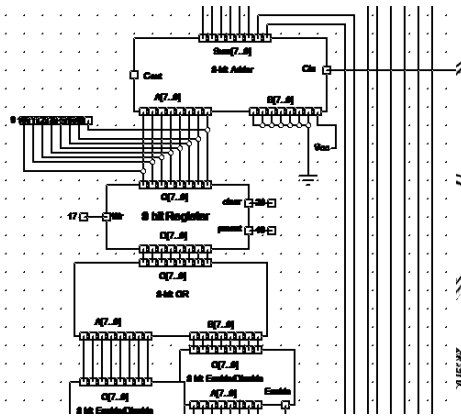
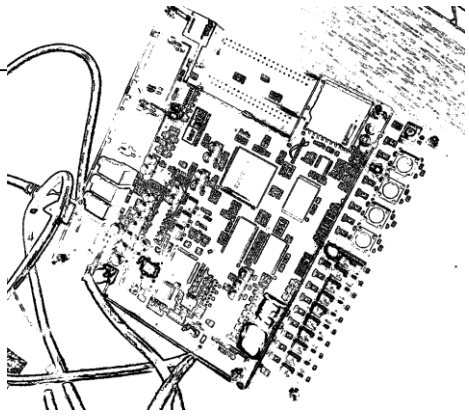
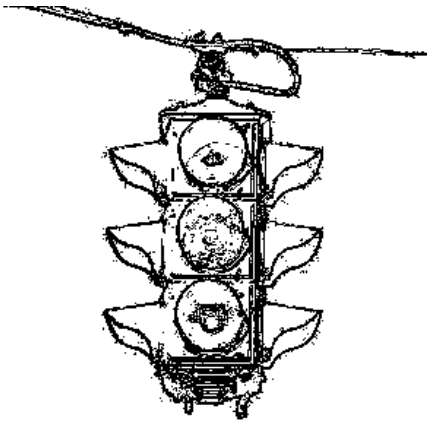
Project 1: Traffic Light Controller





