ELEC3342: Digital System Designs
First Semester, 2021–22

Homework 1 (r1.1)

Due:

- Part (A) -- 10 Oct, 2021, 11:59pm
- Part (B) -- 10 Oct, 2021, 11:59pm

Instruction: Submit your answers electronically through Moodle.

There are 2 major parts in this homework. Part A includes questions that aim to help you with understanding the lecture materials. They resemble the kind of questions you will encounter in quizzes and the final exam. Some of these questions may also ask you to implement and test small designs in VHDL. This part of homework must be completed individually.

Part B of this homework contains a mini-project that you should work in groups of 2. Your submitted work will be graded by an auto tester and therefore you should make sure your submitted files conform to the required format.

The following summarize the 2 parts.

Part	Type	Indv/Grp
A	Basic problem set	Individual
B	Mini-project	Group of 2

In all cases, you are encouraged to discuss the homework problems offline or online using Piazza. However, you should not ask for or give out solution directly as that defeat the idea of having homework exercise. Giving out answers or copying answers directly will likely constitute an act of plagiarism, which is a serious offense.

Part A: Problem Set

A.1

A.1.1 For each of the truth table in Figure A.1, do the following:

(i) Draw the corresponding K-map

(ii) Minimize using K-map and write the resulting Boolean equation in canonical sum-of-products form;

(iii) Design a simple circuit corresponding to the truth table. You may use any gate with any number of inputs.

A.1.2 For each of the truth table in Figure A.1, implement your resulting circuit with VHDL. Your submitted VHDL must meet the following requirements:

- The entity of your circuit should be named as:

  `A_1_<truth_table_number>`

- Each circuit should have a top-level VHDL entity with the corresponding port signal.
- All ports must be of type `STD_LOGIC`
- Save your vhdl file as:

  `A_1_<truth_table_number>.vhdl`

Submit your VHDL files.

(1.)

A	B	Y
0	0	1
0	1	0
1	0	1
1	1	1

(2.)

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(3.)

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

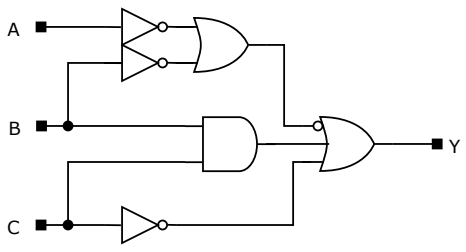
(4.)

A	B	C	D	E	Y
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	1
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

Figure A.1: Truth Tables for questions Question A.1.1 and Question A.1.2

A.2

Using De Morgan equivalent gates and bubble pushing methods, redraw the circuit in Figure **??** so that there is only NAND or NOT gates with optional bubble input.



A.3

In class, we described a two-input XOR gate as gate that outputs TRUE if exactly 1 of its input is TRUE. Unfortunately, this description does not generalize directly to a 3 or more input XOR gate. In particular, consider the following 3-input function:

$$Y = A \oplus B \oplus C$$

A.3.1 Draw the truth table of the above function. When is the output Y TRUE?

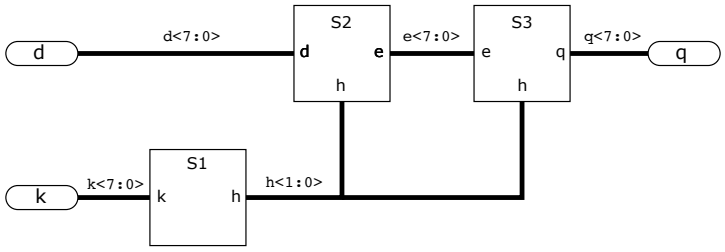
A.3.2 Now, draw the truth table of the following 4-input function. When is the output Y TRUE in this case?

$$Y = A \oplus B \oplus C \oplus D$$

A.3.3 Design a 4-input circuit that will output TRUE only if *exactly 1* input is TRUE.

