Practical 4A

Digital Electronics - Sequential Circuits

Before the practical ensure that you complete the following:

Read and revise:

Chapter 3 - Sequential logic (pp. 39 to 64) in the **EEE2046F/EEE2050F course notes**. Please note that the practicals for this course contain a large self-study component and therefore be prepared before the start of the practical.

4A.1 Introduction

In this practical you will investigate **Sequential Circuit** design using Present/State Next state diagrams, Algorithmic State Machines and Timing Diagrams.

This practical contains both a report and Logisim circuit submission and must be completed before the deadline. Submit both your document and **.circ** files on VULA by the due date.

Please name your practical report as follows:

Prac4A-STUDENTNUMBER

Call a teaching assistant or tutor if you need assistance at any stage during the practical session.

You need the following for this practical:

- A PC/laptop running Logisim

Simulate your logic circuits in Logisim and include a diagram of the circuit in your report when it is complete.

- Download **Logisim** from Vula (Resources → Software) to your computer or from **Logisim**
- Open Logisim on your computer
- Create two new design sheets

- Name them as follows:

Prac4APart1-STUDENTNUMBER.circ

Prac4APart2-STUDENTNUMBER1.circ

This practical contains both physical experiments as well as a practical report write-up and may take longer than the 2 hour practical session to complete. Please ensure that you take accurate and detailed notes during the physical experiments so that you can compile them into a report after the practical for submission on VULA by the due date.

Please name your practical report as follows:

Prac3A-STUDENTNUMBER

Call a teaching assistant or tutor if you need assistance at any stage during the practical session and once you have completed the experiments for sign-off.

4A.2 Sequential circuits

The types of circuits we have considered up to this point have been made up of a combination of logic gates and although they can be designed to produce some useful circuits, more complex circuits often need to 'store' information about their previous behaviour.

In this practical we will investigate logic circuits which have 'memory'. In other words their current output values depend not only on the current input values but on previous input values too. These are known as *sequential circuits*. The 1-bit binary memory elements in a digital sequential circuit are made using *latches* or *flip-flops* and are used to hold the current or *present* state of the system. The *state* of a sequential circuit refers to the contents of all the stored memory elements at a particular point in time. The current value of the inputs together with the present state of the system will determine the current output values [10]. We can see a block diagram representation of this in figure 4A.1.

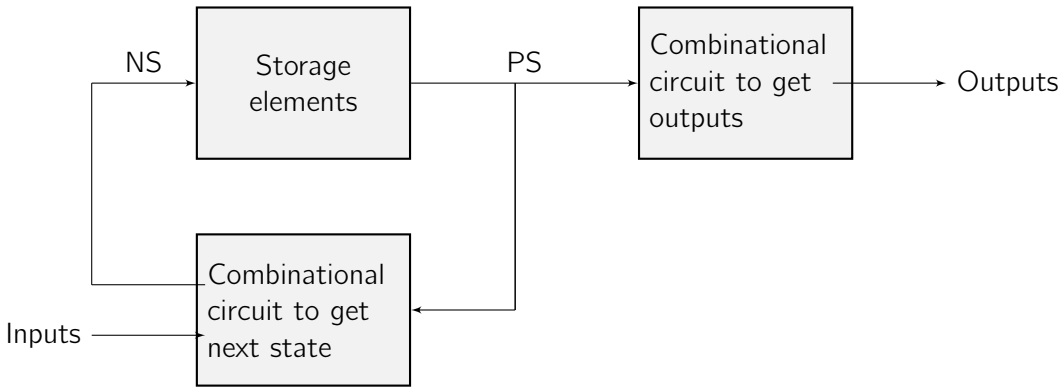


Figure 4A.1: The block diagram for a sequential circuit. The storage elements hold the values of the present state of the system. The next state (NS) is determined by a combination of the present state (PS) and the input values; and the system outputs are determined by a combinational circuit which takes the present state of the system as input.

