"Gateway" lab exercises

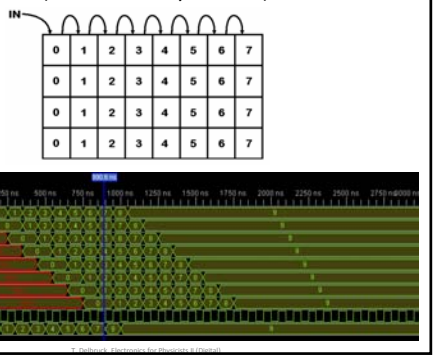
1. HelloWorld - Making a new project with a *module* and controlling a single LED with a button.
2. Hello lots of Worlds - making a *bus* to wire all switches to all LEDs; the UCF (User Constraints File).
3. HelloWorldSynchronous - using *registers* and *wires*, simulation with a verilog *test fixture*; the *sensitivity list* in always@ in simulation. "If within a module you have a signal that is on the left hand side of an assignment within an 'always@(…)' statement, then it needs to be defined as a register ('reg')".
4. ShiftingTheWorld - synthesizing a shift register with *fd* D-FlipFlops using gate level and behavioral level design; *register transfer level (RTL)* design; *module instantiation*; signal*concatenation*; introduction to *generate*.
5. ShiftingManyWorlds - 2d array of shift registers (memory); simulation exercise.
6. CountingWorlds - simple arithmetic, *multiplexing*.

"Gateway" lab exercises

4. ShiftingTheWorld - synthesizing a shift register with *fd* D-FlipFlops using gate level and behavioral level design; *register transfer level (RTL)* design; *module instantiation*; signal*concatenation*; introduction to *generate*.
5. ShiftingManyWorlds - 2d array of shift registers (memory); simulation exercise.

5. Shifting many worlds

(simulation-only exercise)



"Gateway" lab exercises

6. CountingWorlds - simple arithmetic, ***multiplexing***.
7. TimingTheWorld - a second-counter watch using two counters, one clocking the other, both up/down with enable.
8. DecodingTheWorld - Number representation; 7-segment display *decoder* (see BASYS2 manual). See 7seg for the code for this exercise.
9. TimingTheWorldInDecimel - multiple counters, using ***generics*** to instantiate modules with parameters; revisit *generate*.

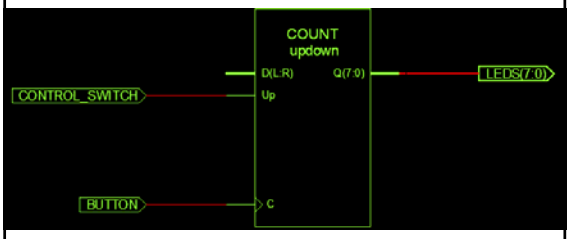
6. Counting the world

Arithmetic, multiplexing, +/-, if/else

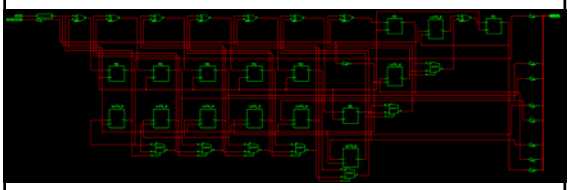
On the rising edge of the signal from the Button, **if** the control signal from the slide switch is '1' **then** the number stored in the register takes the value of the number stored in the register **plus** one. **Else** then the number stored in the register takes the value of the number stored in the register **minus** one.

```
21  module CountingWorlds(
22      input BUTTON,
23      input CONTROL_SWITCH,
24      output [7:0] LEDS
25      );
26
27    reg [7:0] Value;
28
29    always@(posedge BUTTON) beg
30       if (CONTROL_SWITCH)
31          Value <= Value + 1;
32       else
33          Value <= Value - 1;
34    end
35
36    assign LEDS = Value;
37
38  endmodule
39
```