A.4 Encrypting with Hardware

Adopted from Fall 2020 final exam You are designing a high speed device to encrypt text messages in hardware. Your hardware encryptor receives a stream of 8-bit ASCII data as input (`d<7:0>`) and produces an encrypted data stream (`q<7:0>`). The following diagram shows a top-level block diagram of your encryptor:



In words, there are 3 major steps involved in encrypting each byte of data:

1. Start with a secret key k, a hashed value h is produced using the function `S1`. h is a 2-bit value (`h<1:0>`).
2. Perform a function (`S2`) to produce intermediate value e based on d and h, i.e., $e = \texttt{S2}(d, h)$.
3. Perform a *left rotation* (i.e., the `S3` block) on e by an amount h to obtain q.

A.4.1 Design a circuit for the function `S1`. The function `S1` takes as input an 8-bit value `k<7:0>` and produces a 2-bit output value `h<1:0>` using the following function:

$$h = (\text{number of '0' in } \texttt{k<7:0>}) \textbf{ mod } 4$$

In other words, to obtain the value of h, you count the number of '0's in the 8-bit input k. Then you divide the result by 4, taking only the remainder. For example, if `k<7:0>` = `10100001`, then $h = 01$.

You may use gates and muxes with any number of inputs, and you may use 2-input adders with any input data width. Label the width of any bus signal on the circuit diagram.

A.4.2 Design a circuit for the function `S3`. It takes as input an 8-bit value `e<7:0>` and produces output `q<7:0>` by *left rotating* the bits in e by h. Recall that h is a 2-bit value so the maximum amount of rotation is 3. For example:

$$h = 3,\ e = \texttt{10010111} \xrightarrow{\texttt{S3}} \texttt{10111100}$$

Design the circuit for `S3` below. You may use gates and muxes with any number of inputs, and you may use 2-input adders with any input data width. To avoid drawing too many wires, use labels on a wire instead. Label the width of all bus signals.

A.4.3 In an attempt to improve the efficiency of your encryptor circuit, you realize that if the key is *fixed*, then the entire circuit can be *customized* for that particular fixed key and become smaller. This process is called *constant propagation*. For example, if $Y = \overline{AB}$ is an NAND gate, and you know that A is fixed at '1', then the circuit can be simplified as $Y = \overline{B}$. In the rest of this question, you will perform simplification to the encryptor using constant propagation based on a constant key k. The key you will use is the ASCII code of the **LAST DIGIT** of your **university number**. Use the following ASCII code table for your reference:

Chatercter	ASCII Code (dec)	ASCII Code (hex)
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A

With the last digital of your university number as key k, further transform it using the function `S1` into the 2-bit hash value `h<1:0>`. *Hint:* you can obtain h from k according to the definition of `S1` without worrying about the underlying circuit of `S1`.

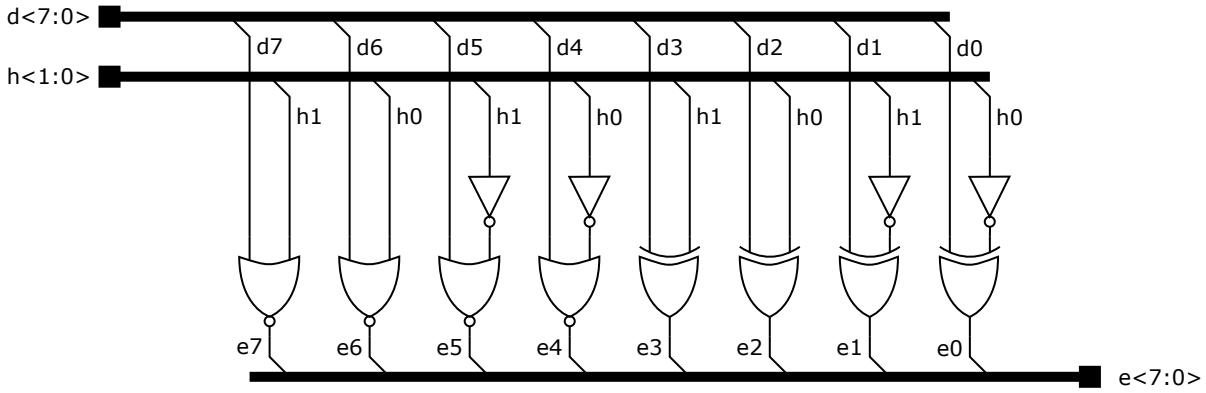
In the space below, write down: (i) the last digit of your university number; (ii) its corresponding ASCII code in *binary*, and (iii) it's corresponding hash value `h<1:0>`.

