3. Semester, 2024

Computerarkitektur og -organisering

SW-CAO1

Course Assignment 1

Authored by:

Kristian M. P. Dashnaw (344849)

Task A:

Write down the Arithmetic equation for the logic described in Table 1, and reduce it as much as possible. Put the results in the document.

Table 1: Truth table

A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

In order to formulate a arithmetic equation for the shown truth table, we must first look at all the instances where **Out** results in an output of **1**. For each instance of A, B or C on such lines we must either put NOT A (if A is 0) and A (if A is 1). Each line is then added to the equation through the OR (+) statement.

From the table we extract this initial equation:

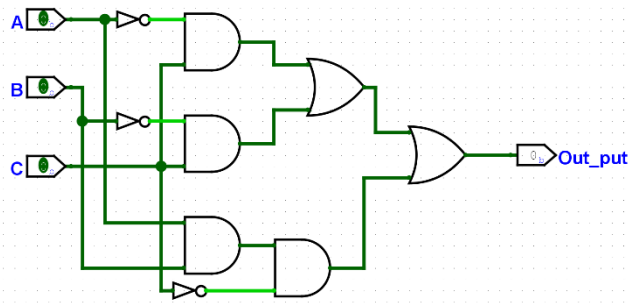
1) $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C}$

The above can be further reduced:

1) $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C}$
2) $\bar{A}C(\bar{B} + B) + A\bar{B}C + AB\bar{C}$
3) $\bar{A}C(1) + A\bar{B}C + AB\bar{C}$
4) $\bar{A}C + A\bar{B}C + AB\bar{C}$
5) $C(\bar{A} + A\bar{B}) + AB\bar{C}$
6) $C(\bar{A} + \bar{B}) + AB\bar{C}$
7) $\underline{C\bar{B} + C\bar{A} + AB\bar{C}}$

Task B:

*Implement the circuit in Logisim using **AND, OR** and **INVERTER** gates. Create a Logisim file called **task_b.circ** with the designed circuit and insert in the zip file.*



A	B	C	Out_put
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Please see attached file for Logisim schematic. Above is extracted from that file!

Task C:

Write down the Arithmetic equation for the logic described in Table 2 and reduce it as much as possible. Put the result in the document.

Table 2: Truth table

A	B	C	Out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

In order to formulate a arithmetic equation for the shown truth table, we must first look at all the instances where **Out** results in an output of **1**. For each instance of A, B or C on such lines we must either put NOT A (if A is 0) and A (if A is 1). Each line is then added to the equation through the OR (+) statement.

From the table we extract this initial equation:

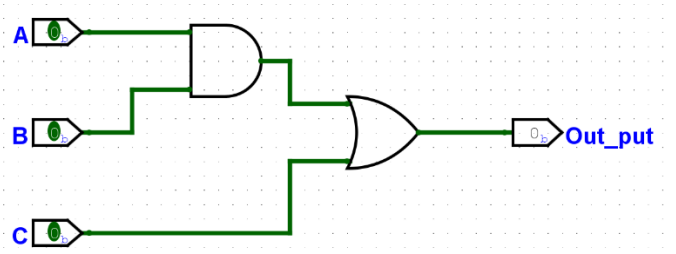
1) $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

The above can be further reduced:

1) $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$
2) $\bar{A}C(\bar{B} + B) + A\bar{B}C + AB(\bar{C} + C)$
3) $\bar{A}C(1) + A\bar{B}C + AB(1)$
4) $\bar{A}C + A\bar{B}C + AB$
5) $\bar{A}C + A(\bar{B}C + B)$
6) $\bar{A}C + A(C + B)$
7) $\bar{A}C + AC + AB$
8) $C(\bar{A} + A) + AB$
9) $\underline{C + AB}$

Task D:

*Implement the circuit in Logisim using **AND**, **OR** and **INVERTER** gates. Create a Logisim file called **task_d.circ** with the designed circuit and insert in the zip file.*



A	B	C	Out_put
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Please see attached file for Logisim schematic. Above is extracted from that file!

Task E:

Draw the output of the DFF in Figure 1 on Figure 2 below and insert the result in the document.

