University of New South Wales

School of Electrical Engineering and Telecommunications

INDIVIDUAL Assessment Task: Assignment 1

Course Code	ELEC 2141	Course Name	
Week/Session/Year	T1 2020	Lecturer	

Student Number	z5206032
Family Name	Nguyen
Given Names	Dan

Mark/Grade given (For official use only)

Marker's Comments:

Since this work counts toward your formal assessment for this course, please write your name and student number where indicated above, and sign the declaration below. Attach this cover sheet to the front of your submission, so that your name and student number can be seen without any cover needing to be opened.

For further information, please see http://www.lc.unsw.edu.au/plagiarism/index.html

---

I declare that this assessment item is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere, and acknowledge that the assessor of this item may, for the purpose of assessing this item:

- ***Reproduce this assessment item and provide a copy to another member of the University ; and/or,***
- ***Communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the assessment item on its database for the purpose of future plagiarism checking).***

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct.

Signature of student _____ Date: 5/4/20

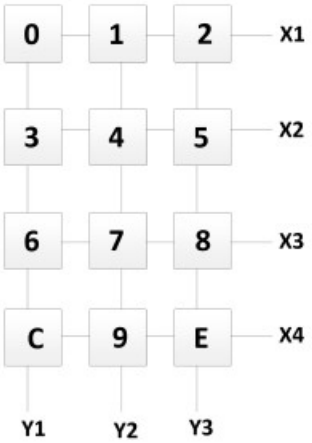
Specification

Friday, 13 March, 2020 1:37 AM

Specification:

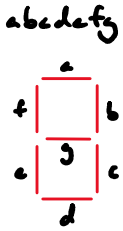
- A keypad with 12 buttons is wired in a grid pattern for 4 rows (X1, X2, X3, X4) and 3 columns (Y1, Y2, Y3). A button press will send a HIGH signal to the wires corresponding to the coordinate of the button.

  e.g. Pressing button 3 will send a HIGH signal to wires X2 and Y1 and nothing else.



- The keypad requires the design of a circuit to decode each of the 12 buttons that have a unique number/letter to be displayed in a 7-segment display. The required outputs are therefore a, b, c, d, e, f, g.

  e.g. Pressing button 3 will display the number 3 on a 7-segment display by turning on the LED segments: a, b, c, d, g.



- It is assumed that the 7-segment display is a common-cathode display (i.e. Turns on when HIGH).
- An invalid input is assumed that it is mechanically impossible to press any two buttons at the same time. Therefore if there is at least 2 HIGHs from any row or 2 HIGHS from any column at the same time, this will be considered a DON'T CARE condition.
- An invalid input constitutes an impossible scenario that is a single HIGH from any row but no HIGH from any column, and a single HIGH from any column but no HIGH from any row. This is a DON'T CARE CONDITION as this scenario is undefined.
- A valid input occurs if there is only 1 HIGH from all rows AND only 1 HIGH from all columns at the same time.
- A valid input occurs when no button is pressed, the 7-segment display does not display anything.

Design Procedure:

1. Get the truth table for the 7 inputs (X1, X2, X3, X4, Y1, Y2, Y3) and 7 outputs (a, b, c, d, e, f, g).
2. Draw 7-variable K-maps for each output (there are 7 outputs).
3. Get the prime implicants and essential prime implicants for the product of maxterms.
4. Write the product of maxterms from the prime implicants.
5. Reduce the GIC by factorisation/decomposition of the product of maxterms then write the GIC.
6. Implement the logic circuit diagrams in a Xilinx schematic.
7. Write a verilog test fixture.
8. Simulate.

Formulation

Wednesday, 18 March 2020 4:11 PM

Input							Output							Key
X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	a	b	c	d	e	f	g	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	1	0	0	1	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0	1	1	0	0	0	0	1
1	0	0	0	0	0	1	1	1	0	1	1	0	1	2
0	1	0	0	1	0	0	1	1	1	1	0	0	1	3
0	1	0	0	0	1	0	0	1	1	0	0	1	1	4
0	1	0	0	0	0	1	1	0	1	1	0	1	1	5
0	0	1	0	1	0	0	1	0	1	1	1	1	1	6
0	0	1	0	0	1	0	1	1	1	0	0	0	0	7
0	0	1	0	0	0	1	1	1	1	1	1	1	1	8
0	0	0	1	1	0	0	1	0	0	1	1	1	0	C
0	0	0	1	0	1	0	1	1	1	1	0	1	1	9
0	0	0	1	0	0	1	1	0	0	1	1	1	1	E

Figure 1: 7-Segment Display Truth Table

Every combination of inputs not listed in the truth table in figure 1 are don't care conditions.

Optimisation

Wednesday, 18 March 2020 4:52 PM

Considering the truth table from figure 1, 7 variable K-maps for each output were obtained.

