**CSCU9V4: Systems**

**Assignment - Spring 2019**

**The 'Why-Aye' Pad Controller**

**Student name**: Titirat Seeharach

**Student ID**: 2636568

# Task Marking Scheme Marks distribution

**PART I: Systems**

1. Build the truth table **/5**
2. Derive algebraic expressions: **/20**
   1. Sum-of-minterms expressions /5
   2. Simplification of the expression for Clock Wise /15
3. Draw the CW circuit **/5**
4. Convert CW expression to NAND-only form **/10**
5. Demonstrate NAND-only version is correct with a truth table **/5**
6. Draw the NAND-only version of the CW circuit **/5**

**PART 2: C Programming**

1. Controller Simulator **/20**
2. Simulate Sum-of-Minterms Boolean Expressions **/15**
3. Draw on a Bitmap File **/10**
4. Produce a well presented submission **/5**

## TOTAL **/100\***

(\*This assignment constitutes 40% of the overall module assessment.)

Assignment Semester: Spring Year: 2018-19

|  |  |  |  |
| --- | --- | --- | --- |
| **Module Title:** | *Systems* | **Module Code:** | CSCU9V4 |
| **Tutor:** | *Dr Deepayan Bhowmik & Dr Jingpeng Li* | **Division:** | *Computing Science* |
| **Date Set:** | *Tuesday 5th March 2019* | **Submission ate:** | **Friday 5th April 2019** |
| **Title of Assignment:**  The 'Why-Aye' Pad Controller | | | |
| **Nature of Task & Brief Description:**  The 'Why-Aye' Pad controller has a six-core wire connecting it to the Wi-i console and each wire carries a Boolean/binary value of 0 or 5 volts (or 0 and 1 for convenience). The combination of six values on the wires controls the sprite. Inside the controller there is some circuitry to create the correct 0 or 1 value on each wire for each button combination and it is your job to design part of that circuitry (part 1) and simulate the complete controller using C programming (part 2). | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Word Count:** | n/a | **Assessment Weighting:** | **40 %** |
| **Objectives** | **This assignment is designed to address the following learning outcome of the module** :   * *Design and manipulate simple logic circuits using* truth tables and *Boolean algebra.* * *Develop an emulation lower-level systems details using programming skills in C.* | | |
| **Task Details:** | *The finished assignment must:*   1. *Complete all tasks as described below.* 2. *Clearly be identified and be the student’s own work.* | | |

**Assignment Specification**

**The Wi-iTM *Controller***

Number 910 Fulchester Road in Newcastle is a famous address in the games industry. Whilst other provincial games manufacturers took on the big guns in the games industry with products such as the *Phony PlaySatan* to compete with the *iPhone*, and the Welsh assembly-supported *X-Cist 360* and ‘*GameBoyo*’ (as 'leeked' to the press recently), a small business operating out of a terraced house in the North East of England decided to call its revolutionary new product after the digits in their house number. Thus was born the *NineTenDoor* *Wi-i* (pronounced, as expected in Newcastle, "*Why-Aye*"). This somewhat radical new console was designed to be used by inebriated players, and so had a ruggedised minimalist controller with only 4 buttons, as shown opposite. This was dubbed the '*Eee!*', reflecting the uncalled-for enthrall when first seen by a drunk game-player.

***Wi-iTM***

Games soon appeared for this new console, such as "*You're Me Best Mate!*" a bus-station-platform game, and "*Down The Bigg Market*", a multi-person, free-for-all, shoot and stab 'em up. "*Are You Looking At Me?*" and "*You've Spilled Me Pint!*" soon followed, followed by ports of popular games such as "*Spore*" but with just the first level available, as deemed appropriate. *Shoot 'em Up's*, educational games warning of the perils of drunken drug abuse, are expected Q1 of 2010.

The controller's buttons are designed to be used singly, or in combination with one or two of the others. There are 14 possible combinations with 2 ‘no operations’ - it was decided that pressing all four simultaneously was probably not only impossible for a drunkard, but insensible in terms of game play. The buttons control a sprite in a game (we're not concerned with the game's details here) in the following way:

no buttons nothing happens!