There are two categories of sequential circuits, *synchronous* and *asynchronous* sequential circuits. Synchronous sequential circuits are used most often in digital design as the behaviour is most predictable and

reliable. Synchronous sequential circuits can only change their states based on their inputs at discrete instances in time and this synchronisation is achieved using a *clock generator* which produces a series of *clock pulses* [10]. A clock cycle is a logic signal that changes state from HIGH to LOW at a particular frequency. The duration of the HIGH pulse (t_H) is the same as the LOW pulse (t_L) and the output therefore looks like a square wave. The frequency of the clock can be calculated as follows:

$$\begin{aligned} f &= \frac{1}{(t_H + t_L)} \\ &= \frac{1}{T} \text{ [hertz]} \end{aligned} \tag{4A.1}$$

Where f is the frequency of the clock signal in hertz and T is the period of a clock cycle in seconds.

An asynchronous circuit, on the other hand, can change its state at any time based on the input values.

4A.3 Sequential Circuit Design

A synchronous sequential circuit is able to store its present state and use this information together with the input signals to determine the next state for the system. A synchronous sequential circuit is only able to change states at the start of the next clock cycle.

A D flip-flop is the simplest sequential circuit element. It has one input (D), one output (Q) and two possible states (0 or 1). The *CLK* signal is used to synchronise the state transitions [7]. The present state of the flip-flop is the current logical value of Q while the next state is determined by the logical value on D at the start of the next rising edge of the clock.

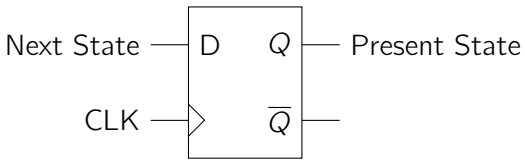


Figure 4A.2: D flip-flop as synchronous sequential circuit element. The present state of the device is held by the Q output and the next state is determined by the logical value of the D input at the start of the next rising edge of the clock.

We can therefore use D flip-flops to hold the present system's state and a combination of these state values and the sequential circuit's input signals to change the logical values on the D inputs at the start of the next clock cycle. The sequential circuit outputs are determined by what state the system is presently in and will be a logical combination of the present state values of the system. We now consider the techniques we can use to design this type of circuit.

4A.3.1 Present State/Next State Diagrams

A *Present State/Next State* diagram is a tool which can be used to analyse the state transition behaviour of a synchronous sequential circuit. We can then use this table to design the combinational circuit that will be used to calculate the logical values present on the D inputs at the start of the next rising edge of the clock. It is drawn in a similar manner to a truth table except that each row represents the system inputs, the present state values, the next state values and 'time'. The inputs and present state columns represent all possible combinations of input conditions and states, while the next state columns show what the values on the D inputs should be at the start of the next clock cycle. Once the table has been

completed, we can use standard logic reduction techniques to design the combinational circuits. Each row in this table represents a full clock cycle and it is always important to consider the inherent timing in this diagram.

4A.4 Algorithmic State Machine Design

Some systems are simply too complex to analyse using present state/next state tables. A complex synchronous sequential circuit might have many states and numerous inputs, not all of which are relevant in all possible states. PS/NS charts start to become clumsy and tedious to use in these cases and therefore we need alternative approaches to the representation and design of these systems. One approach is known as an *Algorithmic State Machine* chart or ASM chart. An ASM chart looks very similar to a standard flow chart used to describe a software programme in computer science. However it differs in one important way, timing is an inherent part of this representation and it can therefore be used to describe synchronous sequential logic circuits. It not only describes the way the system changes from one state to the next but also at what point in time they change. This provides us with a powerful graphical approach for design sequential circuits. We will now describe each of the graphical blocks in the ASM chart.

4A.4.1 State Box

Each state in a synchronous sequential system can be represented as a *state box* in an ASM chart. A state box is drawn as a rectangular box, with a single input (state entry path) and a single output (state exit path). Every state box has a unique binary *state code* and often has a state name which describes the state in some way. Every state box represents a full clock cycle and transitions into a state box occur on the rising edge of the clock cycle and exits out of the state box occur on the next rising edge of the clock. All state outputs are listed inside the box and will be active for a single clock cycle. Therefore we can see how timing is inherent in this type of representation.