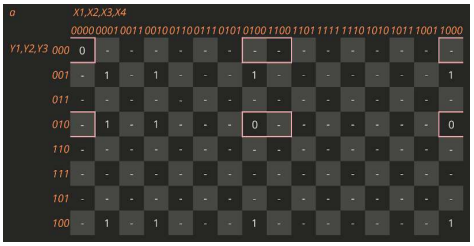


Figure 2.1: Segment a

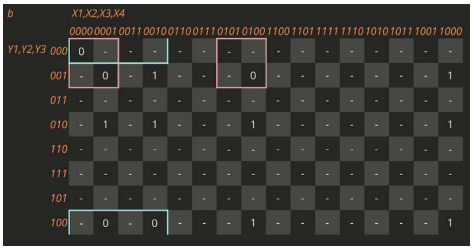


Figure 2.2: Segment b

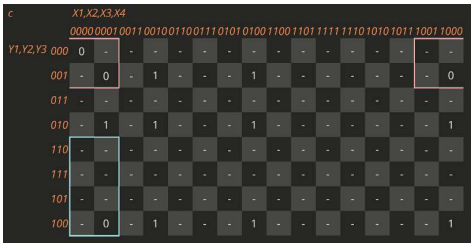


Figure 2.3: Segment c

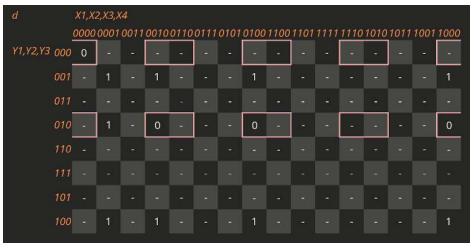


Figure 2.4: Segment d

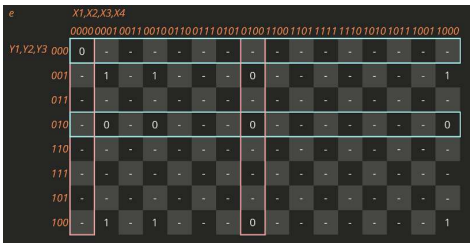


Figure 2.5: Segment e

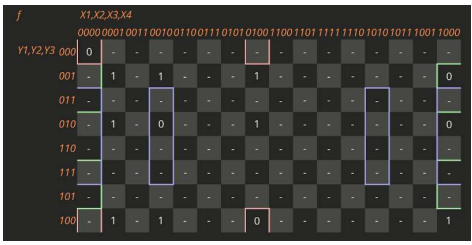


Figure 2.6: Segment f

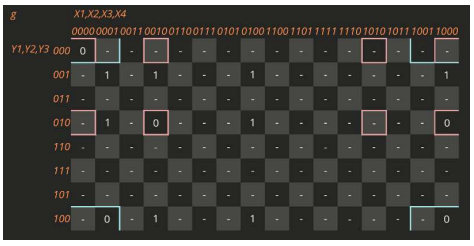


Figure 2.7: Segment g

Prime Implicant: Implicant that is not a subset of any other implicant.
Essential Prime Implicant: Implicant where at least one of its elements is not a subset of another implicant.

The prime implicants of all the K-maps are also the essential prime implicants which are:

Figure 2.1 / a: (Y1 + Y3 + X3 + X4)
Figure 2.2 / b: (Y1 + Y2 + X1 + X3), (Y2 + Y3 + X1 + X2)
Figure 2.3 / c: (Y1 + Y2 + X2 + X3), (Y1' + X1 + X2 + X3)
Figure 2.4 / d: (Y1 + Y3 + X4)
Figure 2.5 / e: (X1 + X3 + X4), (Y1 + Y3)
Figure 2.6 / f: (Y2 + Y3 + X1 + X3 + X4), (Y3' + X2 + X3 + X4), (Y2' + X2 + X4)
Figure 2.7 / g: (Y1 + Y3 + X2 + X4), (Y2 + Y3 + X2 + X3)

The product of maxterm expressions are therefore:

a = (Y1 + Y3 + X3 + X4) b = (Y1 + Y2 + X1 + X3)(Y2 + Y3 + X1 + X2) c = (Y1 + Y2 + X2 + X3)(Y1' + X1 + X2 + X3) d = (Y1 + Y3 + X4) e = (X1 + X3 + X4)(Y1 + Y3) f = (Y2 + Y3 + X1 + X3 + X4)(Y3' + X2 + X3 + X4)(Y2' + X2 + X4) g = (Y1 + Y3 + X2 + X4)(Y2 + Y3 + X2 + X3)	(1)

Equation set 1 can be decomposed:
Let D1 = (Y1 + Y3 + X4)
Let D2 = (Y1 + Y2 + X3)
Let D3 = (Y2 + Y3 + X2)
Let D4 = (X1 + X3 + X4)

The GIC of the decomposed set of D expressions is 12.

a = (X3 + D1) b = (X1 + D2)(X1 + D3) c = (X2 + D2)(Y1' + X1 + X2 + X3) d = (D1) e = (D4)(Y1 + Y3) f = (Y2 + Y3 +D4)(Y3' + X2 + X3 + X4)(Y2' + X2 + X4) g = (X2 + D1)(X3 + D3)	(2)

The GIC of equation set 2 is therefore 39.

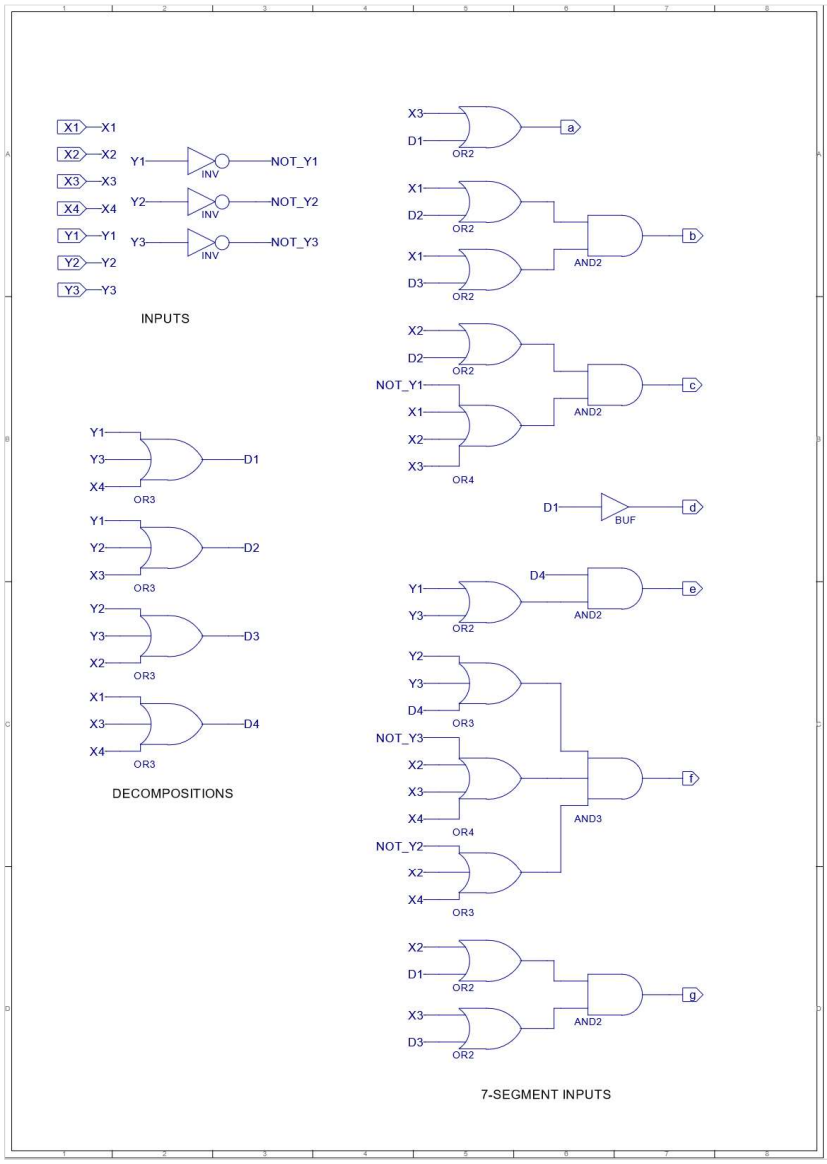
The number of complemented terms is 3.