Last digit of your university number:

ASCII code of the last digit of your university number in binary (`k<7:0>`):

Value of corresponding `h<1:0>`:

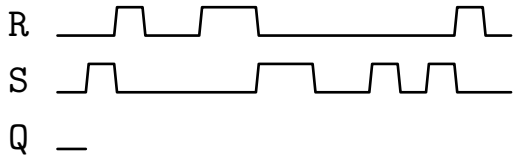
A.4.4 Optimize S2 Using the constant hash value (`h<1:0>`) you found above, optimize the circuit `S2`. The original circuit is shown below.



Your optimized circuit, called `S2k`, should perform the same function as `S2` except there is no input `h<1:0>` because it has a constant value.

A.5

Given the input waveform shown below, sketch the output, Q, of an SR latch.

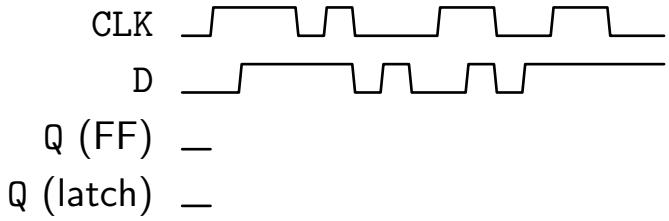


A.6

Given the input waveform below, sketch the output (Q), of :

(a) a D-latch

(b) a D-flip-flop

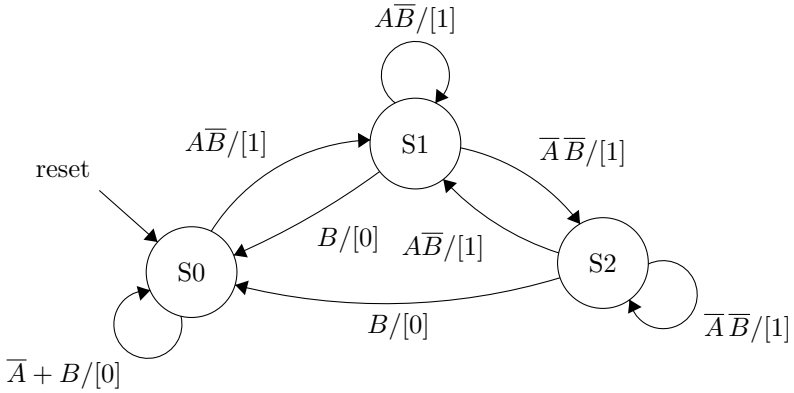


A.7

Describe in words what is the function of the following finite state machine. Is this a Mealy machine or a Moore machine?

Using binary state encodings, complete the next state logic and output logic table for the FSM. Write Boolean equations for the next stage and output logic.

Implement this FSM in vhdl using 3 separated processes, one for state registers, one for next state logic and one for output logic. Submit your FSM as `A_7_fsm.vhdl`.



Part B: Mini-Project

B.1 Morse Code Decoder

This semester you will be building a complete Morse Code decoding system. In this homework, you will begin your design with the core decoder finite state machine (FSM).

Morse Code Introduction Originally developed in the 19th century, Morse code has become one of the most widely used communication coding system even to this day. Morse code is a relatively simple system that uses a series of on-off signals to transmit text information (letters and numerals). These on-off signals can be transmit as telegram, sound, or even light.

The key differentiating factor is the duration of the signal. A long signal is called a 'dash', while a short signal is called a 'dot' in Morse code convention. Figure B.1 shows a chart of the standard international Morse code that you will adopt in this homework.