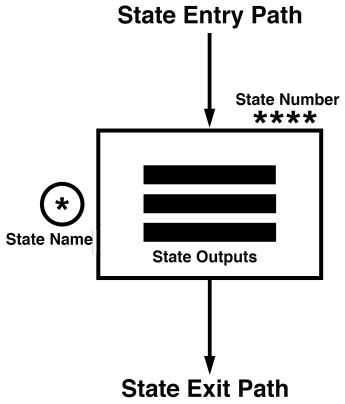


Figure 4A.3: A state box

4A.4.2 Decision Box

A decision box is used to select between two possible state output paths based on some logical input or control condition. It is not a state box and is only active in the transition between clock cycles or on the rising edge of the clock. A decision box has a single input path and two possible exit paths, one if the condition is TRUE and the other if the condition is FALSE. It examines the condition of a single variable or input and only one exit path can be active during a state transition.

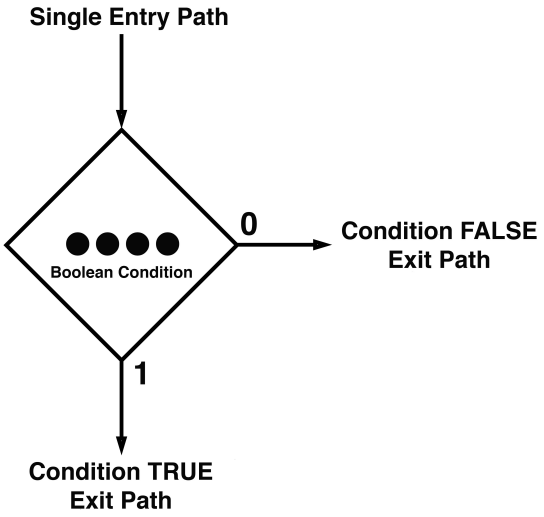


Figure 4A.4: A decision box

4A.4.3 Conditional Output Box

A conditional output box contains output values that are dependent on both the current state and the current input condition. They therefore differ to state outputs which are only dependent on the current state. They are represented in an ASM chart as a rectangular block with rounded corners and only have a single entry and exit path. The outputs of a conditional output box are asserted in the transition between states and therefore can be of variable length in time. They should not be used when the length of timing of the output's assertion is critical.

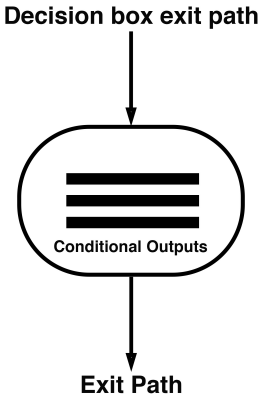


Figure 4A.5: A conditional output box

The behaviour of the system is modelled by combining these three elements in a particular order which describes how the system moves from one state to another.

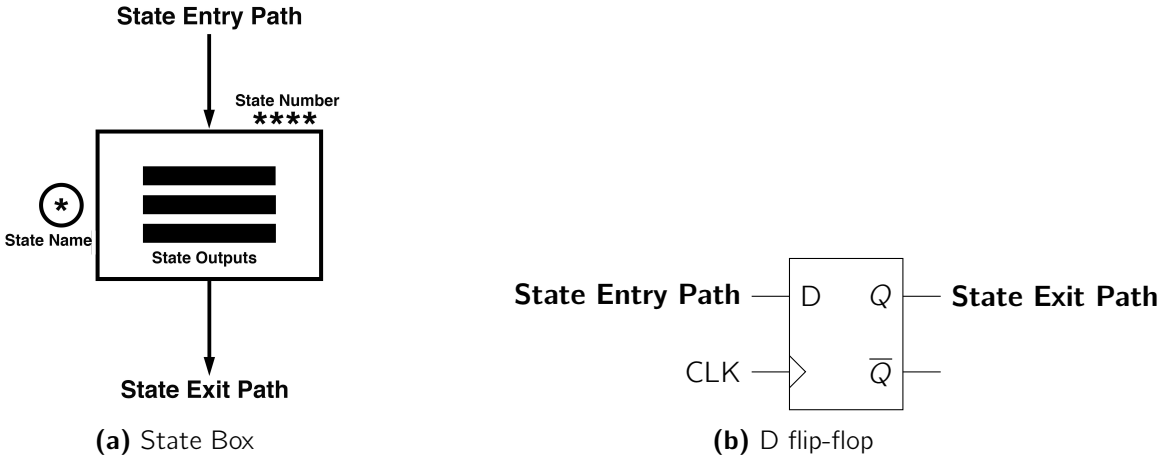
4A.4.4 Designing with One flip-flop per state

