PROJECT 4
FINITE STATE MACHINES

Deadline: 23:55 on Friday 28th January, 2022

Last updated*: 15:50 on Saturday 7th January, 2023

Instructions

(1) The project must be implemented and submitted individually.
(2) Submissions in pairs or groups are not allowed. Plagiarism will not be tolerated.
(3) Submit exactly 1 Logisim (".circ") file under the appropriate submission box on Moodle.
The file must be named `ID_fsm.circ`, with `ID` replaced by your 9 digit ID number.
(4) Use the provided `template_fsm.circ` file as a template, and implement your designs in the respective circuits. *Do not* move or modify the input/output ports, the "blackbox" layout, and the names of the circuits!
(5) If you need to use a constant 0 or 1 as a component in your circuits, use the "Constant" component from the "Wiring" library.
(6) Use tunnels to keep your circuit clean and free of spaghetti wiring. Do not remove the tunnels provided in the template. Use tunnels with matching names to connect the inputs/outputs to your circuit. Note: There is no limit on the number of tunnels corresponding to a signal. As long as they have the same name, any number of tunnels can be used for a single signal.
(7) All submissions will be graded using an automated grading system. There will be *no manual grading*, and the grade you receive will be based *exclusively* on the functionality of your circuits. No credit will be given purely for "attempted" solutions.
(8) In order to ensure that your file is compatible with the automated grading system, and to get a preliminary evaluation of the functionality, you are provided access to a validation system. In order to use this validation system, you need to send an email to `DLSBodek@gmail.com` with your Logisim (".circ") file attached.
(9) If your file contains compatibility issues and/or errors, the validation system will send you a reply with a list of error which you need to fix.
(10) If your file contains no compatibility issues and errors, the validation system will run a small number of tests on your circuits, and will return the percentage of correct outputs observed. Note that this

*This file may be updated once the project has been released, in order to fix mistakes and add clarifications. It is recommended to always download and use the latest file.

validation system checks only a small fraction of the possible inputs to your circuit. If all of these outputs are as expected, it means that your implementation may or may not be correct. However, if some of these outputs are not as expected, it means that your implementation is certainly not correct. In other words, if the preliminary score given by the validation system is less than 100 %, you still need to fix your implementation; but a preliminary score of 100 % does not guarantee a final grade of 100 %.

(11) The project consists of a number of exercises, with grades which sum up to 1 points. The maximal grade for each exercise is mentioned at the beginning of the exercise. Your final grade will be the grade you receive (out of 1), expressed as a percentage.

(12) Before submitting the file to Moodle, make sure you have validated it, and have received a "No compatibility issues detected." remark. In addition, make sure that your preliminary score is as high as possible.

(13) You *may not* use gates with fan-in[†] larger than 2. Splitters and multiplexers with any fan-in are acceptable.

(14) Unless specified otherwise, you may use components only from the following libraries provided by Logisim:
   (a) Wiring
   (b) Gates (basic gates only[‡])
   (c) Base
   (d) Arithmetic (shifters only)
   (e) Memory (D flip-flops only)
   (f) Plexers (multiplexers only)

(15) Unless specified otherwise, your implementation of a circuit in Exercise m of Project n may use circuits you designed in Exercises $m - i$ in Projects $n - j$, for all $0 < i \leq m$, $0 \leq j \leq n$.

[†]The fan-in of a gate g is the number of input terminals of g, i.e., the number of bits in the domain of the Boolean function that specifies the functionality of g.

[‡]Basic gates include NOT, AND, OR, NAND, NOR, XOR, and NXOR. Parity gates, controlled buffers, and controlled inverters do not fall under basic gates.

1. Outline

This project deals with the synthesis of a finite state machine. You will implement the logic of a simple vending machine, modelled as a Mealy machine[§] The implementation should be done using a parallel-load register (or equivalently multiple D flip-flops) and two combinational circuits, of which one computes the next state and another computes the outputs.

The vending machine to be implemented is a overly simplified (and quite impractical) one. The machine can dispense two products: p_0 which costs 12 shekels, and p_1 which costs 5 shekels. The machine can accept coins of 5 and 10 shekels only, and can return any amount of change as required. In every clock cycle, the user of the machine is required to input the product they wish to buy and insert either a 5 shekel coin or a 10 shekel coin.