The total GIC is therefore 12 + 39 + 3 = 54.

Circuit Implementation

Wednesday, 18 March 2020 4:53 PM

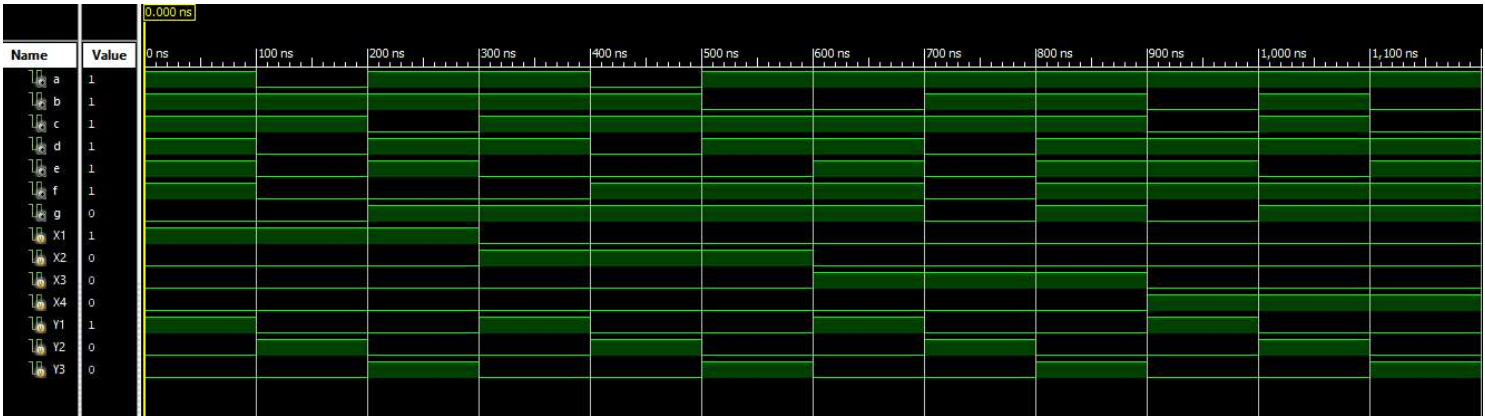
The final circuit is implemented using basic logic gates from POS expressions of equation set 2. Buffers in the circuit are ignored in the GIC count as Xilinx requires a buffer to distinguish two I/O markers to the same node.



Verification

Wednesday, 18 March 2020 4:11 PM

```
// Verilog test fixture created from schematic C:\Users\Dan\Documents\ELEC2141\ass1\ass1_q1\ass1
_q1.sch - Sun Apr 05 02:37:58 2020

`timescale 1ns / 1ps

module ass1_q1_ass1_q1_sch_tb();

// Inputs
reg X1;
reg X2;
reg X3;
reg X4;
reg Y1;
reg Y2;
reg Y3;

// Output
wire a;
wire b;
wire c;
wire d;
wire e;
wire f;
wire g;

// Instantiate the UUT
ass1_q1 UUT (
	.X1(X1),
	.X2(X2),
	.X3(X3),
	.X4(X4),
	.Y1(Y1),
	.Y2(Y2),
	.Y3(Y3),
	.a(a),
	.b(b),
	.c(c),
	.d(d),
	.e(e),
	.f(f),
	.g(g)
);

task cycle_y;
	begin
		{Y1} = {Y1} + 1;
		#100
		{Y1} = {Y1} + 1;
		{Y2} = {Y2} + 1;
		#100
		{Y2} = {Y2} + 1;
		{Y3} = {Y3} + 1;
		#100
		{Y3} = {Y3} + 1;
	end
endtask

// Initialize Inputs
initial begin
	X1 = 0;
	X2 = 0;
	X3 = 0;
	X4 = 0;
	Y1 = 0;
	Y2 = 0;
	Y3 = 0;
	forever begin
		X1 = 1; cycle_y(); X1 = 0;
		X2 = 1; cycle_y(); X2 = 0;
		X3 = 1; cycle_y(); X3 = 0;
		X4 = 1; cycle_y(); X4 = 0;
	end
end

endmodule
```



Specification

Friday, 13 March, 2020 1:38 AM

Specification:

- A water irrigation system has the following dependencies:
  - CLOCK (Clk)
    - CLOCK = 1 is HIGH
    - CLOCK = 0 is LOW
  - TSWITCH (TSW)
    - TSW = 1 is HIGH
    - TSW = 0 is not LOW
  - SALINE (SAL)
    - SAL = 1 is salty
    - SAL = 0 is not salty
  - DRY (DRY)
    - DRY = 1 is dry
    - DRY = 0 is not dry
  - RAIN (RAIN)
    - RAIN = 1 is raining
    - RAIN = 0 is not raining
  - HUMIDITY (HUM1, HUM2)
    For HUM1 is the MSB and HUM2 is the LSB
    - HUMIDITY = 00 is very dry
    - HUMIDITY = 01 is dry
    - HUMIDITY = 10 is humid
    - HUMIDITY = 11 is very humid

- The system requires two operating modes standby and pumping. Where on standby, water is not flowing; and while pumping, water is flowing. Let PUMP represent the state of the system where PUMP = 0 is standby and PUMP = 1 is pumping.
  - PUMP = 0 if RAIN = 1
  - PUMP = 1 if CLOCK or TSWITCH = 1

- The system requires the implementation of a 7-segment display to indicate the operating mode of the system. Let this display be DISPLAY1 which displays S for standby (PUMP = 0) and P for pumping (PUMP = 1).