Up to this point we have assumed that it is optimal to have the minimum number of flip-flops in a synchronous sequential design. However there is a trade off between the simplicity of the resulting circuit and the ease of generating this design from the ASM chart. However there is an alternative design approach

known as *one flip-flop per state*, which automates this process. In this design scheme, each element in the ASM chart has an equivalent circuit element, so one can simply replace one ASM element for the equivalent circuit element using the following rules:

State box

In this scheme, a state box is replaced by a single D flip-flop. The entry path into the state is modelled by the *D* input into the flip-flop, while the exit path from the state is modelled by the *Q* output from the flip-flop. Only one flip-flop is active at any given time, which also makes the behaviour of this circuit more predictable.

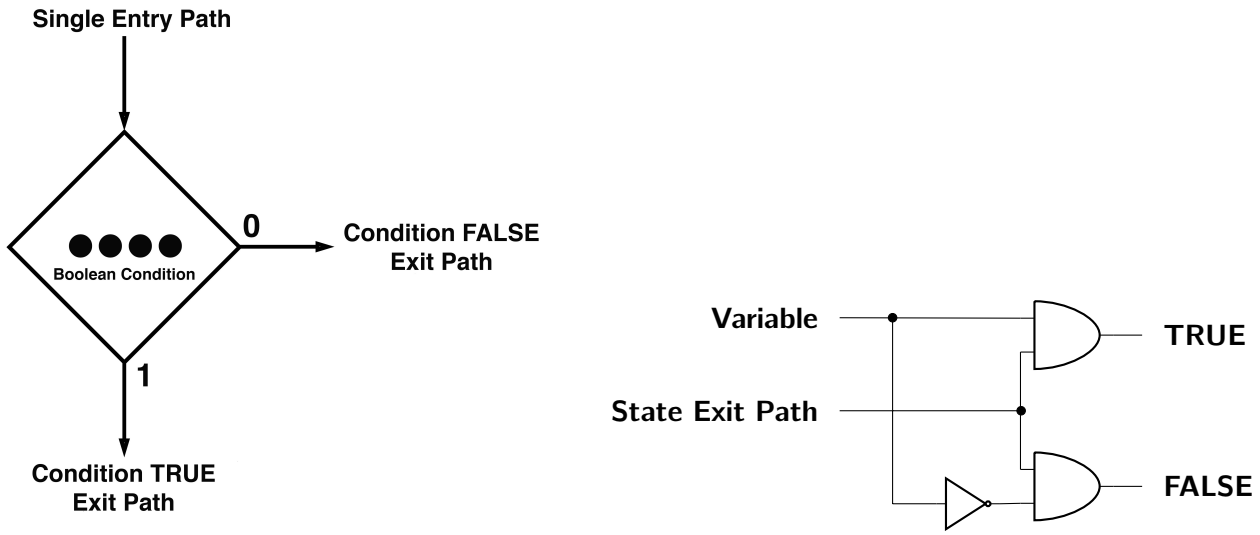


(a) State Box **(b)** D flip-flop

Figure 4A.6: The state box equivalent circuit. A state box can be replaced by a single D flip-flop when designing using one flip-flop per state.