Note that the user may change their choice of product when they are in the process of inserting money. The last choice made by the user *before* they insert sufficient money should be considered to be the final choice. The table below shows an example of such a scenario in action.

In addition, the vending machine does not support cancellation of an order, nor does it support insertion of zero money.

2. Modelling

Consider the following encoding of the states, inputs, and outputs of the FSM.

Inputs: $p(t) \in \{0,1\}$ which represents the choice of the product, $m(t) \in \{0,1\}$ which represents the amount of money inserted, and a clock CLK.

$$p(t) = \begin{cases} 0 & ; \quad p_0 \text{ (with a price of 12 shekels) selected} \\ 1 & ; \quad p_1 \text{ (with a price of 5 shekels) selected} \end{cases}$$

$$m(t) = \begin{cases} 0 & ; \quad 5 \text{ shekels inserted} \\ 1 & ; \quad 10 \text{ shekels inserted} \end{cases}$$

For the sake of notational convenience, let $\widetilde{m}(t)$ represent the amount of money encoded by $m(t)$. Therefore,

$$\widetilde{m} = \begin{cases} 5 & ; \quad m = 0 \\ 10 & ; \quad m = 1 \end{cases}$$

States: $S(t) \in \{0,1\}^2$ which encodes the state of the FSM, based on the balance currently available in the machine.

$$S(t) = \begin{cases} 00 & ; \quad 0 \text{ shekels balance in machine} \\ 01 & ; \quad 5 \text{ shekels balance in machine} \\ 10 & ; \quad 10 \text{ shekels balance in machine} \end{cases}$$

[§]A Mealy machine is a finite state machine whose outputs depend on the inputs and the state at a given clock cycle.

For the sake of notational convenience, let $\widetilde{S}(t)$ represent the state encoded by $S(t)$. Therefore,

$$\widetilde{S}(t) = \begin{cases} q_0 & ; \quad S(t) = 00 \\ q_5 & ; \quad S(t) = 01 \\ q_{10} & ; \quad S(t) = 10 \end{cases}$$

Consistent with the standard notation the standard notation $NS(t)$ represents the state that FSM will have in the next clock cycle, i.e.

$$S(t+1) = NS(t)$$

As with $S(t)$ and $\widetilde{S}(t)$, $\widetilde{NS}(t)$ represents the state encoded by $NS(t)$.

Outputs: $d(t) \in \{0,1\}$ which represents whether the product was delivered, and $c(t) \in \{0,1\}^4$ which represents the amount of change returned.

$$d(t) = \begin{cases} 0 & ; \quad \text{product not delivered (not enough money inserted)} \\ 1 & ; \quad \text{product delivered (enough money inserted)} \end{cases}$$

$\langle c(t) \rangle$, i.e. the number represented by $c(t)$ in standard binary representation equals the amount of change to be returned.

The machine delivers the product and returns change as soon as sufficient balance is reached. For example, if at $t = i - 1$ the balance is zero, and at $t = i$ the user inserts 10 shekels and selects p_1, then the machine delivers the product and the change at $t = i$. Equivalently, if

$$\begin{aligned} \widetilde{S}(i-1) &= q_0 \\ p(i) &= 1 \\ \widetilde{m}(i) &= 10 \end{aligned}$$

then

$$\begin{aligned} d(i) &= 1 \\ c(i) &= 0101 \end{aligned}$$

Preliminary Task 1. *Not for submission*
Draw the state diagram for the FSM, using the above encoding. What is the initial state of the FSM? Are all of the states reachable?

Preliminary Task 2. *Not for submission*
What values of $c(t)$ are possible outputs? Equivalently, what amounts of change can the machine return?