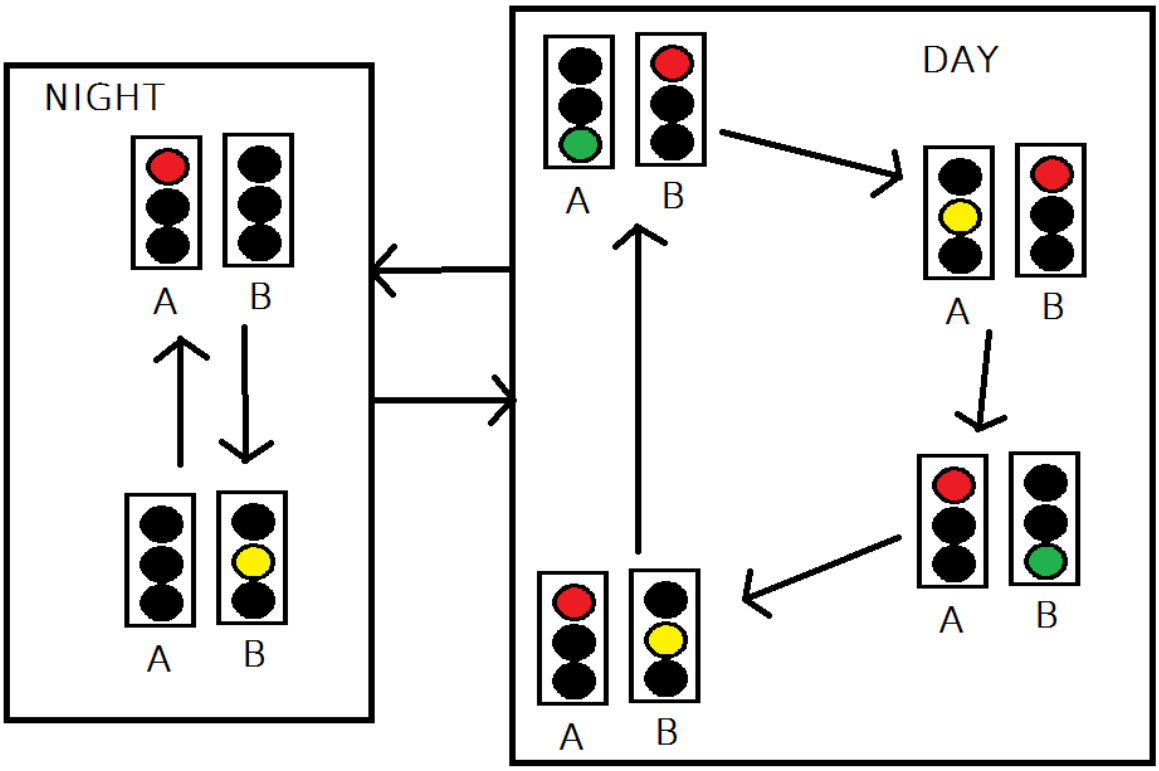


AIM: Build the traffic light controller which follows the rules of the states defined in the Assignment.

1) State Table:

state	timing state	day/night	A red	A yellow	A green	B red	B yellow	B green
0	0 to 4	DAY	0	0	1	1	0	0
1	5 to 9	DAY	0	0	1	1	0	0
2	10 to 59	DAY	0	0	1	1	0	0
3	60 to 64	DAY	0	1	0	1	0	0
4	65 to 124	DAY	1	0	0	0	0	1
5	125 to 129	DAY	1	0	0	0	1	0
6	0 to 4	NIGHT	1	0	0	0	0	0
7	5 to 9	NIGHT	0	0	0	0	1	0
8	> 9	NIGHT	UNDEF	UNDEF	UNDEF	UNDEF	UNDEF	UNDEF

2) State Diagram



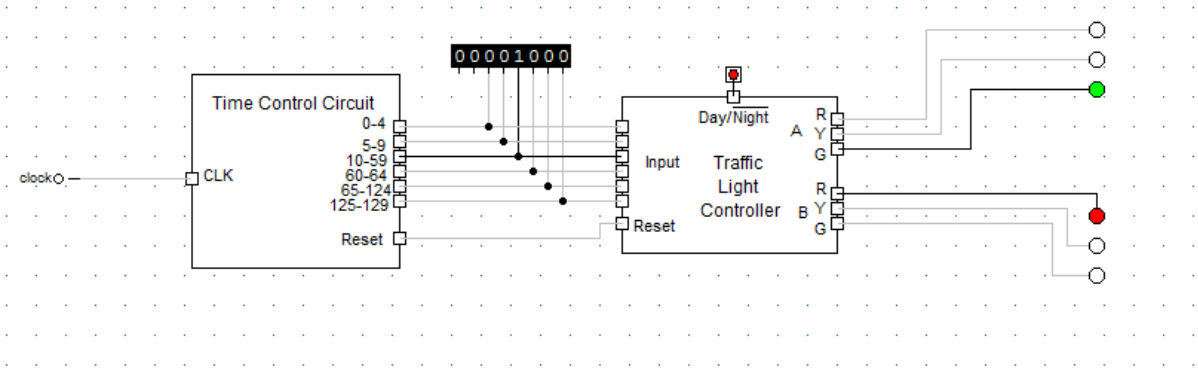
3) Comment on the complexity of using only sequential logic to implement:

Using only state shifters could require the setup and design of multiple sequential registers to implement the state machine, this problem can be better solved using a combination of sequential, combinational and MSI circuits.

4) Implementation:

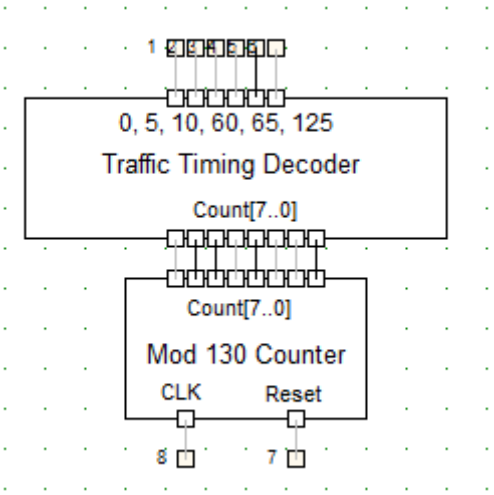
This project was implemented both in pure Diagram form in Digital Works and in pure HDL form using Quartus 2, the Altera DE1 board and ADHL.