- The system requires variable water flow (FLOW) that is dependent on SALINE, DRY, RAIN, and HUMIDITY.
  - Assume that when PUMP = 0, FLOW = 000 and FLOW is otherwise when PUMP = 1.
  - FLOW (FLOW1, FLOW2, FLOW3)
    Where FLOW1 is the MSB and FLOW3 is the LSB
    - FLOW = 000 is no flow
    - FLOW = 001 is very low flow
    - FLOW = 010 is low flow
    - FLOW = 011 is normal flow
    - FLOW = 100 is high flow
    - FLOW = 101 is very high flow
  - The conditions for water flow are:
    - No flow when PUMP = 0
    - Very low flow when DRY = 0, SAL = 0, HUM = 11
    - Low flow when DRY = 0, SAL = 0, HUM = 10
    - Low flow when DRY = 0, SAL = 1
    - Normal flow when DRY = 0, SAL = 0, HUM = 01
    - Normal flow when DRY = 1, SAL = 0
    - High flow when DRY = 0, SAL = 0, HUM = 00
    - Very high flow when DRY = 1, SAL = 1

- The system requires the implementation of a second 7-segment display to indicate the water flow in decimal. Let this display be DISPLAY2 which displays decimal numbers from 0 to 5 inclusive.

  e.g. FLOW = 010 displays a binary-converted decimal number 2 on DISPLAY2.

  The converted binary FLOW as decimal are:
  - FLOW = 000 = 0
  - FLOW = 001 = 1
  - FLOW = 010 = 2
  - FLOW = 011 = 3
  - FLOW = 100 = 4
  - FLOW = 101 = 5

- Assume all 7-segment displays are common-cathode displays (i.e. Turns on when HIGH).

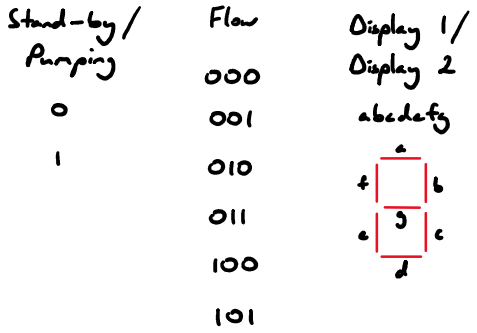
- Assume that there is no case for a blank 7-segment display

Design Procedure:

1. Get the truth table for the 3 inputs: Clk, TSW, RAIN and output: PUMP.
2. Get the truth table for the 4 inputs: SAL, DRY, HUM and PUMP as an enable; and outputs: FLOW, DISPLAY1, DISPLAY2.
3. Draw the 3-variable K-map for PUMP from the truth table in step 1.

Possible binary values of inputs are:

Possible binary values of outputs are:



4. Draw 4-variable K-maps (exclude PUMP) for each FLOW bit from the truth table in step 2.
5. Draw 7-variable K-maps for each DISPLAY2 segment.
6. Get the boolean expression by inspection from DISPLAY1.
7. Get the prime implicants and essential prime implicants for the product of maxterms from each K-map.
8. Write the product of maxterms from the prime implicants.
9. Reduce the GIC by factorisation/decomposition of the product of maxterms then write the GIC.
10. Implement the logic circuit diagrams in a Xilinx schematic.
11. Write a verilog test fixture.
12. Simulate.

Formulation

Thursday, 19 March 2020 12:02 AM

Input			Output
Rain	T	Clk	Pump
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	X	X	0

Figure 1: Pump Truth Table

Input				Output		
Pump	Saline	Dry	Humidity	Flow	Display 1	Display 2
0	X	X	X	000	1011011	1111110
1	0	0	00	100	1100111	0110011
1	0	0	01	011	1100111	1111001
1	0	0	10	010	1100111	1101101
1	0	0	11	001	1100111	0110000
1	0	1	X	011	1100111	1111001
1	1	0	X	010	1100111	1101101
1	1	1	X	101	1100111	1011011

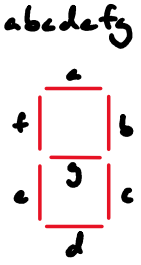


Figure 2: Flow and Display Truth Table

Optimisation

Thursday, 19 March 2020 12:04 AM

Considering the truth table from figure 1, a 3 variable K-map is obtained where the prime implicant maxterms are considered:

Figure 3: Pump K-Map

From figure 3, the prime implicants which are also the essential prime implicants are:

(T + Clk), (R')

Therefore the product of maxterm expression is:

P = (T + Clk)(R')	(1)

The GIC of expression P is 5.

For P = 1 and excluding P: K-maps for each bit of Flow (F1, F2, F3) are obtained from figure 2 where the prime implicant maxterms are considered:

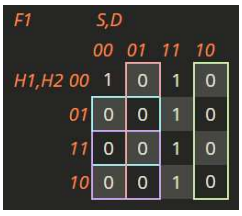


Figure 4.1: Flow Bit 1

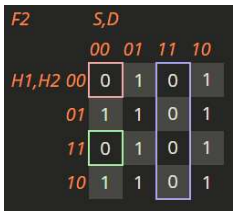


Figure 4.2: Flow Bit 2

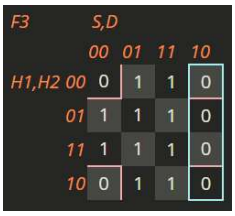


Figure 4.3: Flow Bit 3

The prime implicants for figures 4.1, 4.2 and 4.3 are also the essential prime implicants. The prime implicants are:

Figure 4.1 / F1: (S + D'), (S' + D), (H2' + S), (H1' + S)

Figure 4.2 / F2: (H1 + H2 + S + D), (H1' + H2' + S + D), (S' + D')

Figure 4.3 / F3: (H2 + D), (S' + D)

Note the maxterm (S' + D) is common between figure 4.1 and 4.3.

The product of maxterm expressions are obtained for each respective k-map with P as an enable.

F1 = (P)(S + D')(S' + D)(H2' + S)(H1' + S) F2 = (P)(H1 + H2 + S + D)(H1' + H2' + S + D)(S' + D') F3 = (P)(H2 + D)(S' + D)	(2)

The GIC is calculated later.

Consider the display 1 output from figure 2:

By inspection of the display 1 output:
- a, f, g are always 1
- b, c, d, e are complemented for keys S and P

a	b	c	d	e	f	g	Display 1	Key
1	0	1	1	0	1	1	1011011	S
1	1	0	0	1	1	1	1100111	P

Therefore:

a = 1 b = P c = P' d = P' e = P f = 1 g = 1	(3)

Therefore GIC of D1 is 1.

