Theory of Logic Circuits

Laboratory manual

Exercise 4

Asynchronous sequential logic circuits

1. Introduction

All switching circuits belong to one of two classes: combinational or sequential. On contrary to combinational logic circuits, the operation of sequential ones is dependent not only on the present state of external inputs, but also on the state of these inputs in past. Therefore, sequential logic circuits are sometimes referred to as circuits with memory. The memory block is formed by some additional signals looped backward from the output to the input of the circuit. These signals are referred to as signals of internal state of the circuit. By this backward loop, the dependence on previous values of inputs is implemented as dependence on the current internal state of the machine.

The sequential switching circuits are asynchronous or synchronous. Asynchronous ones operate at time events defined by changes of inputs (on contrary to synchronous circuits operating at time events defined by additional control signal called clock). Asynchronous sequential logic circuits can be further divided into two subclasses: fundamental mode (or static), and pulse mode (or dynamic) circuits. Fundamental mode asynchronous switching circuits operate according to voltage levels of inputs, while pulse mode ones operate according to changes in voltage levels. Since dynamic asynchronous sequential logic circuit are very sensitive on error condition like spikes, they are used very rarely and will not be discussed here.

Let us denote by **x** the input vector and by **Z** the output vector. Furthermore, let **q** and **Q** denote vectors of internal state of the machine at present and at next time event respectively. Then the operation of asynchronous sequential circuit can be formally assigned as:

$$\begin{cases} \mathbf{Z} = \mathbf{F}_Z(\mathbf{x}, \mathbf{q}) \\ \mathbf{Q} = \mathbf{F}_\mathbf{Q}(\mathbf{x}, \mathbf{q}), \end{cases}$$

if Mealy's machine is concerned, or as:

$$\begin{cases} \mathbf{Z} = \mathbf{F}_Z(\mathbf{q}) \\ \mathbf{Q} = \mathbf{F}_\mathbf{Q}(\mathbf{x}, \mathbf{q}), \end{cases}$$

for Moore's machine.

Asynchronous sequential logic circuits operate faster than synchronous ones, as the outputs are set up just after the propagation delays of used elements (on contrary to synchronous systems operating with a speed determined by a clock frequency). However, this speed is occupied by existence of some requirements that have to be taken into consideration at the design stage, to assure the desired operation of the circuit. These requirements are given below:

1. Only one input signal can be changed at a time. Since this requirement is always satisfied in reality, hence the designer is not allowed to assume the possibility of simultaneous change of two (or more) inputs.
2. Each next change of input state can be done after time τ needed to make internal state of automata settled.
3. It is strongly recommended (not always required, as explained below) to assume at the design stage that only one internal state signal can change as the result of a change of input state. The reason is similar to requirement no. 1. If the designer allows at the design stage the simultaneous change of two internal state signals (what is impossible in reality) the phenomenon called race will occur. However, the race does not always cause erroneous operation of the circuit. Therefore it is possible to design circuit with a race if it is non-critical or driven one. But it is necessary to avoid critical race since the design with it can make erroneous operation of the circuit.
   The design of the asynchronous sequential logic circuits can be made with the use of Switching Sequence Table (SST) or Huffman's method. Both methods give possibility for implementing the memory block with or without flip-flops. The advantage of SST method is that it always designs circuits without a race. However its drawback is that using this method it is unable to obtained Mealy's machine (which is sometimes simpler comparing to Moore's machine). Huffman's method is more universal, since it is possible to design both types of circuits with it.
   Below the main steps of two methods are given.

1.1. Switching Sequence Table Method

1. Obtain SST from the description of operation of the circuit
2. Check solvability of SST.
3. If SST is solvable go to 5.
4. Resolve unsolvable SST
5. Obtain cannonical SOP and POS forms for output signals and additional signals
6. Obtain minimum equations describing circuit for implementation with or without flip-flops
7. Draw logical diagram of the designed circuit

1.2. Huffman's Method

1. Obtain the primitive flow map of the circuit
2. Reduce equivalent and pseudo-equivalent states and obtain the primitive flow map without redundant states
3. Merge rows of the primitive flow map without redundant states (The merging should be done according to merger diagram, and it differs dependent on the Mealy or Moore type of the machine) and obtain merged flow map.
4. Encode rows of the merged flow map, using encoding diagram to avoid races (at least to avoid critical races).
5. Obtain K-maps for outputs and internal state signals from encoded merged flow map.
6. Obtain minimum equations describing circuit for implementation with or without flip-flops
7. Draw logical diagram of the designed circuit

2. Tasks to be performed during laboratory

1. Implement with chosen by the supervisor elements (with or without sr latches), a circuit controlling the switching of two pumps. The pumps P_1 and P_2 (see fig.1) should be switched on alternately (only one pump can work at a time) when water exceeds the level of the sensor x_2 (i.e. when $x_2 = 1$). Working pump should be switched off when the water lever is below the sensor x_1 (i.e. when $x_1 = 0$). Assume that water level grows when pumps are off, and that it decreases when any pump is working.

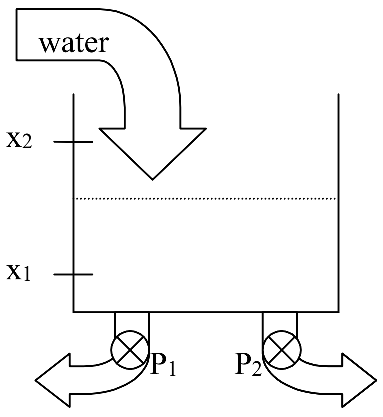

Fig 1. Pumps controlling the water level

2. Implement with chosen by the supervisor elements (with or without sr latches), a circuit controlling the operation of the inertial two-directional engine (fig 2). The engine can start to rotate only if it is stopped (RIGHT = 0, LEFT = 0, STOP = 1). The engine should start to rotate in right direction (RIGHT = 1) when button R is pressed for a short moment and it should keep rotating until button S is pressed. Pressing the R or L button when engine rotates right should be ignored. The engine should start to rotate in left direction (LEFT = 1) when button L is pressed for a short moment and it should keep rotating until button S is pressed. Pressing the L or R button when engine rotates left should be ignored. Similarly pressing S button when engine is stopped should not change its state. Pressing it when engine rotates in any direction should stop it by assigning outputs: RIGHT = 0, LEFT = 0, STOP = 1. Since all input buttons are monostable radio ones, it is assumed that only one of buttons S, L, R can be equal to one at a time.

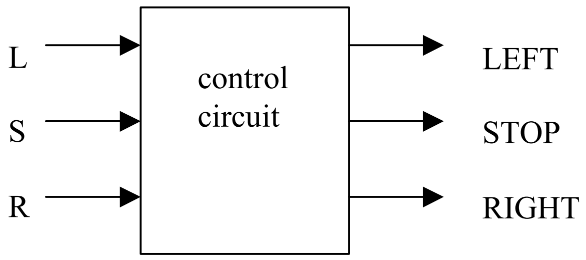

Fig. 2 Control circuit for inertial two-directional engine

3. Instructions to follow

1. Solve all tasks before the exercise.
2. Implement the circuits specified by your supervisor (using given elements).
3. Present working circuits to your supervisor for acceptance.