RTL schematic



Technology schematic for counting up/down

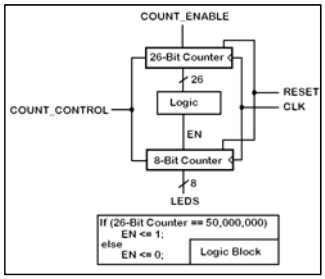


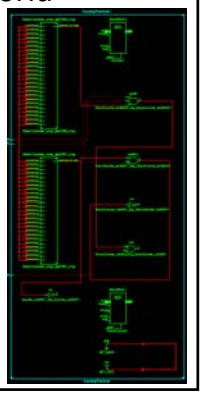
"Gateway" lab exercises

6. CountingWorlds - simple arithmetic, ***multiplexing***.
7. TimingTheWorld - a second-counter watch using two counters, one clocking the other, both up/down with enable.
8. DecodingTheWorld - Number representation; 7-segment display *decoder* (see BASYS2 manual). See 7seg for the code for this exercise.
9. TimingTheWorldInDecimel - multiple counters, using ***generics*** to instantiate modules with parameters; revisit *generate*.

7. Timing the world

- Using on-board clock (instead of switch)
- Multiplexing






```
`timescale 1ns / 1ps
module counter(
	input CLK,
	input UP,
	input RESET,
	input ENABLE,
	output [7:0] LEDS
	);
	reg [7:0]value;
	reg [24:0] downCounter;
	always@(posedge CLK) begin
		if(RESET)
			downCounter<=0;
		else begin
			if(ENABLE) begin
				if(UP) begin
					if(downCounter==25000000)
						downCounter<=0;
					else

	downCounter<=downCounter+1;

				end
				else begin
```



```
	always@(posedge CLK) begin
		if(RESET)
			downCounter<=0;
		else begin
			if(ENABLE) begin
				if(UP) begin
					if(downCounter==25000000)
						downCounter<=0;
					else

	downCounter<=downCounter+1;

				end
				else begin
					if(downCounter==0)
						downCounter<=25000000;
					else
						downCounter<=downCounter-
1;
				end
			end
		end
	end

	always@(posedge CLK) begin
```



```
	always@(posedge CLK) begin
		if(RESET)
			value<=0;
		else begin
			if(downCounter == 0) begin
				if(UP)
					value  <= value + 1;
				else
					value <= value - 1;
			end
		end

	end


	assign LEDS=value;
endmodule
```

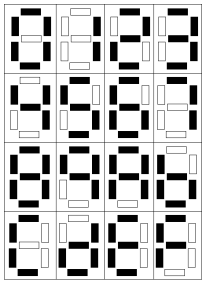
```
NET "UP" LOC="P11";  // SW0 controls up / down
NET "ENABLE" LOC=L3;  // SW1 must be up to count
NET "RESET" LOC=C11;  // BTN1 resets counter
NET "CLK" LOC=B8;              // global clock @ default
50MHz
NET "LEDS<7>" LOC="G1";
NET "LEDS<6>" LOC="P4";
NET "LEDS<5>" LOC="N4";
NET "LEDS<4>" LOC="N5";
NET "LEDS<3>" LOC="P6";
NET "LEDS<2>" LOC="P7";
NET "LEDS<1>" LOC="M11";
NET "LEDS<0>" LOC="M5";
```

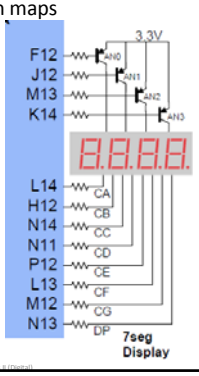

"Gateway" lab exercises

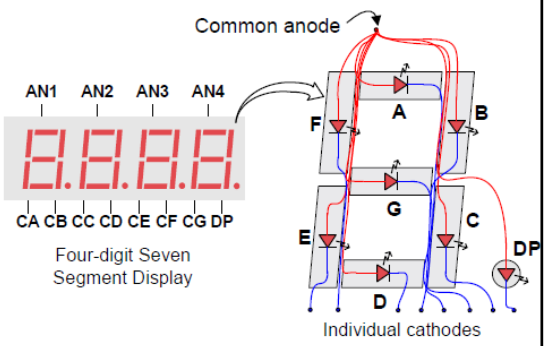
6. CountingWorlds - simple arithmetic, ***multiplexing***.
7. TimingTheWorld - a second-counter watch using two counters, one clocking the other, both up/down with enable.
8. DecodingTheWorld - Number representation; 7-segment display *decoder* (see BASYS2 manual). See 7seg for the code for this exercise.
9. TimingTheWorldInDecimel - multiple counters, using ***generics*** to instantiate modules with parameters; revisit *generate*.