A) Digital Works Version

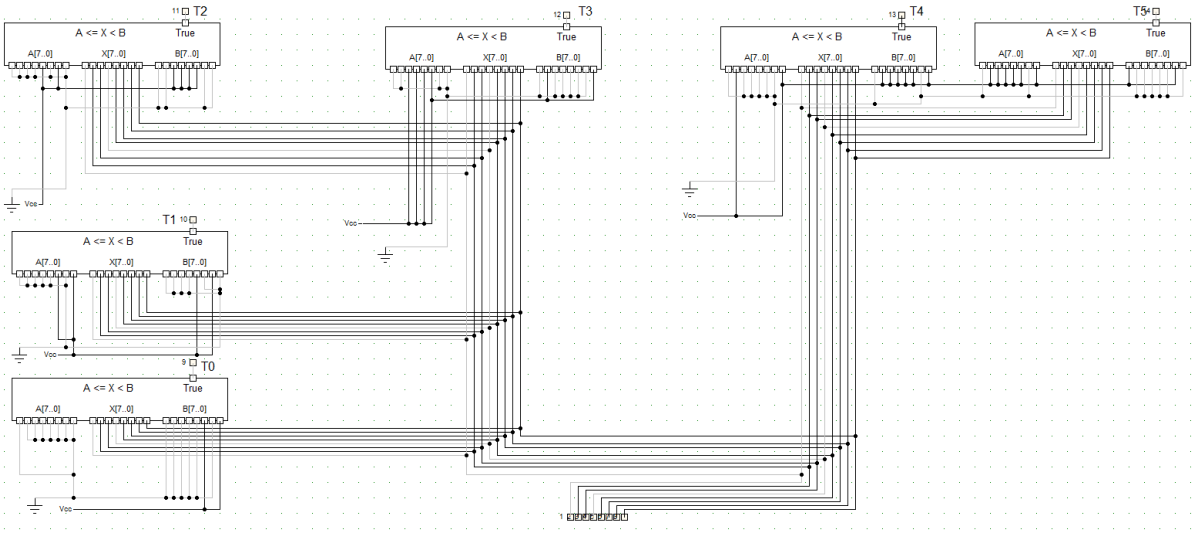


High level view of the controller, and timing circuit interacting.

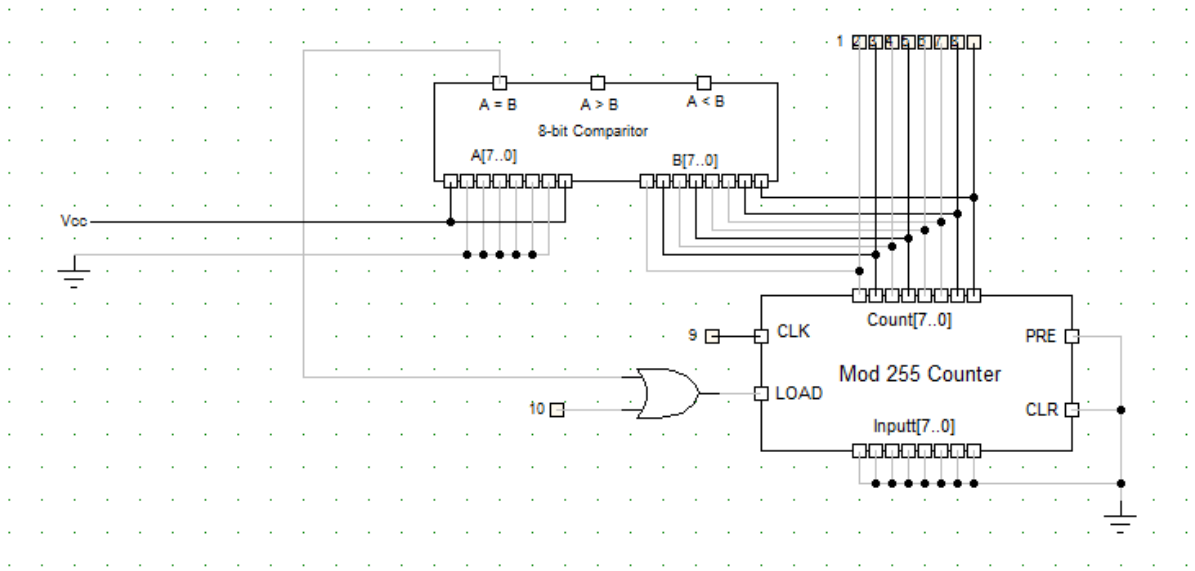
The timing control circuit in more detail:



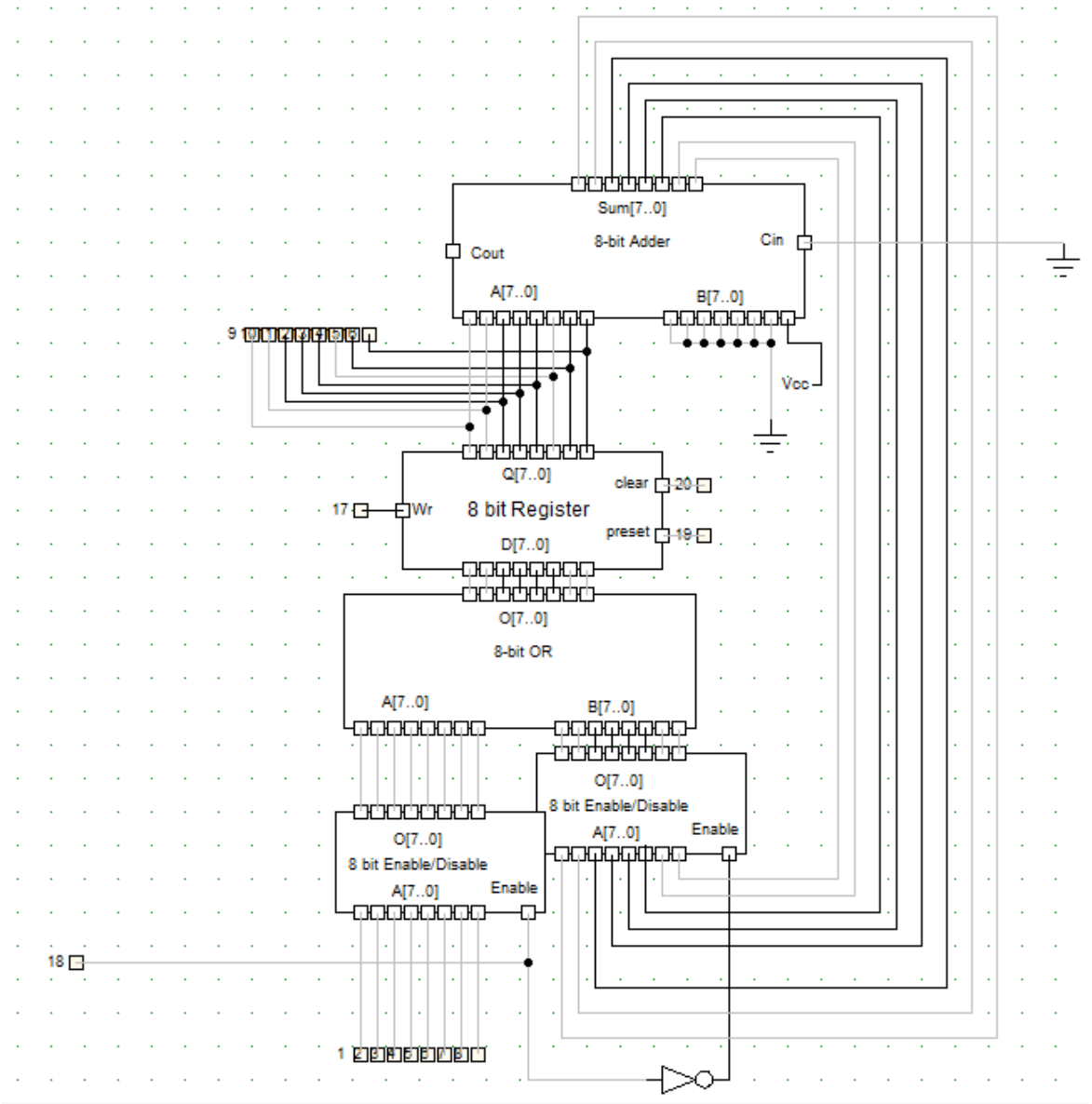
The mod-130 Counter & it's Decoder.