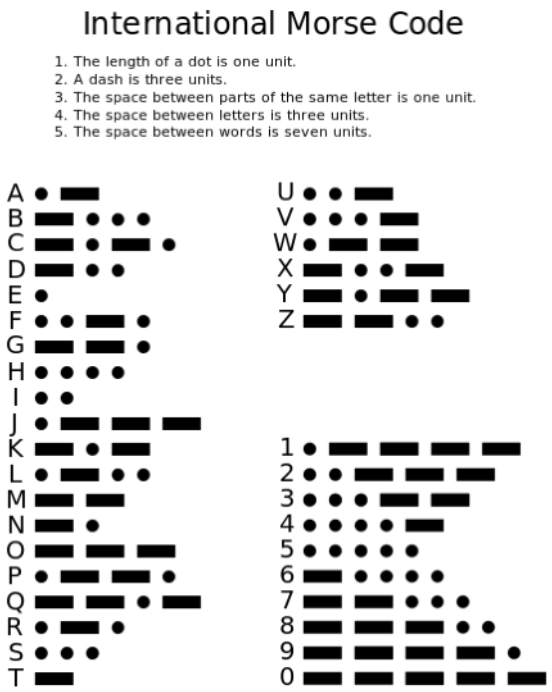


Figure B.1: Chart of International Morse Code

Furthermore, the duration of the time between a 'dot' and a 'dash' (a signal gap), the time between 2 letters (letter gap), and the duration between 2 words (word gap) are also important and are well

defined. Using the length of a 'dot' as a unit, the following table summaries the relative duration of each part of the code.

dot	1
dash	3
signal gap	1
letter gap	3
word gap	7

Although the relative duration of various parts of the code are well defined, the actual length of a 'dot' may vary. Furthermore, in practical applications, due to human error and other factors, the relative length of the signals and gaps may vary slightly as well.

B.1.1 Task 1: Morse Code Decoder Your main task for this homework is to develop the core morse code decoder of the system. Figure B.2 shows a block diagram of your decoder.

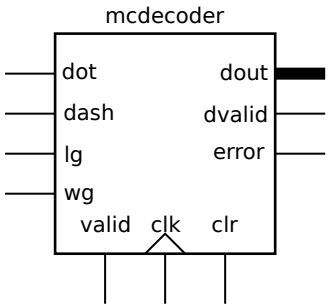


Figure B.2: Block diagram of your Morse code decoder

Your decoder has the following input and output port:

Port Name	Dir	Width	Description
`dot`	in	1	'1' when input is a dot, '0' otherwise
`dash`	in	1	'1' when input is a dash, '0' otherwise
`lg`	in	1	'1' when input is a letter gap, '0' otherwise
`wg`	in	1	'1' when input is a word gap, '0' otherwise
`valid`	in	1	When `valid` is '0' your decoder should ignore the input `dot`, `dash`, `lg` and `wg`. Your decoder should only respond to the above input when `valid` is '1'.
`clr`	in	1	When asserted ('1'), the entire decoder should reset to initial state regardless of the value of `clk`.
`clk`	in	1	The main clock to your module
`dout`	out	8	Detected symbol encoded as 8-bit ASCII code
`dvalid`	out	1	Asserted when `dout` is valid. '0' otherwise
`error`	out	1	Asserted for 1 cycle when an error in the input is detected (the `dot`s and `dash`es do not lead to a correct encoding). Your machine should restart and read the next `dot`/`dash` as the first symbol.

B.1.2 Example Run The waveform shown in Figure B.3 illustrates the expected behavior of your block when the input is the sequence "AT M" (A, T, space, M).