8. Decoding the world

- Combinational logic, Karnaugh maps
- Module reuse with wrappers







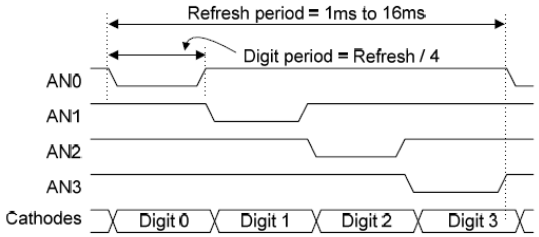
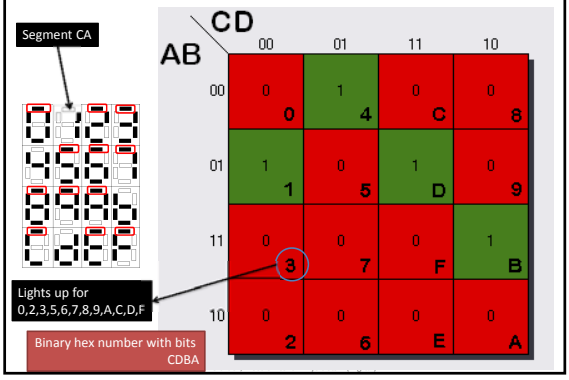


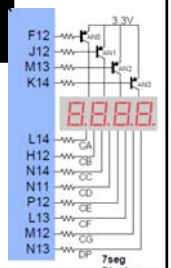
Figure 8. Multiplexed 7seg display timing

Karnaugh map for segment CA



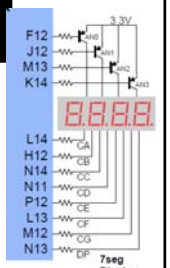
```
module DecodingTheWorld(
  input [1:0] SEG_SELECT_IN,
  output reg [3:0] SEG_SELECT_OUT,
  input [3:0] BIN_IN,
  output reg [7:0] HEX_OUT,
  input DOT_IN
  );
        always@(SEG_SELECT_IN) begin
                case(SEG_SELECT_IN)
                2'b00 : SEG_SELECT_OUT <= 4'b1110 ;
                2'b01 : SEG_SELECT_OUT <= 4'b1101 ;
                2'b10 : SEG_SELECT_OUT <= 4'b1011 ;
                2'b11 : SEG_SELECT_OUT <= 4'b0111 ;
                default: SEG_SELECT_OUT <= 4'b1111 ;
                endcase
        end
        always@(BIN_IN or DOT_IN) begin
                case(BIN_IN)
                4'h0 : HEX_OUT[6:0] <= 7'b1000000;
```

Demultiplexor to select one 7 segment display




```
        always@(BIN_IN or DOT_IN) begin
                case(BIN_IN)
                4'h0 : HEX_OUT[6:0] <= 7'b1000000;
                4'h1 : HEX_OUT[6:0] <= 7'b1001111;
                4'h2 : HEX_OUT[6:0] <= 
                4'h3 : HEX_OUT[6:0] <= 

                4'h4 : HEX_OUT[6:0] <= 
                4'h5 : HEX_OUT[6:0] <= 7'b0010010;
.....
                4'hF : HEX_OUT[6:0] <= 7'b0001110;
                default: HEX_OUT[6:0] <= 7'b1111111;
                endcase
                HEX_OUT[7] <= ~DOT_IN;
        end
endmodule
```