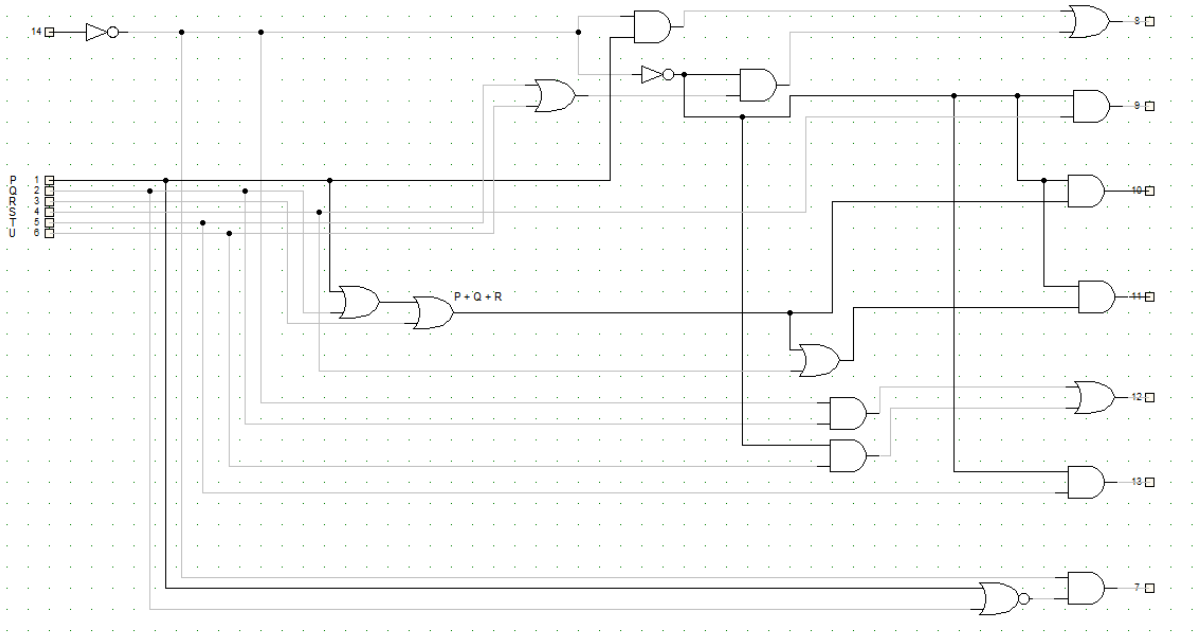
The decoder in more detail.



The mod-130 counter



The mod-256 counter.



The Traffic Light Controller.

B) The HDL Version:

The project was described in ADHL, and contains 7 modules. These are described on the following pages.

The high level Traffic Light Project:

```
INCLUDE "TIMINGCONTROLCIRCUIT";
INCLUDE "TRAFFICLIGHTCONTROLLER";
INCLUDE "SEVENSEGCODER";

SUBDESIGN TRAFFICLIGHTPROJECT(
        CLOCK_50 :INPUT;
        SW[0] :INPUT;
        LEDR[2..0] :OUTPUT;
        LEDG[2..0] :OUTPUT;
        HEX0[6..0] :OUTPUT;
        HEX1[6..0] :OUTPUT;
)
VARIABLE
        hd1 :SEVENSEGCODER;
        hd2 :SEVENSEGCODER;
        controller :TRAFFICLIGHTCONTROLLER;
        timer : TIMINGCONTROLCIRCUIT;
BEGIN

        timer.CLOCKINP = CLOCK_50;
        hd1.NUM[] = timer.OUTCOUNT[3..0];
        hd2.NUM[] = timer.OUTCOUNT[7..4];

        controller.STATE[] = timer.STATE[];
        controller.N = SW[0];
        timer.RESET = controller.RESET;

        HEX0[] = hd1.OUTV[];
        HEX1[] = hd2.OUTV[];
        LEDR[0] = controller.AG;
        LEDR[1] = controller.AY;
        LEDR[2] = controller.AR;

        LEDG[0] = controller.BG;
        LEDG[1] = controller.BY;
        LEDG[2] = controller.BR;


END;
```

