ELEC3342: Digital System Designs
First Semester, 2018–19

Homework 2 (r1.0)

Due:

- Part (A) -- 11 Nov, 2018, 11:59pm
- Part (B) -- 11 Nov, 2018, 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 Short Questions

A.1.1 Given the input waveform shown in Figure A.1, sketch the following:

1. Q output of a D-latch
2. Q output of a D-flip-flop

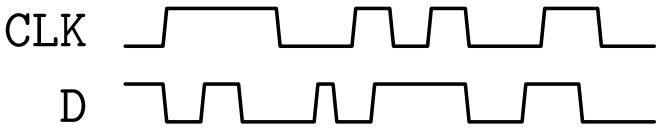


Figure A.1: Input waveform

A.1.2 Describe the function of the circuit shown in Figure A.2. Is it a sequential circuit or a combinational circuit?

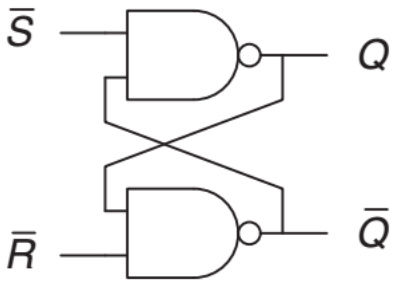


Figure A.2: Mytery Circuit

A.1.3 The circuit shown in Figure A.3 implements an FSM. Draw the state transition diagram of this FSM, and describe in words what is its function.

A.1.4 Registered Circuit Figure A.4 shows a circuit that computes a registered four-input XOR function. Each two-input XOR gate has a propagation delay of 100 ps and a contamination delay of 55 ps. Each flip-flop has a setup time of 60 ps, a hold time of 20 ps, a clock-to-Q maximum delay of 70 ps, and a clock-to-Q minimum delay of 50 ps. Determine the following:

(a) If there is no clock skew, what is the maximum operating frequency of the circuit?

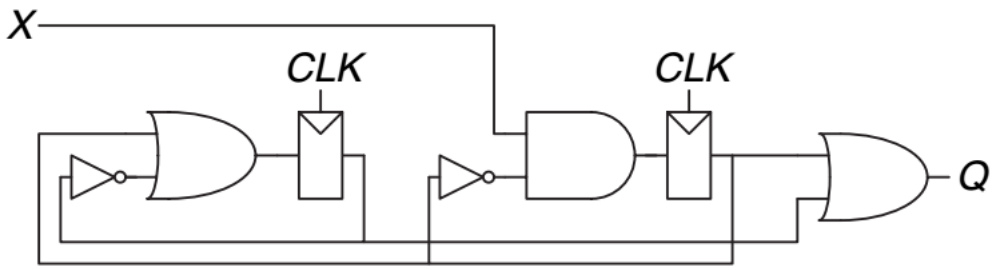


Figure A.3: Circuit for an FSM

(b) How much clock skew can the circuit tolerate if it must operate at 2 GHz?

(c) How much clock skew can the circuit tolerate before it might experience a hold time violation?

(d) Your project partner points out that she can redesign the combinational logic between the registers to be faster and tolerate more clock skew. Her improved circuit also uses three two-input XORs, but they are arranged differently. What is her circuit? What is its maximum frequency if there is no clock skew? How much clock skew can the circuit tolerate before it might experience a hold time violation?

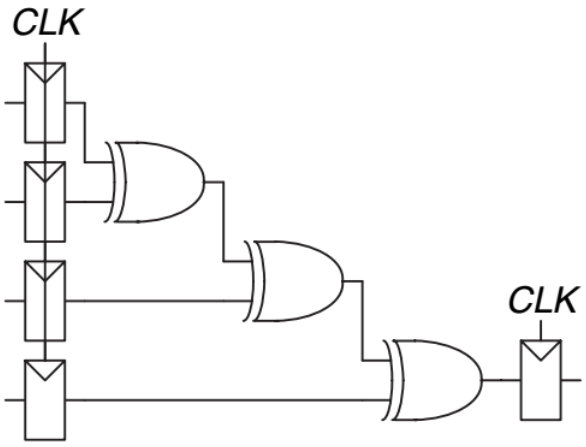


Figure A.4: A registered 4-input XOR gate

A.1.5 Design the following comparators for 32-bit signed integers. Sketch the schematics. You may reuse existing modules (adder, subtractor, etc), or build from the group up yourself.

(a) not equal

(b) greater than

(c) less than or equal to

A.1.6 Find the critical path for the 4×4 multiplier shown in Figure A.5 in terms of an AND gate delay (t_{AND}) and a full adder delay (t_{FA}). Also, in general, what is the delay of an $N \times N$ multiplier built in the same way?