Demultiplexor to select one 7 segment display



8. Using a lookup table

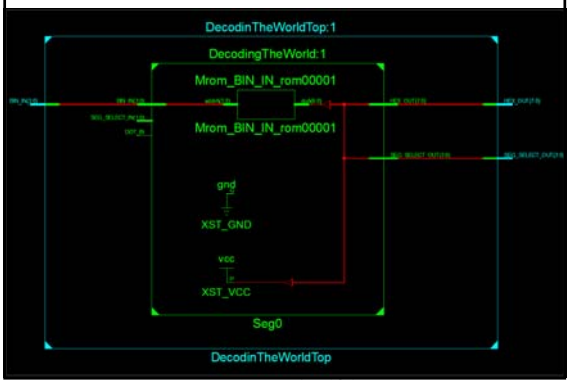
```
        always@(BIN_IN or DOT_IN) begin
                case(BIN_IN)
                4'h0 : HEX_OUT[6:0] <= 7'b1000000;
                4'h1 : HEX_OUT[6:0] <= 7'b1001111;
                4'h2 : HEX_OUT[6:0] <= 7'b0100100;
                4'h3 : HEX_OUT[6:0] <= 7'b0110000;

                4'h4 : HEX_OUT[6:0] <= 7'b0011001;
                4'h5 : HEX_OUT[6:0] <= 7'b0010010;
                4'h6 : HEX_OUT[6:0] <= 7'b0000010;
                4'h7 : HEX_OUT[6:0] <= 7'b1111000;
                default: HEX_OUT[6:0] <= 7'b1111111;
                endcase
                HEX_OUT[7] <= ~DOT_IN;
        end
endmodule
```

Patterns for each value

Decimel point

8.DecodingTheWorld schematic - interface

8.DecodingTheWorld technology schematic

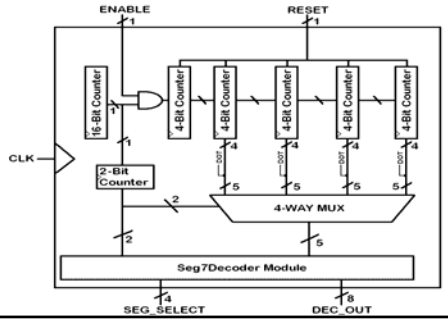


"Gateway" lab exercises

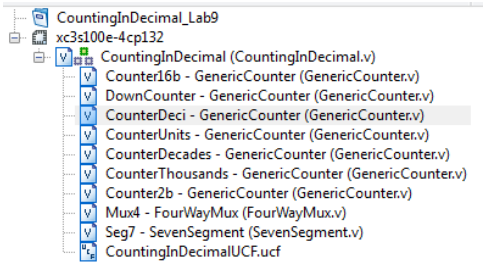
6. CountingWorlds - simple arithmetic, ***multiplexing***.
7. TimingTheWorld - a second-counter watch using two counters, one clocking the other, both up/down with enable.
8. DecodingTheWorld - Number representation; 7-segment display *decoder* (see BASYS2 manual). See 7seg for the code for this exercise.
9. TimingTheWorldInDecimel - multiple counters, using ***generics*** to instantiate modules with parameters; revisit *generate*.

9. TimingTheWorldInDecimel

A decimal up timer that displays its result in seconds on the 7-segment display block



9. TimingTheWorldInDecimel



GenericCounter

```
module GenericCounter(CLK,    RESET,     ENABLE_IN,     TRIG_OUT,
COUNT);
parameter COUNTER_WIDTH = 4;
parameter COUNTER_MAX = 9;

input CLK;
input RESET;
input ENABLE_IN;
output TRIG_OUT;
output [COUNTER_WIDTH-1:0] COUNT;

reg [COUNTER_WIDTH-1 : 0]  Counter;
reg TriggerOut;

always@(posedge CLK) begin
        if(RESET)
                Counter <= 0;
        else begin
                if(ENABLE_IN) begin
                        if(Counter == COUNTER_MAX)
                                Counter <= 0;
                        else
                                Counter <= Counter + 1;
```

GenericCounter

```
always@(posedge CLK) begin
        if(RESET)
                Counter <= 0;
        else begin
                if(ENABLE_IN) begin
                        if(Counter == COUNTER_MAX)
                                Counter <= 0;
                        else
                                Counter <= Counter + 1;
                end
        end
end
always@(posedge CLK) begin
        if(RESET)
                TriggerOut <= 0;
        else begin
                if(ENABLE_IN && Counter == COUNTER_MAX)
                        TriggerOut <= 1;
                else
                        TriggerOut <= 0;
        end
end
assign COUNT = Counter;
assign TRIG_OUT = TriggerOut;
endmodule
```

Generic Instantiation

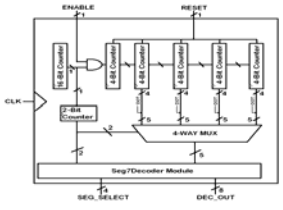
[Module Type] [Unique Module Name] (...[Interface List]...);

```
161     //Instantiate the 7 Segment Decoder
162     Seg7Decoder Seg7 (.SEG_SELECT_IN(StrobeCount),
163                       .BIN_IN(MuxOut[3:0]),
164                       .DOT_IN(MuxOut[4]),
165                       .SEG_SELECT_OUT(SEG_SELECT),
166                       .HEX_OUT(DEC_OUT)
167                       );
```