The Traffic Light Controller

This is the controlling device which turns the state signal into the output.

```
SUBDESIGN TRAFFICLIGHTCONTROLLER(
        STATE[5..0]:INPUT;
        N :INPUT;
        AR, AY, AG, BR, BY, BG, RESET:OUTPUT;
)
VARIABLE
        P, Q, R, S, T, U :NODE;
        BEGIN
        P = STATE[0];
        Q = STATE[1];
        R = STATE[2];
        S = STATE[3];
        T = STATE[4];
        U = STATE[5];

        AR = (N & P) # (!N & (T # U));
        AY = !N & S;
        AG = !N & (P # Q # R);
        BR = !N & (P # Q # R # S);
        BY = (N & Q) # (!N & U);
        BG = !N & T;
        RESET = N & (!(P # Q));

END;
```

The Timing Controller decodes the time signal to a set of states.

```
INCLUDE "ONEHZCLOCK.inc";
INCLUDE "MOD130COUNTER.inc";

SUBDESIGN TIMINGCONTROLCIRCUIT(
CLOCKINP: INPUT;
RESET:INPUT;
OUTCOUNT[7..0] :OUTPUT;
STATE[5..0] :OUTPUT;
)
VARIABLE
CLOCKA :ONEHZCLOCK;
COUNTER :MOD130COUNTER;
BEGIN
CLOCKA.CLKIN50 = CLOCKINP;
COUNTER.CLK = CLOCKA.OUTCLK;
COUNTER.RESET = RESET;

OUTCOUNT[] = COUNTER.OUTNUM;

IF COUNTER.OUTNUM[] >= 0 AND COUNTER.OUTNUM[] < 5 THEN
        STATE[0] = VCC;
ELSE
        STATE[0] = GND;
END IF;

IF COUNTER.OUTNUM[] >= 5 AND COUNTER.OUTNUM[] < 10 THEN
        STATE[1] = VCC;
ELSE
        STATE[1] = GND;
END IF;

IF COUNTER.OUTNUM[] >= 10 AND COUNTER.OUTNUM[] < 60 THEN
        STATE[2] = VCC;
ELSE
        STATE[2] = GND;
END IF;

IF COUNTER.OUTNUM[] >= 60 AND COUNTER.OUTNUM[] < 65 THEN
        STATE[3] = VCC;
ELSE
        STATE[3] = GND;
END IF;

IF COUNTER.OUTNUM[] >= 65 AND COUNTER.OUTNUM[] < 125 THEN
        STATE[4] = VCC;
ELSE
        STATE[4] = GND;
END IF;

IF COUNTER.OUTNUM[] >= 125 AND COUNTER.OUTNUM[] <= 130 THEN
        STATE[5] = VCC;
ELSE
        STATE[5] = GND;
END IF;

END;
```