➊ move left (L)

➋ rotate anti-clockwise (ACW)

➌ move right (R)

➍ rotate clockwise (CW)

➊+➋ move down (D)

➊+➍ move up (U)

➊+➌ rotate anticlockwise (ACW) & move left (L) simultaneously

➋+➍ rotate clockwise (CW) & move left (L) simultaneously

➋+➌ rotate anti-clockwise (ACW) & move right (R) simultaneously

➌+➍ rotate clockwise (CW) & move right (R) simultaneously

➊+➋+➌ rotate anticlockwise (ACW) & move down (D) simultaneously

➊+➌+➍ rotate anticlockwise (ACW) & move up (U) simultaneously

➊+➋+➍ rotate clockwise (CW) & move down (D) simultaneously

➋+➌+➍ rotate clockwise (CW) & move up (U) simultaneously

➊+➋+➌+➍ nothing happens!

Figure 1: Button control details

**YOUR TASKS**

The controller logic circuit needs to be designed. The controller has a six-core wire connecting it to the Wi-i console and each wire carries a boolean/binary value of 0 or 5 volts (or 0 and 1 for convenience). The combination of six values on the wires controls the sprite. Inside the controller there is some circuitry to create the correct 0 or 1 value on each wire for each button combination and it is your job to design part of that circuitry (Part-1) and simulate the complete controller using C programming (Part-2). Each part consists of multiple tasks as described below.

Each task has a maximum percentage mark allocated to it - this is repeated on the front sheet of this assignment (which should be attached to your final submission). You will be given full credit for each task done correctly, and pro-rata credit where you have not managed to complete a task or have made a mistake/omission. You will be given feedback on your work after all submissions have been marked.

**Part 1: Systems**

**Task 1: Build the Truth Table**

Create the truth table for the controller. Assume that the buttons when pressed individually or in combination create a 4 bit binary number, between 0000 (010) and 1111 (1510). These four bits are represented for convenience as A,B,C and D as shown below, with D representing the least significant bit. The code for each button combination can be taken to be its ordinality in the table given above.

The table will thus have this form:

button combination code effect on sprite dictated by 0/1 values on these six wires

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **Buttons** | **CW**  **clockwise** | **ACW**  **anti-clockwise** | **L**  **left** | **R**  **right** | **U**  **up** | **D**  **down** |
| 0 | 0 | 0 | 0 | none | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | ➊ | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | ➋ | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | ➊+➋ | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | ➌ | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | ➊+➌ | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | ➋+➌ | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | ➊+➋+➌ | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | ➍ | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | ➊+➍ | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | ➋+➍ | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | ➊+➋+➍ | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | ➌+➍ | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | ➊+➌+➍ | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | ➋+➌+➍ | 1 | 0 | 0 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | ➊➋➌➍ | 0 | 0 | 0 | 0 | 0 | 0 |

In the beginning, I have to set up input such as

Button ➊ is D.

Button ➋ is C.

Button ➌is B.

Button ➍ is A.

In four buttons, they can occur 16 combination following truth table.

In the Button combination code table, If A or other buttons have 1, it means we press button 1 or others. But if A or others have 0, it means we don’t press button 1 or others. For others column, it consists of 6 function on WII from six wires. They are shown about the effect function when we press buttons on WII for example if we see CW (rotate clockwise) has 1, it means CW function occurs. If it has 0, which mean it doesn’t work.

**Task 2: Deriving Sum-of-Minterms Boolean Expressions**

b) Using your truth table from Task 1, create six **sum-of-minterms** expressions, one for each output column:

(Following to truth table, look at values which have 1 in six wire column)

(A’,B’,C’ and D’ are inverse of A,B,C,D)

1. CW = AB’C’D’ + AB’CD’ +AB’CD + ABC’D’ + ABCD’

2. ACW = A’B’CD’ + A’BC’D + A’BCD’ + A’BCD +ABC’D

3. L=A’B’C’D + A’BC’D + AB’CD’

4. R= A’BC’D’ + A’BCD’ + ABC’D’

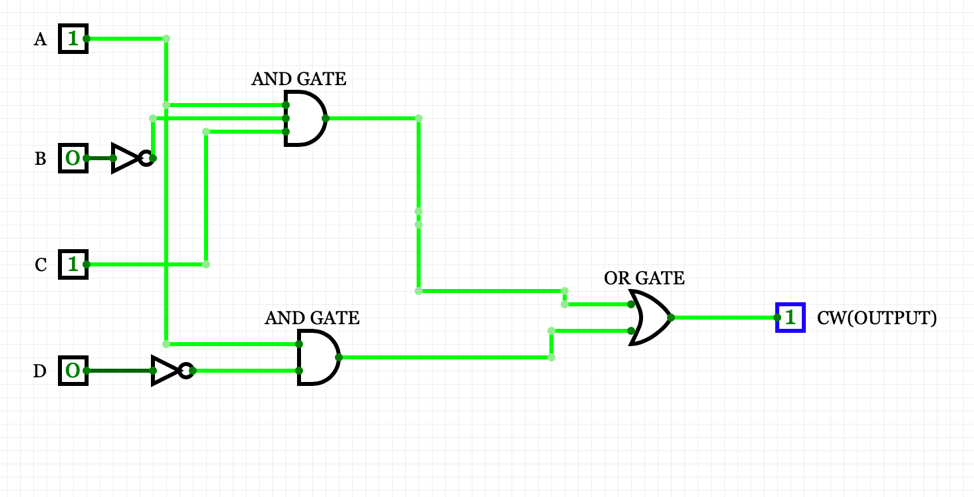
5. U =AB’C’D + ABC’D + ABCD’