[Module Type] #(...[Parameter Definition List]...) [Unique Module Name] (...[Interface List]...);

```
54      //The 16 bit Counter
55      GenericCounter #(  .COUNTER_WIDTH(16),
56                         .COUNTER_MAX(49999)
57                         )
58                      Bit16  (
59                         .CLK(CLK),
60                         .RESET(1'b0),
61                         .ENABLE_IN(1'b1),
62                         .TRIGG_OUT(Bit16TrigOut),
63                         .COUNT(Bit16Count)
64                         );
```

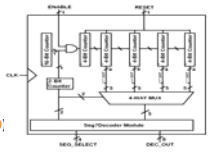
```
`timescale 1ns / 1ps
module CountingInDecimal(
  input CLK,
  input RESET,
  input ENABLE,
  output [3:0] SEG_SELECT,
  output [7:0] DEC_OUT
  );

wire TriggerOut_16b, DownCounterEnable, TriggerOut_DownCounter, TriggerOut_CounterDeci,
TriggerOut_CounterUnits, TriggerOut_CounterDecades, TriggerOut_CounterThousands;
wire [3:0] DownCount;
wire [3:0] CountDeci;
wire [3:0] CountUnits;
wire [3:0] CountDecades;
wire [3:0] CountThousands;
wire [1:0] Select;
wire [4:0] Bin_Dec;
wire [4:0] DownCounterOut;
wire [4:0] CountDeciOut;
wire [4:0] CountUnitsOut;
wire [4:0] CountDecadesOut;
wire [4:0] CountThousandsOut;
```



```
wire TriggerOut_16b, DownCounterEnable, TriggerOut_DownCounter, TriggerOut_CounterDeci,
TriggerOut_CounterUnits, TriggerOut_CounterDecades, TriggerOut_CounterThousands;
wire [3:0] DownCount;
wire [3:0] CountDeci;
wire [3:0] CountUnits;
wire [3:0] CountDecades;
wire [3:0] CountThousands;
wire [1:0] Select;
wire [4:0] Bin_Dec;
wire [4:0] DownCounterOut;
wire [4:0] CountDeciOut;
wire [4:0] CountUnitsOut;
wire [4:0] CountDecadesOut;
wire [4:0] CountThousandsOut;


GenericCounter  #(.COUNTER_WIDTH(16), .COUNTER_MAX(49999))
Counter16b(
                    .CLK(CLK),
                    .RESET(1'b0),
                    .ENABLE_IN(1'b1),
                    .TRIG_OUT(TriggerOut_16b)
);

assign DownCounterEnable = ENABLE && TriggerOut_16b;
```



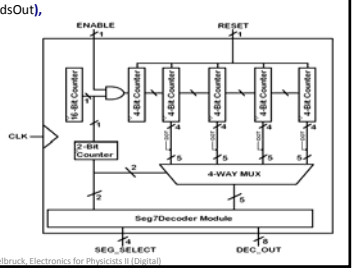
```
GenericCounter  #(.COUNTER_WIDTH(16), .COUNTER_MAX(49999))
Counter16b(
                    .CLK(CLK),
                    .RESET(1'b0),
                    .ENABLE_IN(1'b1),
                    .TRIG_OUT(TriggerOut_16b)
);

assign DownCounterEnable = ENABLE && TriggerOut_16b;

GenericCounter  #(.COUNTER_WIDTH(7), .COUNTER_MAX(99))
DownCounter(
                    .CLK(CLK),
                    .RESET(RESET),
                    .ENABLE_IN(DownCounterEnable),
                    .TRIG_OUT(TriggerOut_DownCounter),
                    .COUNT(DownCount)
);

GenericCounter  #(.COUNTER_WIDTH(4), .COUNTER_MAX(9
CounterDeci(
                    .CLK(CLK),
                    .RESET(RESET),
                    .ENABLE_IN(TriggerOut_DownCounter),
                    .TRIG_OUT(TriggerOut_CounterDeci),
```