The Seven Segment Display Decoder Driver, simply converts a binary number to the display signal.

```
SUBDESIGN SEVENSEGCODER(
NUM[3..0]:INPUT;
OUTV[6..0]:OUTPUT;
)
BEGIN
	IF NUM[3..0] == 0 THEN
		OUTV[0..6] = (0,0,0,0,0,0,1);
	ELSIF NUM[3..0] == 1 THEN
		OUTV[0..6] = (1,0,0,1,1,1,1);
	ELSIF NUM[3..0] == 2 THEN
		OUTV[0..6] = (0,0,1,0,0,1,0);
	ELSIF NUM[3..0] == 3 THEN
		OUTV[0..6] = (0,0,0,0,1,1,0);
	ELSIF NUM[3..0] == 4 THEN
		OUTV[0..6] = (1,0,0,1,1,0,0);
	ELSIF NUM[3..0] == 5 THEN
		OUTV[0..6] = (0,1,0,0,1,0,0);
	ELSIF NUM[3..0] == 6 THEN
		OUTV[0..6] = (0,1,0,0,0,0,0);
	ELSIF NUM[3..0] == 7 THEN
		OUTV[0..6] = (0,0,0,1,1,1,1);
	ELSIF NUM[3..0] == 8 THEN
		OUTV[0..6] = (0,0,0,0,0,0,0);
	ELSIF NUM[3..0] == 9 THEN
		OUTV[0..6] = (0,0,0,0,1,0,0);
	ELSIF NUM[3..0] == 10 THEN
		OUTV[0..6] = (0,0,0,1,0,0,0);
	ELSIF NUM[3..0] == 11 THEN
		OUTV[0..6] = (1,1,0,0,0,0,0);
	ELSIF NUM[3..0] == 12 THEN
		OUTV[0..6] = (0,1,1,0,0,0,1);
	ELSIF NUM[3..0] == 13 THEN
		OUTV[0..6] = (1,0,0,0,0,1,0);
	ELSIF NUM[3..0] == 14 THEN
		OUTV[0..6] = (0,1,1,0,0,0,0);
	ELSIF NUM[3..0] == 15 THEN
		OUTV[0..6] = (0,1,1,1,0,0,0);
	ELSE
		OUTV[6..0] = (1,1,1,1,1,1,1);
	END IF;


END;
```

The One Hz Clock, converts the signal from 50MHz to 1Hz.

```
SUBDESIGN ONEHZCLOCK(
CLKIN50 :INPUT;
OUTCLK :OUTPUT;
)
VARIABLE
ff1[25..0] :JKFF;
BEGIN
	ff1[].j = VCC;
	ff1[].K = VCC;
	ff1[0].clk = !CLKIN50;
	ff1[1].clk = ff1[0].q;
	ff1[2].clk = ff1[1].q;
	ff1[3].clk = ff1[2].q;
	ff1[4].clk = ff1[3].q;
	ff1[5].clk = ff1[4].q;
	ff1[6].clk = ff1[5].q;
	ff1[7].clk = ff1[6].q;
	ff1[8].clk = ff1[7].q;
	ff1[9].clk = ff1[8].q;
	ff1[10].clk = ff1[9].q;
	ff1[11].clk = ff1[10].q;
	ff1[12].clk = ff1[11].q;
	ff1[13].clk = ff1[12].q;
	ff1[14].clk = ff1[13].q;
	ff1[15].clk = ff1[14].q;
	ff1[16].clk = ff1[15].q;
	ff1[17].clk = ff1[16].q;
	ff1[18].clk = ff1[17].q;
	ff1[19].clk = ff1[18].q;
	ff1[20].clk = ff1[19].q;
	ff1[21].clk = ff1[20].q;
	ff1[22].clk = ff1[21].q;
	ff1[23].clk = ff1[22].q;
	ff1[24].clk = ff1[23].q;
	ff1[25].clk = ff1[24].q;


	OUTCLK = !ff1[24].q;

END;
```

The mod-130 counter, counts to 129,

```
INCLUDE "EIGHTBITREGISTER";

SUBDESIGN MOD130COUNTER(
RESET, CLK :INPUT;
OUTNUM[7..0] :OUTPUT;
)
VARIABLE
co : EIGHTBITREGISTER;
BEGIN
co.CLK = CLK;
IF co.OUTN[7..0] == 130 OR RESET THEN
        co.CLR = VCC;
ELSE
        co.CLR = GND;
END IF;

OUTNUM[] = co.OUTN[7..0];
END;
```

Eight Bit Register, Stores and counts 8 bits.

```
SUBDESIGN EIGHTBITREGISTER(
CLK, CLR :INPUT;
OUTN[7..0] :OUTPUT;
)
VARIABLE
Inter[7..0] :JKFF;
abc[7..0] :NODE;
BEGIN
Inter[].clk = CLK;
Inter[].clrn = !CLR;

abc[] = Inter[].q + 1;
Inter[].j = abc[];
Inter[].k = !abc[];

OUTN[] = Inter[].q;

END;
```