Introduction to Programmable Logic Controllers Ex4_logics

DTU 31343

Eduard Maximilian Fiedler s210134

Logic Functions

To achieve the desired logic combinations for this exercise, the following was noticed:

- The red light must illuminate when switch 3 is activated with any or both of the other switches activated, or when all 3 switches are deactivated,
- The yellow light must stay off when there is no switch activated or when switch 1 and 3 are activated,
- The green light must stay off when all switches are activated, only switch 2 is activated, switch 2 and 3 are activated, or when switch 1 is activated.

For the red light, the coil was energised when switch 3 was closed with either switch 2 or 1, or a parallel series of all inverted switches remained. The implemented configuration is shown in Figure 1.

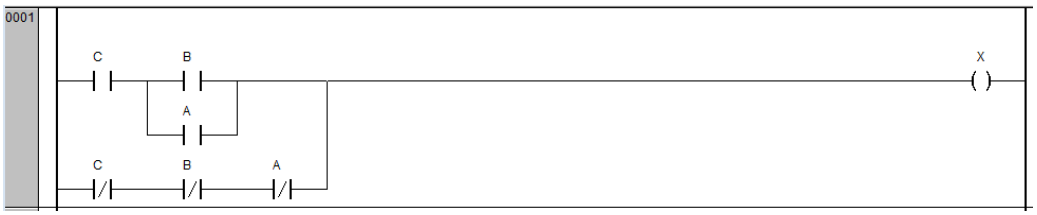

Figure 1: Achieving the red light functionality.

For the yellow light, switch 1 and 3 were placed in parallel, and then again in parallel in series to this. In parallel, a single switch 2 was added such that the yellow light illuminated when this switch was activated. The implemented configuration is shown in Figure 2.

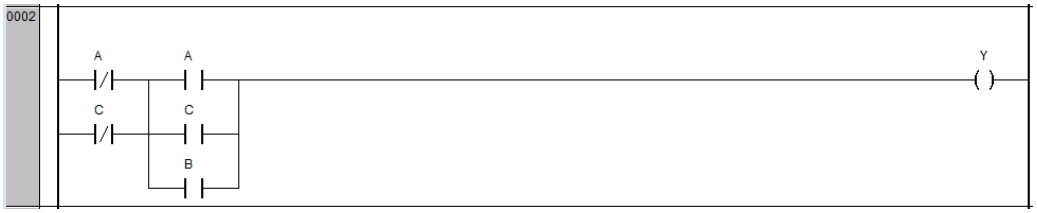

Figure 2: Achieving the yellow light functionality.

For the green light, 3 parallel lines were used. One where switch 3 and an inverted switch 2 were used, another where switch 1 and 2 and an inverted switch 3 were used, and finally where an inverted switch 1, 2, and 3 were used. The implemented configuration is shown in Figure 3.

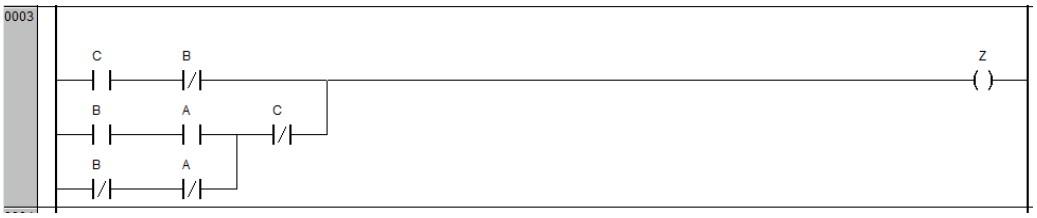

Figure 3: Achieving the green light functionality.

Converting the LD solution for the given truth table, the FBD diagram solution was also correct.

Priority of Networks

Implementing this solution results in a conflict with the prioritisation of outputs in the program. Here, activating switch 1 resulted in no output, activating switch 2 resulted in the green light illuminating, and activating switch 3 resulted in the yellow and red light illuminating.

Problems arise in a ladder diagram implementation when the same output appears in different rungs, as the priority then becomes ambiguous as the complexity of a program increases. A better implementation respects that there is only 1 unique output for each rung in the ladder diagram. Applying this results in the following configuration (see Figure 4).