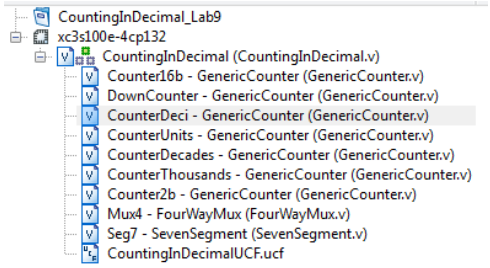



```
GenericCounter  #(.COUNTER_WIDTH(4), .COUNTER_MAX(9))
CounterDeci(
                    .CLK(CLK),
                    .RESET(RESET),
                    .ENABLE_IN(TriggerOut_DownCounter),
                    .TRIG_OUT(TriggerOut_CounterDeci),
                    .COUNT(CountDeci)
);


GenericCounter  #(.COUNTER_WIDTH(4), .COUNTER_MAX(9))
CounterUnits(
                    .CLK(CLK),
                    .RESET(RESET),
                    .ENABLE_IN(TriggerOut_CounterDeci),
                    .TRIG_OUT(TriggerOut_CounterUnits),
                    .COUNT(CountUnits)
);

GenericCounter  #(.COUNTER_WIDTH(4), .COUNTER_MAX(9))
CounterDecades(
                    .CLK(CLK),
                    .RESET(RESET),
                    .ENABLE_IN(TriggerOut_CounterUnits),
                    .TRIG_OUT(TriggerOut_CounterDecades),
```

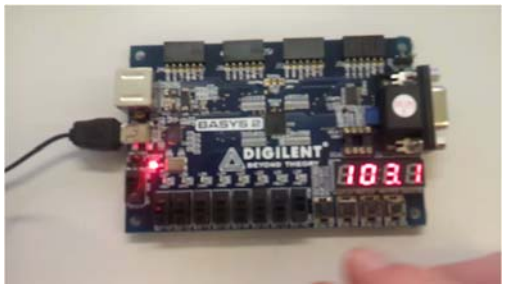


```
assign CountDeciOut = {1'b0,CountDeci};
assign CountUnitsOut = {1'b1,CountUnits};
assign CountDecadesOut = {1'b0,CountDecades};
assign CountThousandsOut = {1'b0,CountThousands};

FourWayMux Mux4(
   .CounterDeci(CountDeciOut),
   .CounterUnit(CountUnitsOut),
   .CounterDec(CountDecadesOut),
   .CounterThousand(CountThousandsOut),
   .Select(Select),
   .Bin_Out(Bin_Dec)
   );

SevenSegment Seg7(
   .SEG_SELECT_IN(Select),
   .SEG_SELECT_OUT(SEG_SELECT),
   .BIN_IN(Bin_Dec[3:0]),
   .HEX_OUT(DEC_OUT),
   .DOT_IN(Bin_Dec[4])
);
endmodule
```





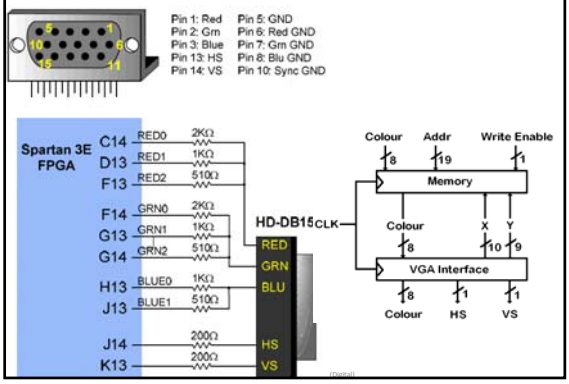
9. TimingTheWorldInDecimel



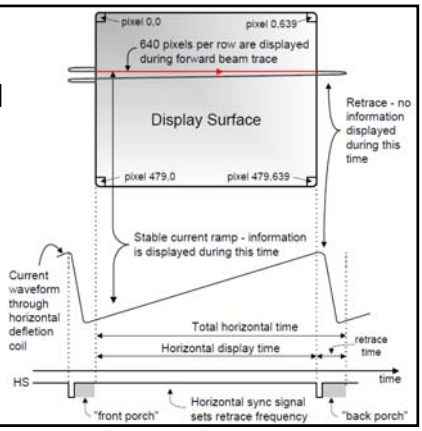
"Gateway" lab exercises

10. ColourTheWorld - *parameters* in generics; VGA display control to generate sync signals and RGB colors (see BASYS2 manual).
11. WorldOfStateMachines - making ***state machines*** using sequential and combinational blocks (switch/case statements) and using ROM modules (**$readmemb**).
12. WorldOfLinkedStateMachines - multiple state machines linked by a master state machine.
13. Snake - a complete snake game.