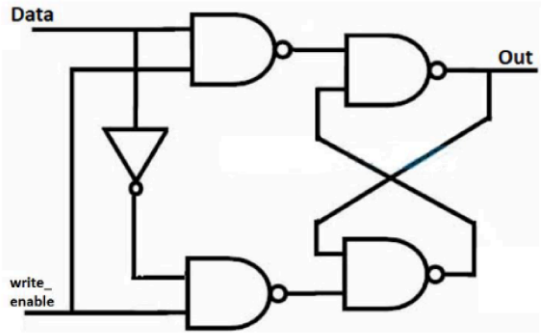


Figure 1 Data Flip Flop (DFF):

Based on the principle of Data Flip Flips, where data output can only be changed when write_enable is powered high, the following output can be drawn onto the timing diagram (See **red line**):

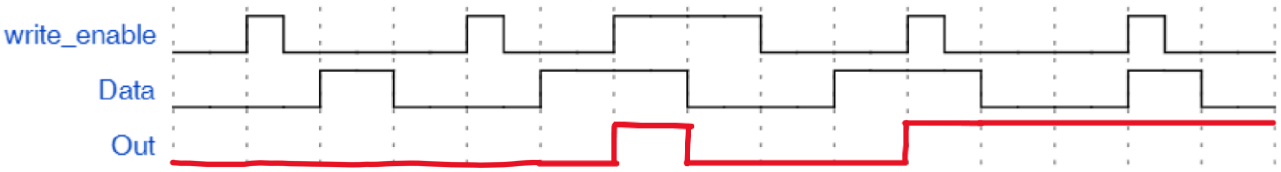


Figure 2: DFF timing diagram

The 8-bit AVR CPU is running a program stored in the memory. The first 3 instructions can be seen in Figure 3.

Address	Opcode	Hint	Explanation
0	1110 0001 0011 1111	Look at page 129 in the AVR instruction set manual	
1	0110 1100 0011 0100	Look at page 106	
2	0111 1010 0011 1100	Look at page 28	

Figure 3: Three instruction, that are fetched and executed by the CPU

The instruction set manual for the AVR processor can be found in the course pages in Itslearning.

Task F:

Explain what happens in each of the instructions above and write down the value of R19 after each of the instructions. Insert the answers in the document.
(Hint if done correctly register 19 (R19) should contain the value 140 in the end)

Address	Opcode	Ref.	Explanation
0	1110 0001 0011 1111 1110 KKKK dddd KKKK	Page 92	Opcode: LDI dddd = 0b 0011 = 3 -> R16+3 = R19 KKKK = 0b 0001 1111 = 31 Command: LDI R19, 31 **Meaning:** Load the value 31 into register 19.
1	0110 1100 0011 0100 0110 KKKK dddd KKKK	Page 106	Opcode: ORI dddd = 0b 0011 = 3 -> R16+3 = R19 KKKK = 0b 1100 0110 = 198 Command: ORI R19, 198 **Meaning:** Perform logical OR between the contents of register 19 (31) and the constant '198' and place the result into register R19. Result can be determined to be: 0001 1111 OR 1100 0100 = 1101 1111 = 223 **Register R19 now contains the value: 223**
2	0111 1010 0011 1100 0110 KKKK dddd KKKK	Page 28	Opcode: ANDI dddd = 0b 0011 = 3 -> R16+3 = R19 KKKK = 0b 1010 0011 = 163 Command: ANDI R19, 163 **Meaning:** Perform logical AND between the contents of register 19 (223) and the constant '163' and place the result into register R19. Result can be determined to be: 1101 1111 AND 1010 1100 = 1000 1100 = 140 **Register R19 now contains the value: 140**

Address	Opcode	Hint	Explanation
0	1110 1111 0010 1111	Look at page 129 in the AVR instruction set manual	
1	1110 0001 0110 0100	Look at page 129	
2	0000 1111 0010 0110	Look at page 25	

Task G:

Fill out the explanations in the table below. If done correctly R18 contains the value 19 in the end.

Address	Opcode	Ref.	Explanation
0	1110 1111 0010 1111 0110 KKKK dddd KKKK	Page 92	Opcode: LDI dddd = 0b 0010 = 2 -> R16+2 = R18 KKKK = 0b 1111 1111 = 255 Command: LDI R18, 255 **Meaning:** Load the value 255 into register 18.
1	1110 0001 0110 0100 0110 KKKK dddd KKKK	Page 92	Opcode: LDI dddd = 0b 0110 = 6 -> R16+6 = R22 KKKK = 0b 0001 0100 = 20 Command: LDI R22, 20 **Meaning:** Load the value 20 into register 22.
2	0000 1111 0010 0110 0000 11rd dddd rrrr	Page 25	Opcode: ADD rrrrr = 10110 => 22 -> R22 ddddd = 10010 => 18 -> R18 **Meaning:** Add Register 18 (255) to Register 22 (20) (**without carry**) and save result in Register 18. Result can be determined to be: ´ 1111 1111 + 0001 0100 = 0001 0011 = 19 (Carryover was thrown away) **Register R18 now contains the value: 19**