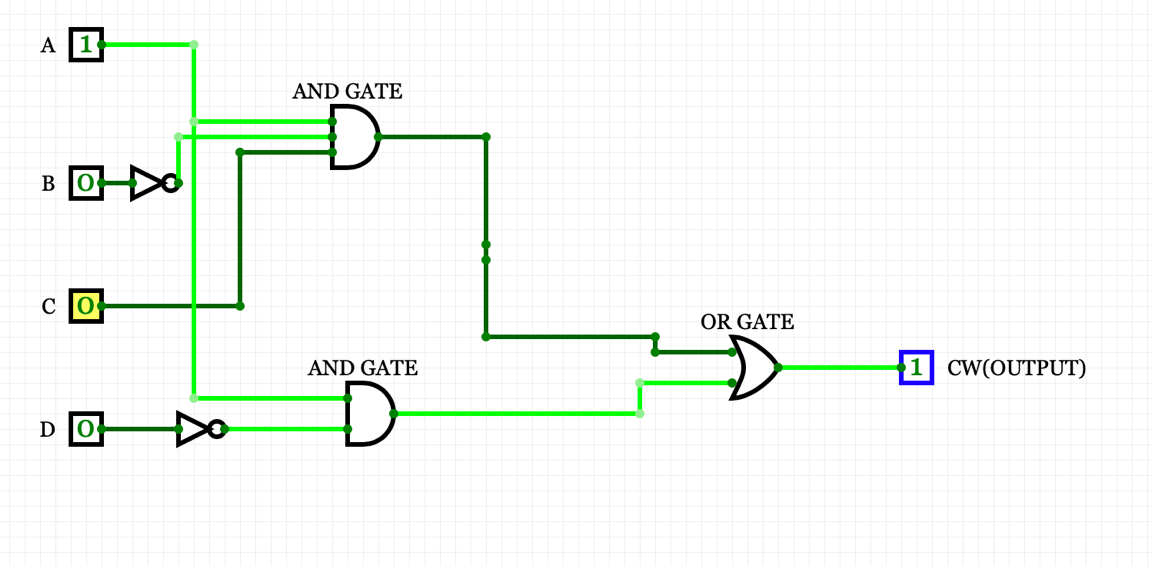
6. D =A’B’CD + A’BCD + AB’CD

1. Using **Boolean Algebra** simplify the minterm expression for **CW** (clockwise) as far as you can, showing each major step that you take. The rule you are applying at each step should also be made clear (e.g. "distributive law", "", etc).
   1. CW = AB’C’D’ + AB’CD’ +AB’CD + ABC’D’ + ABCD’
   2. CW = AB’C’D’ + AB’C (D’ +D) + ABC’D’ + ABCD’ ("D’ + D =1")
   3. CW = A C’D’ (B’+ B) + ABCD’+ AB’C ("B’+B =1")
   4. CW = A C’D’ + ABCD’+ AB’C
   5. CW = AC’D’ + AC (BD’ + B’) (A + A’B = A + B)
   6. CW = AC’D’ + ACD’ + ACB’
   7. CW = A D’ (C’ + C) + ACB’. ("C’+C = 1")
   8. CW = A D’ + ACB’

**Task 3: Draw the circuit for simplified CW logic/expression**

Using your simplified expression for the **CW** (clockwise)logic **only**, draw the logic circuit using AND, OR, NOT and other logic gates, as required. Your diagram should have the following format:





In the picture, I represent about input which is ➋+➍ and ➍ (in Task 1) occurring CW function.

**Task 4: Convert the CW expression to use NAND gates only**

Use **de-Morgan's theorem** to convert your expression for the CW circuit so that it uses **NAND gates only.** You should clearly show each step that you take in doing this in your report.