For the display 2 output from figure 2, a 5 variable K-map for each D2 LED segment is obtained where the prime implicant maxterms are considered:

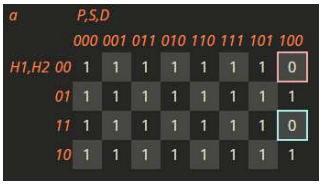


Figure 5.1: D2 Segment a

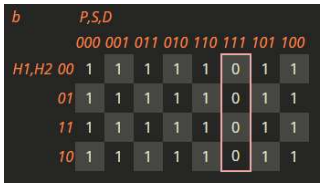


Figure 5.2: D2 Segment b

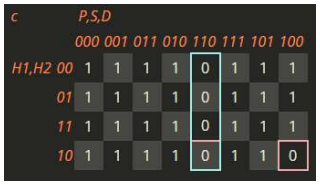


Figure 5.3: D2 Segment c

Figure 5.4: D2 Segment d

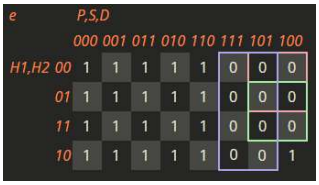


Figure 5.5: D2 Segment e

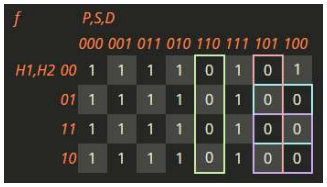


Figure 5.6: D2 Segment f

Figure 5.7: D2 Segment g

The prime implicants for all figures 5 are also the essential prime implicants which are:

Figure 5.1 / a: (H1 + H2 + P' + S + D), (H1' + H2' + P' + S + D)
Figure 5.2 / b: (P' + S' + D')
Figure 5.3 / c: (P' + S' + D), (H1' + H2 + P' + D)
Figure 5.4 / d: (H1 + H2 + P' + S + D), (H1' + H2' + P' + S + D)
Figure 5.5 / e: (H1 + P' + S), (P' + D'), (H2' + P' + S)
Figure 5.6 / f: (P' + S + D'), (P' + S' + D), (H2' + P' + S), (H1' + P' + S)
Figure 5.7 / g: (P), (H1' + H2' + S + D)

Note the common maxterms are:

Figure 5.1 and 5.4: (H1 + H2 + P' + S + D), (H1' + H2' + P' + S + D)
Figure 5.3 and 5.6: (P' + S' + D')
Figure 5.1, 5.4 and 5.7: (H1' + H2' + P' + S + D)
Figure 5.5 and 5.6: (H2' + P' + S)

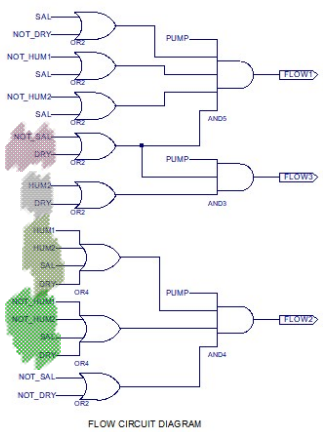
The product of maxterm expressions are obtained for each respective k-map:

a = (H1 + H2 + P' + S + D)(H1' + H2' + P' + S + D) b = (P' + S' + D') c = (P' + S' + D)(H1' + H2 + P' + D) d = (H1 + H2 + P' + S + D)(H1' + H2' + P' + S + D) e = (H1 + P' + S)(P' + D')(H2' + P' + S) f = (P' + S + D')(P' + S' + D)(H2' + P' + S)(H1' + P' + S) g = (P)(H1' + H2' + S + D)	(4)

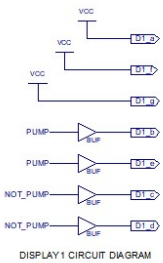
The GIC is calculated below.

a	b	c	d	e	f	g	Display 2	Key
1	1	1	1	1	1	0	1111110	0
0	1	1	0	0	0	0	0110000	1
1	1	0	1	1	0	1	1101101	2
1	1	1	1	0	0	1	1111001	3
0	1	1	0	0	1	1	0110011	4
1	0	1	1	0	1	1	1011011	5

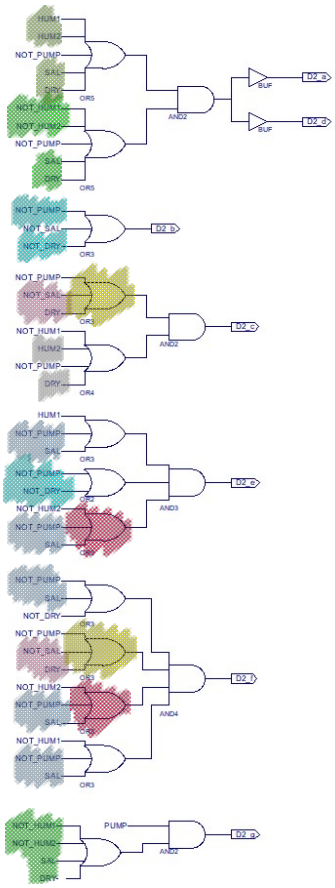
Common terms were observed in equation sets 2 and 4. Therefore these boolean expressions can be further optimised by decomposition. A visual method of decomposition was performed after drawing the logic circuits for equation sets 2 and 4.

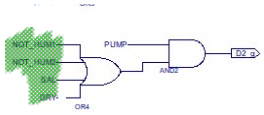


FLOW CIRCUIT DIAGRAM



DISPLAY 1 CIRCUIT DIAGRAM





DISPLAY 2 CIRCUIT DIAGRAM

Figure 6: Visual Decomposition of Equation Sets 2 and 4 Logic Circuits

Let X1 = (S' + D)
Let X2 = (H2 + D)
Let X3 = (S + D)
Let X4 = (H1 + H2 + X3)
Let X5 = (H1' + H2' + X3)
Let X6 = (P' + D')
Let X7 = (P' + S)

The GIC of the decomposed set of X expressions is 16.

Equation set 2 is decomposed to:

F1 = (P)(S + D')(X1)(H2' + S)(H1' + S) F2 = (P)(X4)(X5)(S' + D') F3 = (P)(X2)(X1)	(5)

The GIC of equation set 5 is 20.

Considering equation set 4, further decompositions can be made by looking at its common maxterms:
Let Y1 = (P' + S' + D) = (P' + X1)
Let Y2 = (H2' + P' + S) = (H2' + X7)

The GIC of the decomposed set of Y expressions is 4.

Equation set 4 is decomposed to:

a = (P' + X4)(P' + X5) b = (S' + X6) c = (Y1)(H1' + P' + X2) d = a e = (H1 + X7)(X6)(Y2) f = (D' + X7)(Y1)(Y2)(H1' + X7) g = (P)(X5)	(6)

The GIC of equation set 6 is 28.