Preliminary Task 3. *Not for submission*
Complete the following truth table¶

$\widetilde{S}$	$S[1]$	$S[0]$	p	$\widetilde{m}$	m	$\widetilde{NS}$	$NS[1]$	$NS[0]$	d	$\langle c \rangle$	$c[3]$	$c[2]$	$c[1]$	$c[0]$
q_0			0	5										
q_0			0	10										
q_0			1	5										
q_0			1	10										
q_5			0	5										
q_5			0	10										
q_5			1	5										
q_5			1	10										
q_{10}			0	5										
q_{10}			0	10										
q_{10}			1	5										
q_{10}			1	10										

Table 1

3. Example

t	$p(t)$	$\widetilde{m}(t)$	$m(t)$	$d(t)$	$\langle c(t) \rangle$	$c(t)$
0	1	0	5	1	0	0000
1	1	1	10	1	5	0101
2	0	0	5	0	0	0000
3	1	1	10	1	10	1010
4	0	0	5	0	0	0000
5	0	0	5	0	0	0000
6	0	1	10	1	8	1000
7	0	0	5	0	0	0000
8	0	0	5	0	0	0000
9	1	1	10	1	15	1111
10	0	1	10	0	0	0000
11	1	0	5	1	10	1010

Table 2. Example of Operation of FSM

Table 2 shows an example of the operation of the FSM.

¶All signals in Table 1 are functions of time, however the "(t)" is omitted here for compactness.

At $t = 0$, the user selects p_1 ($p(0) = 1$) which costs 5 shekels, and inserts 5 shekels ($m(0) = 0$). As the total balance in the vending machine is equal to the price of the product, the product is delivered ($d(0) = 1$) and no change is returned ($c(0) = 0000$).

Following this transaction, the remaining balance in the machine is zero, and the FSM returns to its initial state. At $t = 1$, the user again selects p_1 which costs 5 shekels, and inserts 10 shekels. As the balance in the vending machine is greater than the price of the product, the product is delivered, and the appropriate change of 5 shekels is returned. Then, at $t = 2$, the user selects p_0 which costs 12 shekels, and inserts 5 shekels. As the balance in the machine is less than the cost of the product, no item is delivered, and the machine waits for the user to insert more money. At $t = 3$, the user inserts 10 shekels, and also changes their selection to p_1 which costs 5 shekels. The machine considers only the most recent product selection as the "real" selection, and hence delivers p_1 with a change of 10 shekels.

At $t = 4$, the user selects p_0 which costs 12 shekels, and inserts 5 shekels. As the balance in the machine is less than the cost of the product, no item is delivered, and the machine waits for the user to insert more money. At $t = 5$, the user inserts 5 more shekels and retains their selection of p_0. As the balance available in the machine (10 shekels) is less than the cost of p_0 (12 shekels), no item is delivered, and the machine waits for the user to insert more money. At $t = 6$, the user inserts 10 more shekels, and retains their selection of p_0. Now, as the balance available in the machine (20 shekels) is greater than the cost of p_0 (12 shekels), the product is delivered, and the appropriate change of 8 shekels is returned.

At $t = 7$, the user selects p_0 which costs 12 shekels, and inserts 5 shekels. As the balance in the machine is less than the cost of the product, no item is delivered, and the machine waits for the user to insert more money. At $t = 8$, the user inserts 5 more shekels and retains their selection of p_0. As the balance available in the machine (10 shekels) is less than the cost of p_0 (12 shekels), no item is delivered, and the machine waits for the user to insert more money. At $t = 9$, the user inserts 10 more shekels, but changes their selection to p_1 which costs 5 shekels. The machine considers only the most recent product selection as the "real" selection, and hence delivers p_1 with a change of 15 shekels.

At $t = 10$, the user selects p_0 which costs 12 shekels, and inserts 10 shekels. As the balance in the machine is less than the cost of the product, no item is delivered, and the machine waits for the user to insert more money. At $t = 11$, the user inserts 5 more shekels and changes their selection to p_1 which costs 5 shekels. The machine then delivers p_1 with a change of 10 shekels.

4. Implementation

Exercise 1. 1 P.

Complete the circuit `fsm` from the given template to implement the above described FSM of a simplified vending machine. You may create and use any auxiliary circuits you require, as long as these circuits are implemented in the file you submit, and are not "imported" from other files. You may also

use any components provided by Logisim, *except arithmetic components, i.e. adders, subtracters, etc.*.

In addition to the I/O ports, the circuit `fsm` also contains an instance of the circuit `debugger`, which is intended as a helper circuit to verify the functionality of your circuit. This circuit converts the inputs, outputs, and states of the FSM into more human-friendly forms. Note that this debugger will function properly only when the required signals are generated properly and are connected to appropriately labelled tunnels.