(**de-Morgan's theorem** )

. (**de-Morgan's theorem inside** and )

. (**de-Morgan's theorem inside** )

**Task 5: Demonstrate that the NAND version of CW is correct**

Use a truth table to show that the NAND version you now have for CW agrees with the original output. Your table should have the general form shown below:

A, B, C and D are the original inputs and column number 1 is the column for CW (clockwise) taken from the table created in Task 1. Columns 2 to 5 (you will probably need more columns than this) represent the steps taken as you incrementally deduce (from the left to right) the results for your **NAND version** of the expression. You should have sufficient intermediate steps in your table (for example, showing , then  as separate columns) with the final, right-most column representing the final value for CW. This should match up with the original CW in column 1 (if it does not, and you cannot find where you have gone wrong, you will still get credit for the attempt).

**Hint**: Using a spreadsheet program such as Excel is a convenient way of building a truth table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | CW |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

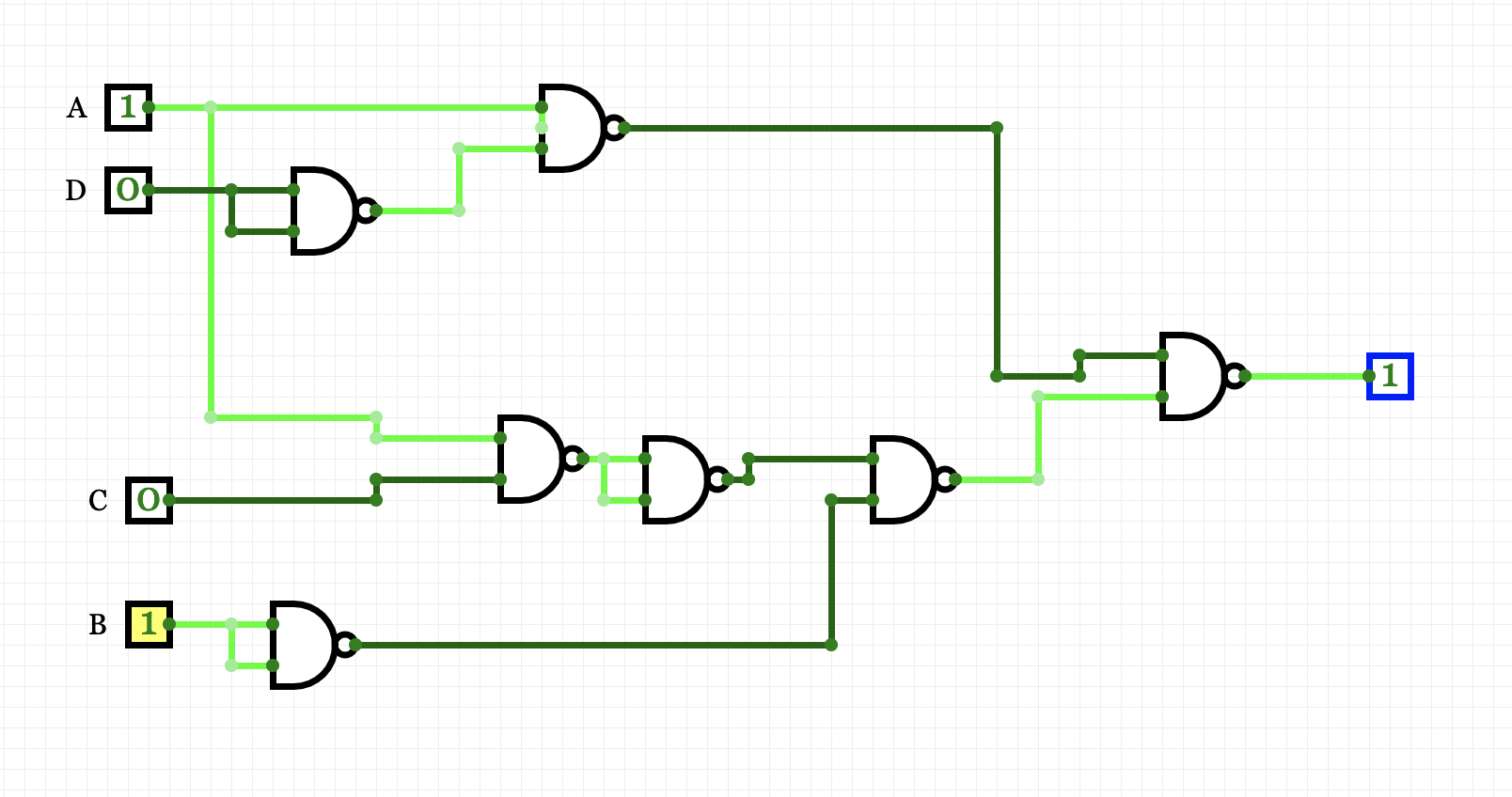
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  | CW |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