Decision box

A decision box is replaced by two AND gates and a NOT gate. The input of the one AND gate is connected to the state input or variable, while the other AND gate is connected to the NOT of this signal. The other two inputs of the AND gates are connected together and linked to the output of a state box or the *Q* of the D flip-flop. This signal acts as an ENABLE, which means that the decision box circuit is only active when the output of the flip-flop is HIGH.

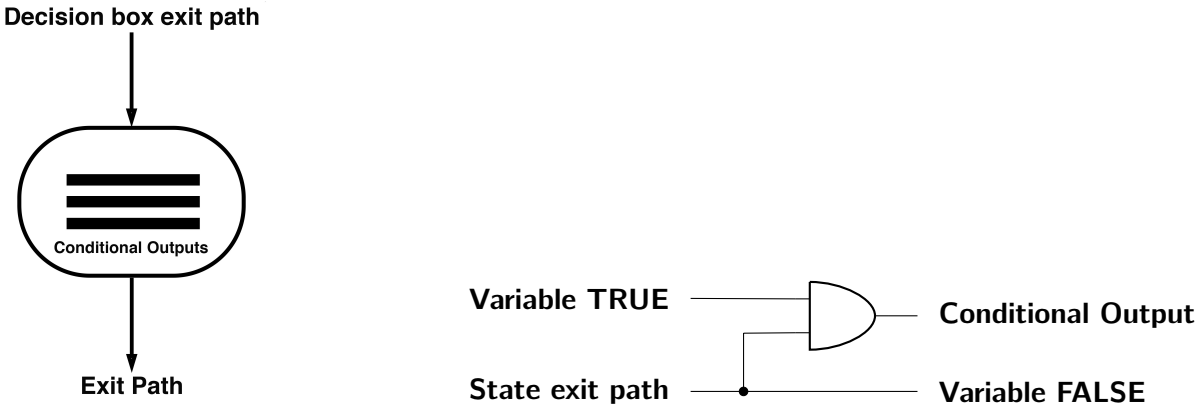


(a) Decision Box **(b)** Decision box circuit

Figure 4A.7: The decision box equivalent circuit. A decision box is replaced by two AND gates which are connected together. The output from the one AND gate is the TRUE path, while the output from the other gate is the FALSE path. The gates have the decision variable as one input and the state exit path as the other. The state exit path acts as an ENABLE signal.

Conditional Output box

The conditional output box will produce a HIGH output during the transitions between states. In other words when a variable is TRUE and we are exiting a state. This can be achieved using a single AND gate, where one input is the variable or condition and the other input is the state exit path. The output will be HIGH if both inputs are TRUE. However we also need a path of the condition is FALSE and this achieved by bypassing the AND gate.



(a) Conditional Output Box **(b)** Conditional Output circuit

Figure 4A.8: The conditional output box equivalent circuit. A conditional output box produces a HIGH when the input variable is TRUE and the state exit path is HIGH, otherwise the conditional output will be FALSE.

Junctions

Lastly we need to ensure that there is never an illegal value state (X) on any of the lines. This can occur if there is a junction or connection between outputs. As was mentioned in section **??**, a contention on a logic line can occur when one output attempts to pull the line HIGH while another output, connected to the same line, is pulling the line LOW. We can correct this by using an OR gate as a replacement for all

junction points in the circuit.

We will now consider how to use these techniques in sequential logic design.

4A.5 Task

In the next section we will design and simulate two sequential circuits.

Read through each task carefully and then complete the system design, construction and question section. Remember to draw your circuits neatly.

4A.6 Questions and circuit simulation

Design the sequential circuits for the questions using the appropriate design technique. Minimise the number of memory elements in your system and before you begin identify all the inputs and outputs in your system.

