Enter your statements here -- //
31 reg spi_side;
32 reg clk_side;
33
34 assign DAVE = spi_side ^ clk_side;
35
36 always @ (posedge reset or posedge clk)
37     begin
38         if (reset) clk_side <= 1'b0;
39         else if (reset_DAVE) clk_side <= spi_side;
40         else clk_side <= clk_side;
41     end
42
43 always @ (posedge reset or posedge spi_clk)
44     begin
45         if (reset) spi_side <= 1'b0;
46         else if (set_DAVE) spi_side <= !clk_side;
47         else spi_side <= spi_side;
48     end
49
50 endmodule
51
```

Module: Write_command



Sreg0