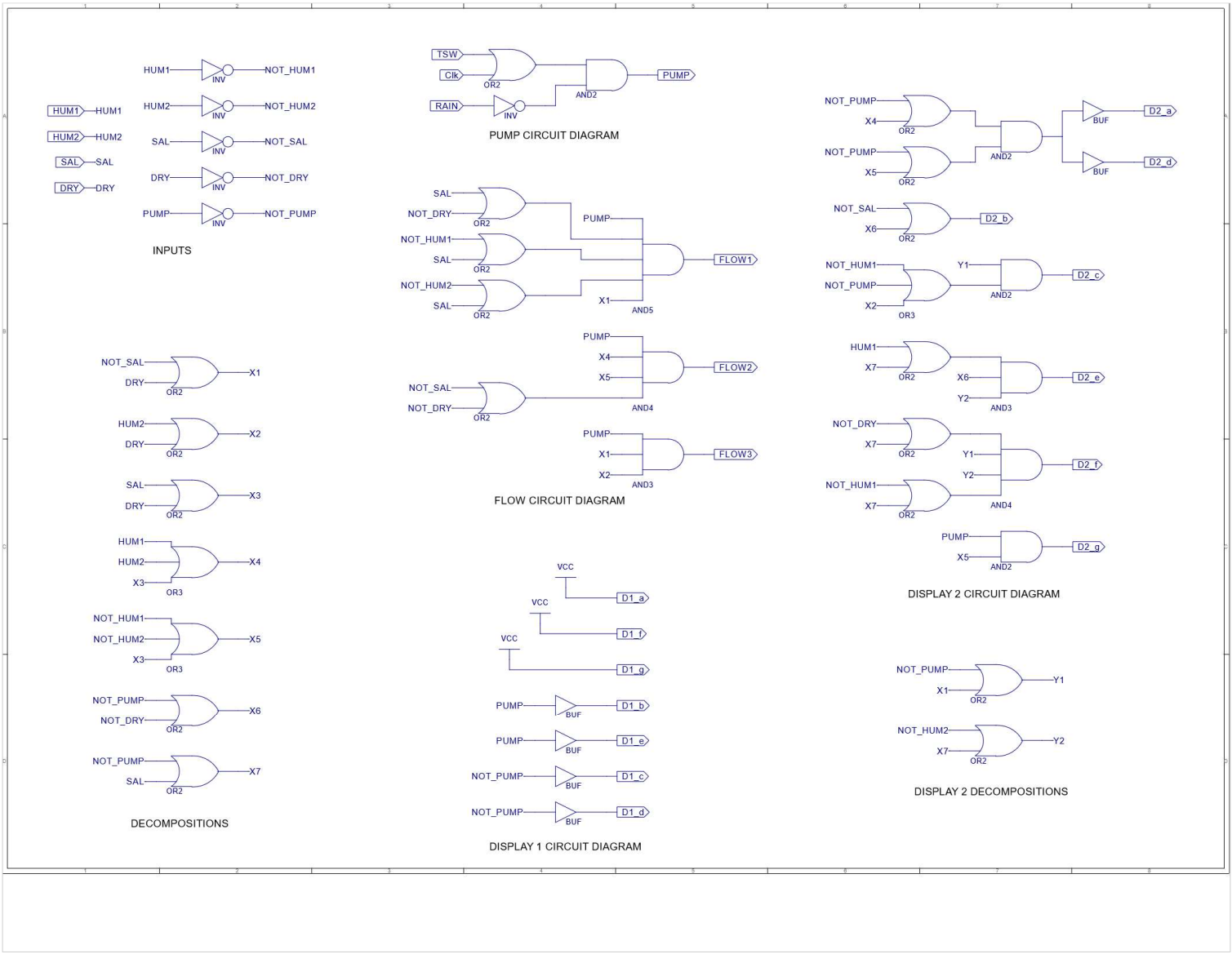
The GIC of complemented terms (excluding the complemented pump as it was already considered in equation set 3) is 4.

Therefore the total GIC is 5 + 1 + 16 + 20 + 4 + 28 + 4 = 78.

Circuit Implementation

Thursday, 19 March 2020 12:04 AM

The final circuit is implemented using basic logic gates from POS expressions of equation 1, equation sets 3, 5, 6, and decomposition expressions. The buffers used in the circuit are ignored in the GIC count as Xilinx requires a buffer to distinguish two I/O markers to the same node.

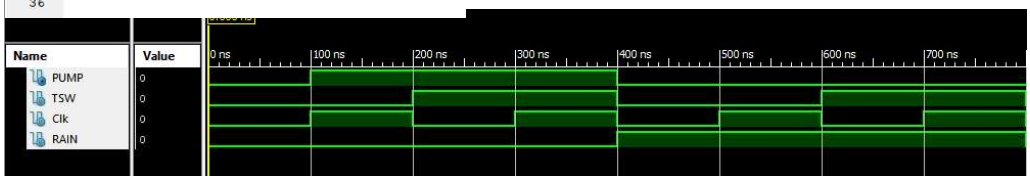


Verification

Thursday, 19 March 2020 12:04 AM

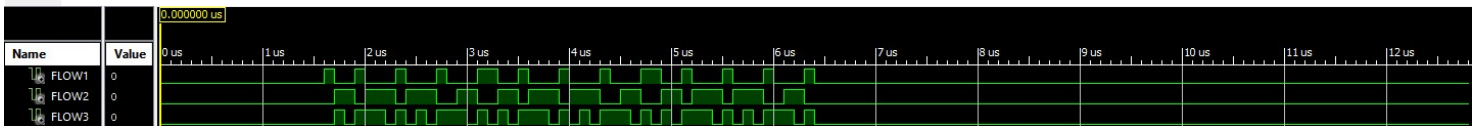
PUMP CIRCUIT:

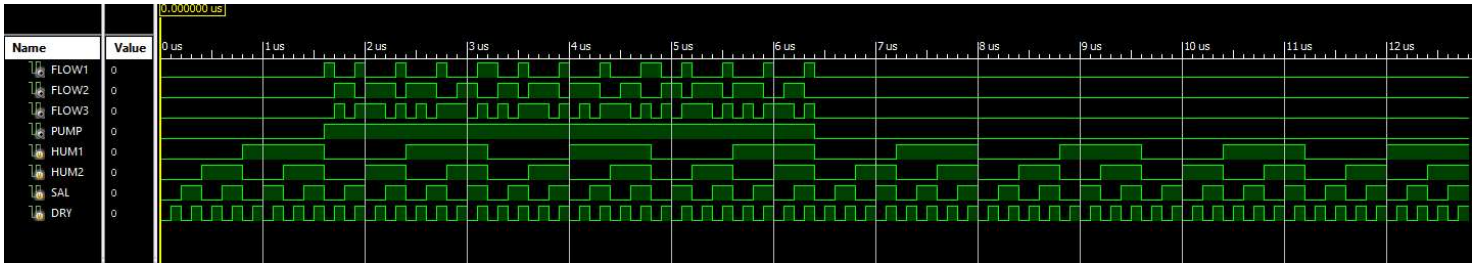
```
// Verilog test fixture created from schematic C:

`timescale 1ns / 1ps

module ass1_q2_ass1_q2_sch_tb();

// Inputs
reg TSW;
reg Clk;
reg RAIN;

// Output
wire PUMP;

// Bidirs

// Instantiate the UUT
ass1_q2 UUT (
    .PUMP(PUMP),
    .TSW(TSW),
    .Clk(Clk),
    .RAIN(RAIN)
);
// Initialize Inputs
initial begin
    TSW = 0;
    Clk = 0;
    RAIN = 0;
    forever begin
        #100
        {RAIN, TSW, Clk} = {RAIN, TSW, Clk} + 1;
    end
end

endmodule
```



FLOW CIRCUIT:

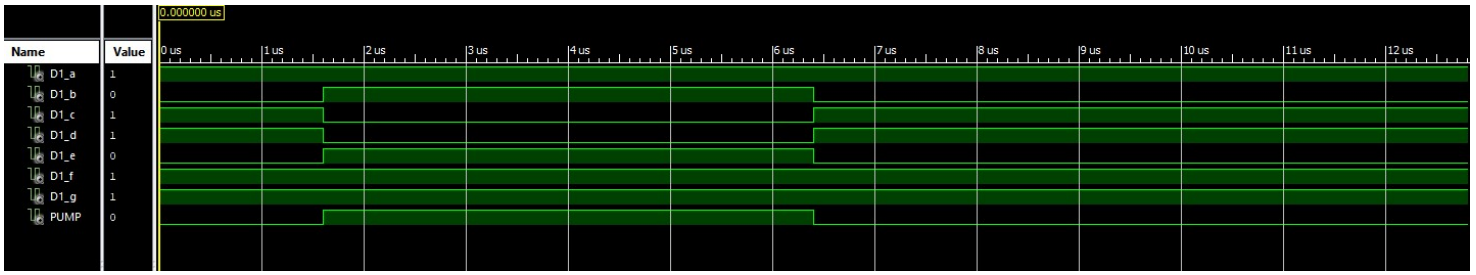
```
// Verilog test fixture created from schematic C:\Users\Dan\Documents\ELEC2141\ass1_q2\ass1_q

`timescale 1ns / 1ps

module ass1_q2_ass1_q2_sch_tb();

// Inputs
reg TSW;
reg Clk;
reg RAIN;
reg HUM1;
reg HUM2;
reg SAL;
reg DRY;

// Output
wire FLOW1;
wire FLOW2;
wire FLOW3;
wire PUMP;

// Bidirs

// Instantiate the UUT
ass1_q2 UUT (
    .TSW(TSW),
    .Clk(Clk),
    .RAIN(RAIN),
    .HUM1(HUM1),
    .HUM2(HUM2),
    .SAL(SAL),
    .DRY(DRY),
    .FLOW1(FLOW1),
    .FLOW2(FLOW2),
    .FLOW3(FLOW3),
    .PUMP(PUMP)
);

// Initialize Inputs
initial begin
    TSW = 0;
    Clk = 0;
    RAIN = 0;
    HUM1 = 0;
    HUM2 = 0;
    SAL = 0;
    DRY = 0;
    forever begin
        #100
        {RAIN, TSW, Clk, HUM1, HUM2, SAL, DRY} = {RAIN, TSW, Clk, HUM1, HUM2, SAL, DRY} + 1;
    end
end

endmodule
```





DISPLAY 1 CIRCUIT:

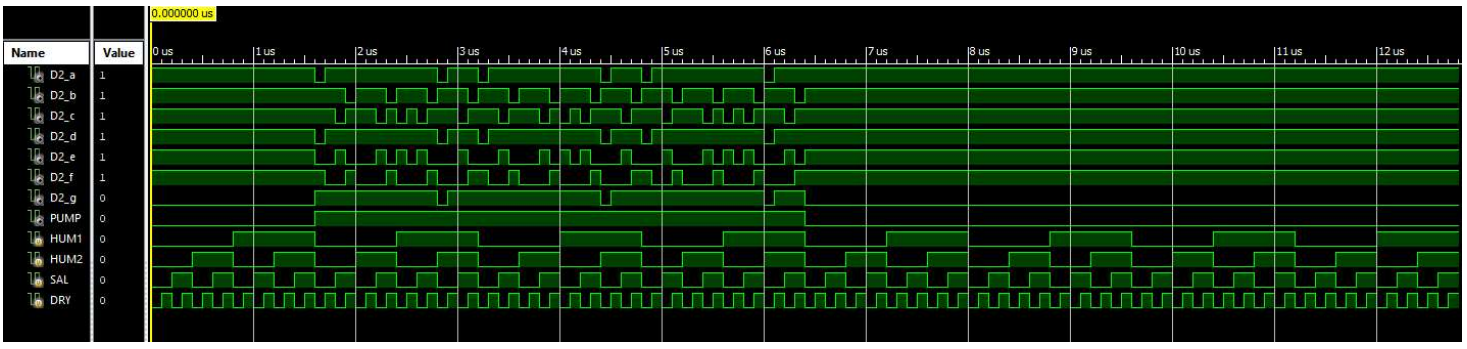
```
module ass1_q2_ass1_q2_sch_tb();

// Inputs
reg TSW;
reg Clk;
reg RAIN;
reg HUM1;
reg HUM2;
reg SAL;
reg DRY;

// Output
wire FLOW1;
wire FLOW2;
wire FLOW3;
wire PUMP;
wire D1_a;
wire D1_b;
wire D1_c;
wire D1_d;
wire D1_e;
wire D1_f;
wire D1_g;

// Instantiate the UUT
ass1_q2 UUT (
    .TSW(TSW),
    .Clk(Clk),
    .RAIN(RAIN),
    .HUM1(HUM1),
    .HUM2(HUM2),
    .SAL(SAL),
    .DRY(DRY),
    .FLOW1(FLOW1),
    .FLOW2(FLOW2),
    .FLOW3(FLOW3),
    .PUMP(PUMP),
    .D1_a(D1_a),
    .D1_b(D1_b),
    .D1_c(D1_c),
    .D1_d(D1_d),
    .D1_e(D1_e),
    .D1_f(D1_f),
    .D1_g(D1_g)
);

// Initialize Inputs
initial begin
    TSW = 0;
    Clk = 0;
    RAIN = 0;
    HUM1 = 0;
    HUM2 = 0;
    SAL = 0;
    DRY = 0;
    forever begin
        #100
        {RAIN, TSW, Clk, HUM1, HUM2, SAL, DRY} = {RAIN, TSW, Clk, HUM1, HUM2, SAL, DRY} + 1;
    end
end
```