Part 1
Design a binary counter using rising edge triggered D-type flip-flops and combinational logic gates (AND, OR, NOT, XOR, NAND, NOR etc.), which produces the following output sequences. It must not be a ripple-type counter.

input = 0: $2 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow$**repeat**

input = 1: $2 \rightarrow 8 \rightarrow 4 \rightarrow 6 \rightarrow$**repeat**

(a) Draw the present state/next state diagram for this counter in your report. Minimise the number of sequential elements (flip-flops) in your system. **[5]**

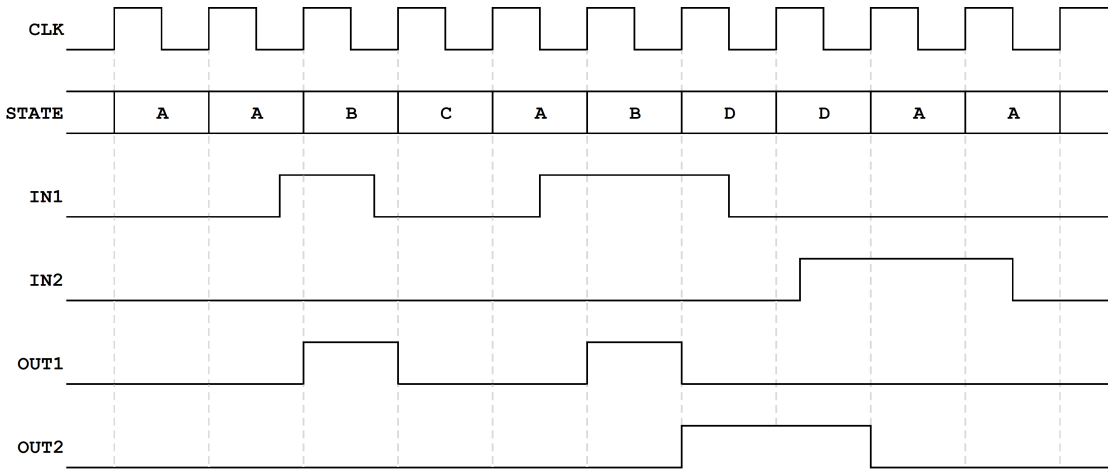
(b) Find simplified output logic expressions for the sequential and output circuits in your system. You may use Karnaugh Maps, Boolean algebra or simple inspection to determine them. **[6]**

(c) Draw the final circuit diagram for this system based on your equations from part (b) in your report. **[6]**

(d) Simulate the circuit in Logisim and save a screenshot of your circuit diagram in your report. **[5]**

Part 2

Consider the following timing diagram of an unknown synchronous sequential circuit built using rising edge triggered D-type flip-flops.



(a) Draw an ASM chart for this system in your report. **[7]**

(b) Using the ASM chart, draw the circuit diagram for the system using the "one flip-flop per state" technique in your report. **[6]**

(c) Simulate the circuit in Logisim and save a screenshot of your circuit diagram in your report. **[5]**

4A.7 Practical Submission

Submit your completed practical report on VULA as a **.pdf**[1] under the correct assignment. Show all your calculations and either take a photo of or scan your circuit diagrams and include them in your report.

Your document must be named as follows so that it is easily identifiable:

Prac4A-STUDENTNUMBER.pdf
Prac4APart1-STUDENTNUMBER.circ
Prac4APart2-STUDENTNUMBER.circ

Do **NOT** zip the files together but upload them individually to VULA. Ensure that your name and student number is clearly written in your practical report. Practical reports must be completed and submitted on VULA by 23h55 by the due date on the VULA calendar.

[1] **Note** if it is submitted as a **.doc/.docx** or equivalent format file, it will **NOT** be marked.

4A.8 Marks Breakdown

Questions	**Marks**
Part A (a)	5
(b)	6
(c)	6
Circuit	5
Part B (a)	7
(b)	6
Circuit	5
Total	**40 Marks**

Up to 5 marks will be deducted for untidy reports. **5% will be deducted per day for late hand in's for up to one week, after which you will receive 0 and the practical report will no longer be accepted.** Please ensure that you submit the files in the correct place and read the instructions given on VULA with regards to file naming. **NO EMAILED** pracs will be accepted.