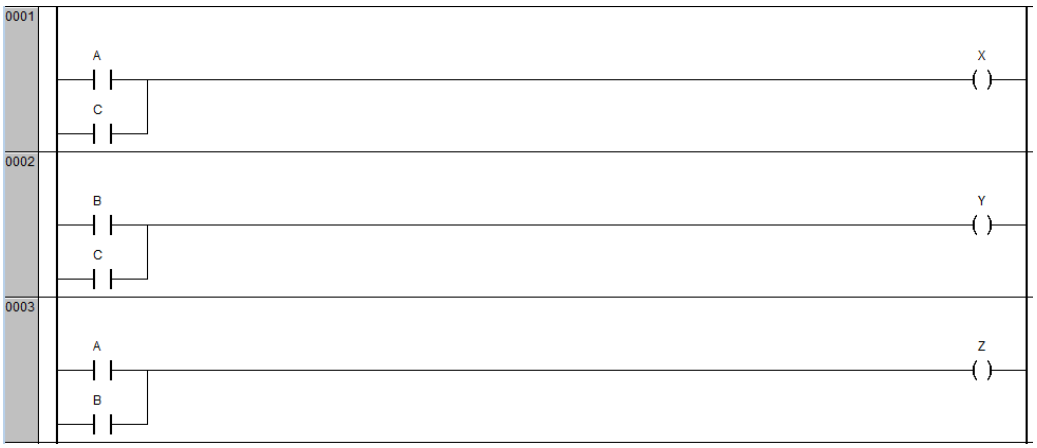

Figure 4: Better solution achieving the desired functionality.

Retentive Coils

For this exercise, it is assumed that the triggering of the individual switches is based off of a single press (i.e. short press and release) and not a prolonged hold, given that the bottle presses the respective switch and will start moving in the other direction which releases the press of the switch. Therefore, in order to implement this a set is given by the activated switch, which is then reset by the other switch. The emergency button resets both latched states.

Using normal coils, the implementation as seen in Figure 5 was used.

Figure 5: Conveyor system control logic using normal coils.

Using the retentive coils, the implementation as seen in Figure 6 was used.

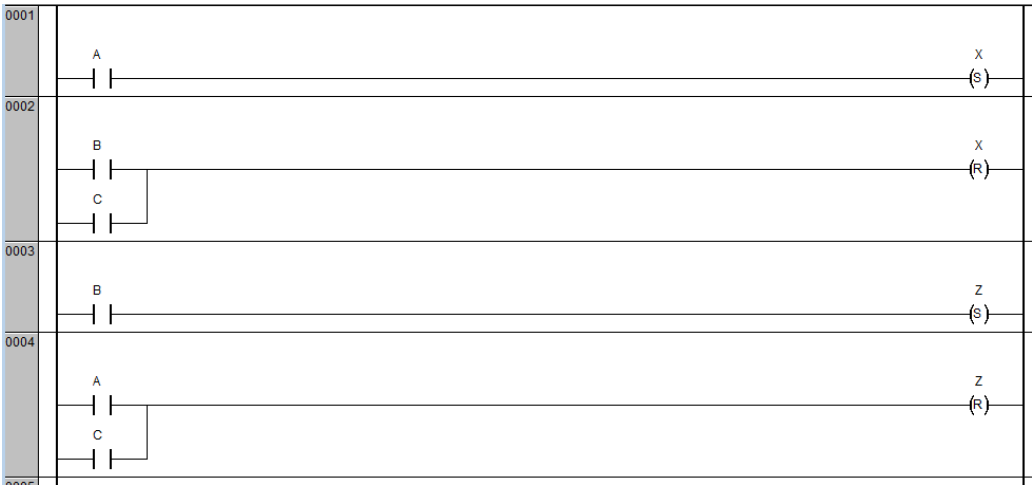

Figure 6: Conveyor system control logic using retentive coils.

Implementing this with a single switch is definitely possible. In that case, button presses could be counted and based on whether the press results in an odd or even number, the conveyor belt could move either left or right. This solution, however, is not as robust as the method using 2 different buttons.