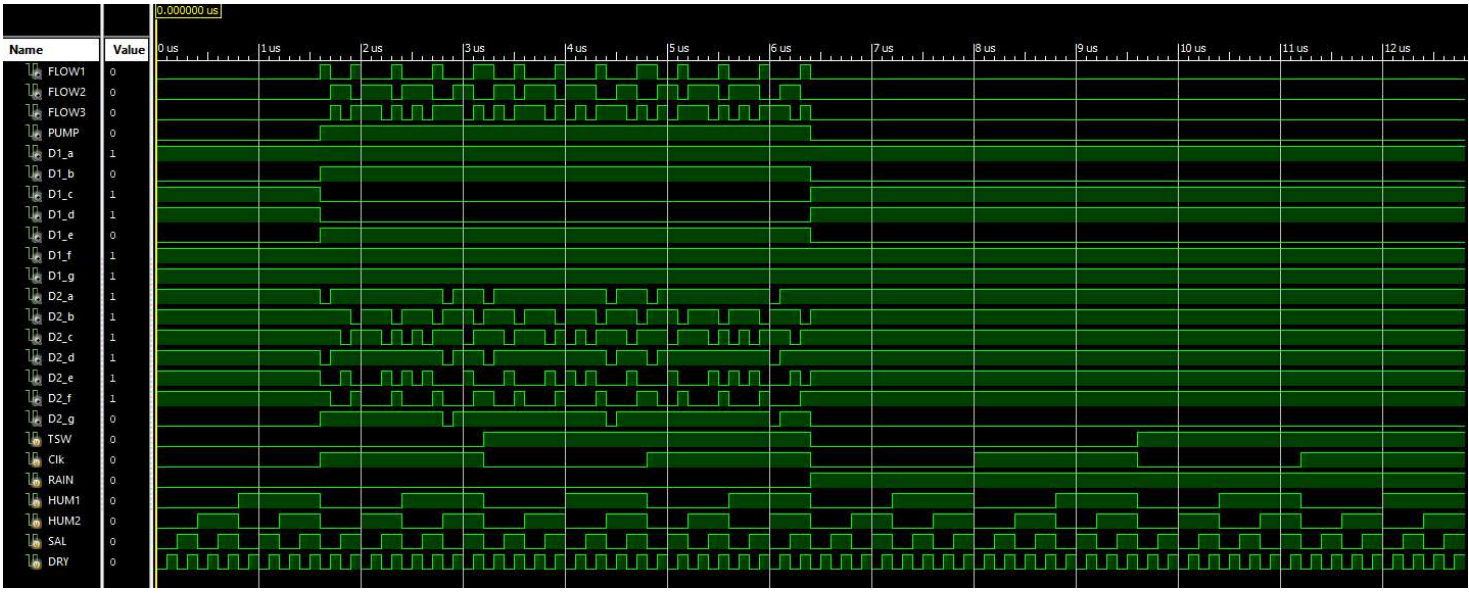
DISPLAY 2 CIRCUIT:

```
// Verilog test fixture created from schematic C:\Users\Dan\Documents\ELEC2141\ass1_q2\ass1_q2
\ass1_q2.sch - Sat Apr 04 16:06:59 2020

`timescale 1ns / 1ps

module ass1_q2_ass1_q2_sch_tb();

// Inputs
reg TSW;
reg Clk;
reg RAIN;
reg HUM1;
reg HUM2;
reg SAL;
```

```
    reg DRY;

    // Output
    wire FLOW1;
    wire FLOW2;
    wire FLOW3;
    wire PUMP;
    wire D1_a;
    wire D1_b;
    wire D1_c;
    wire D1_d;
    wire D1_e;
    wire D1_f;
    wire D1_g;
    wire D2_a;
    wire D2_b;
    wire D2_c;
    wire D2_d;
    wire D2_e;
    wire D2_f;
    wire D2_g;

    // Instantiate the UUT
    ass1_q2 UUT (
            .TSW(TSW),
            .Clk(Clk),
            .RAIN(RAIN),
            .HUM1(HUM1),
            .HUM2(HUM2),
            .SAL(SAL),
            .DRY(DRY),
            .FLOW1(FLOW1),
            .FLOW2(FLOW2),
            .FLOW3(FLOW3),
            .PUMP(PUMP),
            .D1_a(D1_a),
            .D1_b(D1_b),
            .D1_c(D1_c),
            .D1_d(D1_d),
            .D1_e(D1_e),
            .D1_f(D1_f),
            .D1_g(D1_g),
            .D2_a(D2_a),
            .D2_b(D2_b),
            .D2_c(D2_c),
            .D2_d(D2_d),
            .D2_e(D2_e),
            .D2_f(D2_f),
            .D2_g(D2_g)
    );

    // Initialize Inputs
    initial begin
            TSW = 0;
            Clk = 0;
            RAIN = 0;
            HUM1 = 0;
            HUM2 = 0;
            SAL = 0;
            DRY = 0;
            forever begin
                    #100
                    {RAIN, TSW, Clk, HUM1, HUM2, SAL, DRY} = {RAIN, TSW, Clk, HUM1, HUM2, SAL, DRY} + 1;
            end
    end

    endmodule
```



Below is the full simulation results.


0.000000 us
Name
Value
0 us
1 us
2 us
3 us
4 us
5 us
6 us
7 us
8 us
9 us
10 us
11 us
12 us
FLOW1 0
FLOW2 0
FLOW3 0
PUMP 0
D1_a 1
D1_b 0
D1_c 1
D1_d 1
D1_e 0
D1_f 1
D1_g 1
D2_a 1
D2_b 1
D2_c 1
D2_d 1
D2_e 1
D2_f 1
D2_g 0
TSW 0
Clk 0
RAIN 0
HUM1 0
HUM2 0
SAL 0
DRY 0