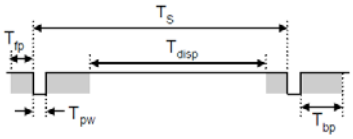
10. ColorTheWorld



VGA Control Signals

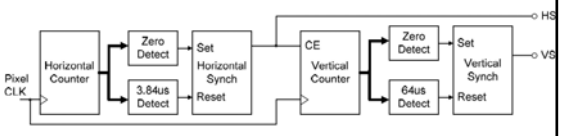


10. ColorTheWorld sync signals

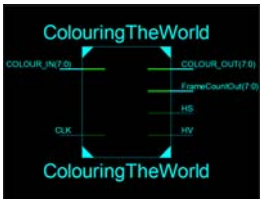


Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

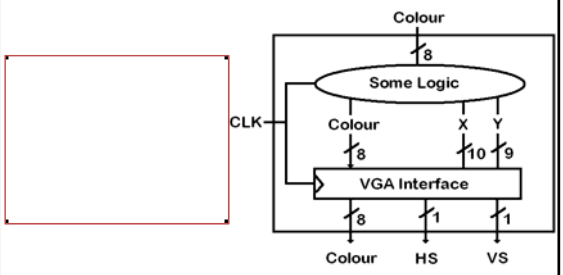
10. ColorTheWorld sync signals




```
21  module VGAControl(
22      input         CLK,
23      output  [9:0] ADDRH,
24      output  [8:0] ADDRV,
25      input   [7:0] CIN,
26      output  reg [7:0] COUT,
27      output        HS,
28      output        VS
29      );
30
31    //Time in Vertical Lines
32    parameter VertTimeToPulseWidthEnd      = 10'd2;
33    parameter VertTimeToBackPorchEnd       = 10'd31;
34    parameter VertTimeToDisplayTimeEnd     = 10'd511;
35    parameter VertTimeToFrontPorchEnd      = 10'd521;
36
37    //Time in Horizontal Lines
38    parameter HorzTimeToPulseWidthEnd      = 10'd96;
39    parameter HorzTimeToBackPorchEnd       = 10'd144;
40    parameter HorzTimeToDisplayTimeEnd     = 10'd784;
41    parameter HorzTimeToFrontPorchEnd      = 10'd800;
```



10. Wrapper module



"Gateway" lab exercises

10. ColourTheWorld - *parameters* in generics; VGA display control to generate sync signals and RGB colors (see BASYS2 manual).
11. WorldOfStateMachines - making ***state machines*** using sequential and combinational blocks (switch/case statements) and using ROM modules (**$readmemb**).
12. WorldOfLinkedStateMachines - multiple state machines linked by a master state machine.
13. Snake - a complete snake game.

11. WorldOfStateMachines

11. WorldOfStateMachines

11. WorldOfStateMachines

11. WorldOfStateMachines

"Gateway" lab exercises

10. ColourTheWorld - *parameters* in generics; VGA display control to generate sync signals and RGB colors (see BASYS2 manual).
11. WorldOfStateMachines - making ***state machines*** using sequential and combinational blocks (switch/case statements) and using ROM modules (**$readmemb**).
12. WorldOfLinkedStateMachines - multiple state machines linked by a master state machine.
13. Snake - a complete snake game.

12. WorldOfLinkedStateMachines

12. WorldOfLinkedStateMachines

12. WorldOfLinkedStateMachines

12. WorldOfLinkedStateMachines

"Gateway" lab exercises

10. ColourTheWorld - *parameters* in generics; VGA display control to generate sync signals and RGB colors (see BASYS2 manual).
11. WorldOfStateMachines - making ***state machines*** using sequential and combinational blocks (switch/case statements) and using ROM modules (**$readmemb**).
12. WorldOfLinkedStateMachines - multiple state machines linked by a master state machine.
13. Snake - a complete snake game.

12. Snake game

12. Snake game

12. Snake game