**In blue columns, They are shown from the Task 1. White columns are proved from Task4 (from last equation) which get the same result both CW column.**

**Task 6: Draw the NAND-only version of the CW circuit**

Using the NAND-only circuit expression derived in Task 4, draw the NAND-only version of the CW circuit. Use the same basic format for the diagram as suggested in Task 3. If task 5 shows up an error in your NAND-only version, you should still draw the circuit, errors included; you will get credit for a correct rendition of the expression.

Following Task 4 (last equation).

****

**Part 2: C Programming**

In Part 2 of the assignment, you are required to build a simulator for 'Why-Aye' Pad Controller using C programming. There are a number of tasks within this part with increasing difficulty.

**NOTE: Programs must compile to be considered for assessment.** Generally speaking, work is graded primarily along the following criteria:

* Scope – the extent to which code implements the features required/specified.
* Correctness – the extent to which code is consistent with the specs and bug-free.
* Design – the extent to which code is well written, i.e. clearly, efficiently, elegantly, or logically.
* Style – the extent to which code is readable, e.g. comments, indentation, apt naming.

**Task 7: ‘Why-Aye’ Pad Controller simple simulator**

In this task you need to write a simple ‘C’ program that performs following operations:

1. Accept input from your keyboard either numbers (1,2,3,4) or keyboard arrow keys representing four buttons in the controller.
2. On pressing different combinations of keys it should display texts on the console as given in Figure 1 on page 3.
3. Improve your sub-task 7.b) by displaying graphical representation on the console. For example, if your key combination indicates ‘Move left’ you can display a **←** shape (which is composed of a number of simple characters like ‘\*’, ‘+’, ‘-‘, ‘x’, or ‘<’) on your console. Be innovative for different outputs.

**It will be appeared in WIITask7 folder in main.c file.**

**Task 8: Simulate Sum-of-Minterms Boolean Expressions**

In this task you need to modify the program created in Task 7 and emulate the Sum-of-Minterms Boolean Expressions you have derived in Task 2. You need to follow the instructions below:

1. Your keyboard input should be stored in four Boolean variables, such as A, B, C, D
2. You should write equations for CW, ACW, L, R, U & D using the variables in previous step and C logical operators.
3. Finally use the output of CW, ACW, L, R, U & D to produce the same output as shown in Figure 1.

Before doing Task 8, we must find all Boolean algebra simplify all 6 function. We get

CW = A D’ + ACB’

ACW= A'CD' + A'BD + BC'D

L = A'C'D + AB'CD'

R= A'BD' + BC'D'

U=AC'D + ABCD'

D = A'CD + B'CD

**Task 9: Draw on a Bitmap File**

[This is an optional component and should only be attempted after other tasks are complete.]

Create a bitmap image file of a plain colour (called *“mydraw.bmp”*). Modify the bitmap file to let it display the graphics you draw in the console window in Task 8. You should not change the file name.

Apart from doing some extra reading on your own, you may refer to the following webpage about how to write code to read, write and crop BMP image files:

<https://engineering.purdue.edu/ece264/17au/hw/HW15>

***It will be appeared in* WIITask8  *folder in main.c file.(both task 8 and 9)***

**Task 10 Produce your submission report**

The tasks outlined above should be carried out and your results presented in the form of a brief report. This means word processing text and creating diagrams using a drawing tool, and not submitting hand-drawn diagrams and tables. Your submission should be well-presented, spell checked where necessary and proof read, and in the order of the tasks requested. Up to 5% of the marks are available for fulfilling this criteria. Please do not forget to include the Mark Sheet from this assignment as the first page of your submission.

**Outcomes**

This work is designed to test your understanding of logic circuits, truth tables, Boolean algebra and your skill in C programming. Marks are awarded for correct and accurate use of these logic design techniques, efficient coding and producing a clear and structured report.

**Submission**

You will need to submit your work on Canvas as a zipped file bearing your university username (3 letters + 5 digits, e.g., xyz00001.zip). This zip file should contain a folder also named after your username. Place all the files created during the assignment in this folder.

* This is an individual piece of work.
* **Please ensure that the cover Mark Sheet (i.e page 1) is included in the report.**

**The deadline for this assignment is**

**5:00 pm on Friday 5th April 2019**