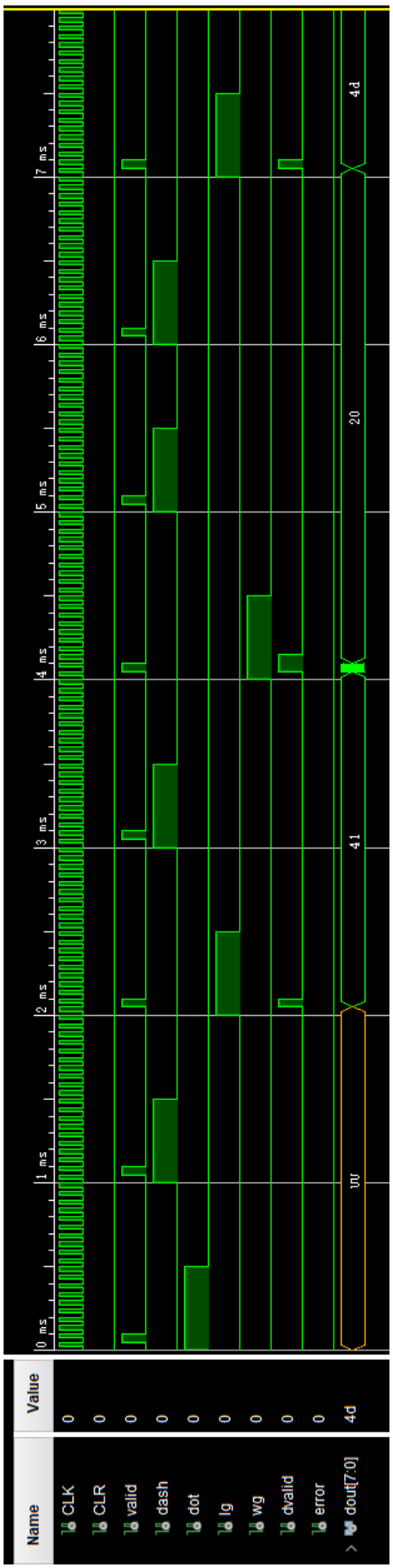


Figure B.3: Sample waveform when the input is the sequence A, T, space, M. Note that mcdecoder ignores its input when `valid` is low.

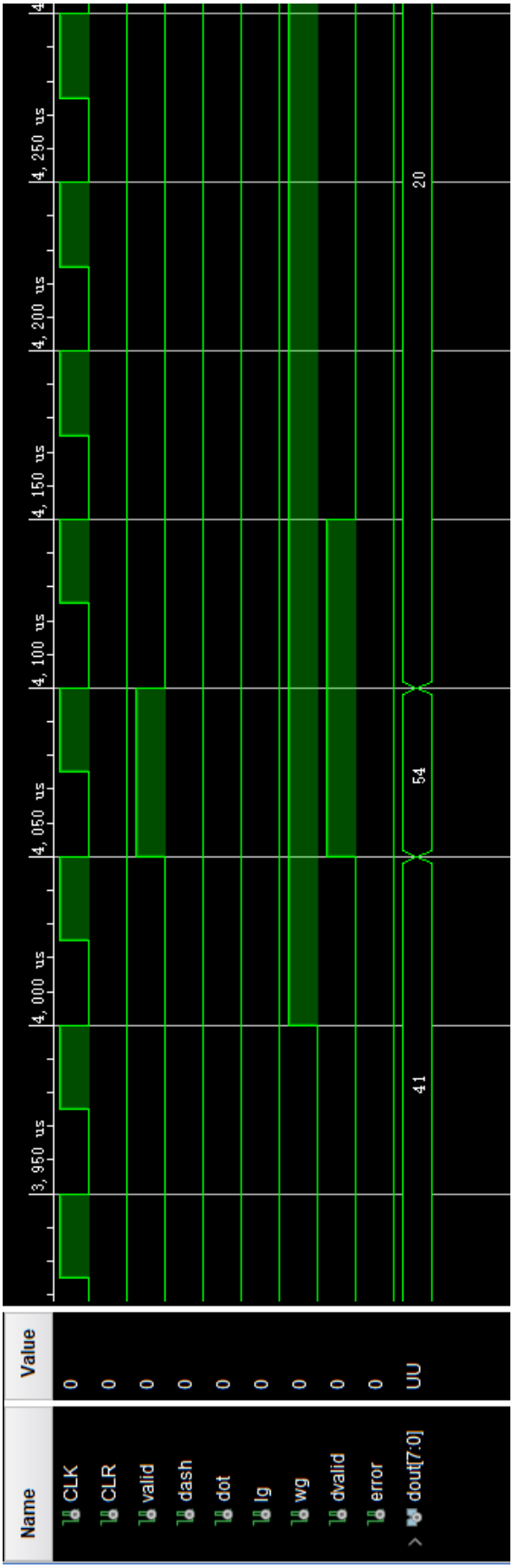


Figure B.4: Zoom into the area near T and space.

From the waveform, you will notice that your decoder reads in an input whenever `valid` is '1'. Once a `lg` or a `wg` is encountered, the correct symbol is determined from the previously received `dot`s and `dash`es.

Furthermore, note that if a `wg` is received, your decoder should also output an additional space symbol (ASCII code 0x20).

B.1.3 Submission For this part of homework, you should submit one file `mcdecoder.vhd` through Moodle. A template for that file is available on the course website, together with a sample testbench.