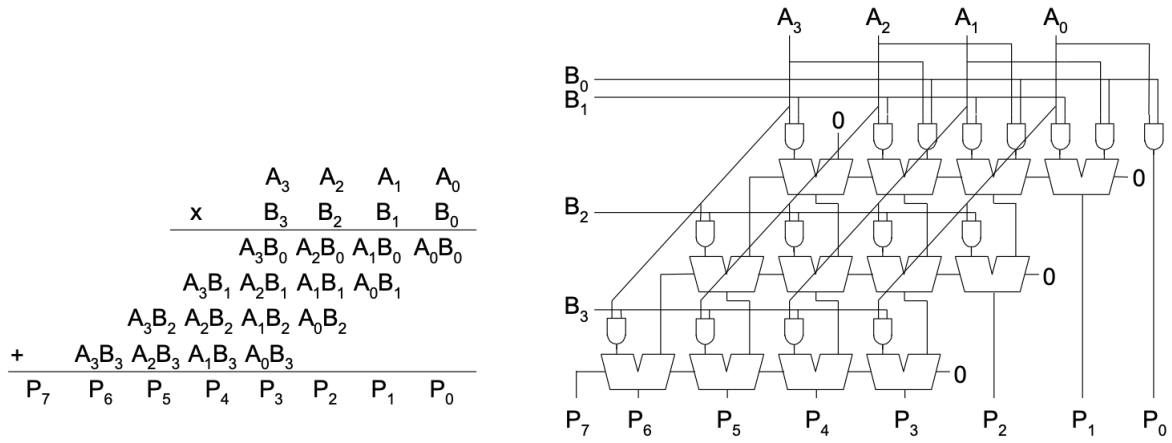


Figure A.5: 4×4 array multiplier

A.2 T-Flip-Flip Implementation

Implement the T-flip-flop shown in class in VHDL. In addition to the standard `T` and clock (`clk`) input, your T-flip-flop should also support an asynchronous clear (`clr`) as well as an enable (`en`) signal.

Submit your TFF VHDL file as `A_3_tff.vhd`.

A.3 Gray Code Counter

Number	Gray code		
0	0	0	0
1	0	0	1
2	0	1	1
3	0	1	0
4	1	1	0
5	1	1	1
6	1	0	1
7	1	0	0

Figure A.6: Gray Code counter

Gray codes have a useful property in that consecutive numbers differ in only a single bit position. Figure A.6 lists a 3-bit Gray code representing the numbers 0 to 7. Design a 3-bit modulo 8 Gray code counter FSM with no inputs and three outputs. (A modulo N counter counts from 0 to $N-1$, then repeats. For example, a watch uses a modulo 60 counter for the minutes and seconds that counts from

0 to 59.) When reset, the output should be `000`. On each clock edge, the output should advance to the next Gray code. After reaching `100`, it should repeat with `000`. *[DDCA 3.27]* Submit your VHDL file as `A_4_gray.vhd`. Also submit a short answer to this question describing your next state and output logic.

A.4 FIFO Design

Design a synchronous FIFO with the following features. Implement your design in VHDL to verify its function. Submit a top-level schematic of your design with a brief description. Also submit your VHDL file.

FIFO features that needs to be supported:

- 5-bit words
- FIFO depth: 10 positions
- To write, assert `push` at the same cycle as input data `din` for 1 cycle. If FIFO is full, the write is discarded.
- To read, assert `pop` for 1 cycle. The data in at the head of the FIFO is popped and is available in `dout` in the next cycle.
- A `full` signal is asserted on the cycle after the last write if the FIFO is full. It is asserted until data is popped.
- A `empty` signal is asserted on the cycle after the last data is popped or when right after reset.
- A `count` signal that always output the number of data in the FIFO.
- An asynchronous `clr` signal that clears all data and reset the FIFO to its initial empty state.
- A single clock signal `clk`.

Part B: Mini-Project

B.1 More Morse Code Decoder

In this homework, you will continue to extend the Morse code decoder that you have developed from homework 1 with 2 new modules. A system overview is shown in Figure **??**.

B.1.1 Symbol Detection The first module you need to design is the *symbol detection* module (`symdet`). `symdet` takes a stream of '0's and '1's that corresponds to the absence and presence of Morse code transmission. From this sequence, `symdet` determines if the input sequence corresponds to a valid Morse code symbol, i.e. "dot", "dash", "letter gap" or "word gap". If such symbol is detected, `symdet` outputs the corresponding signal to the `mcdecoder` module from homework 1. The following table shows a summary of the I/O ports of `symdet`.

Name	Direction	Description
d_bin	in	Binary digital input. A '1' represents the presence of transmission signal while '0' represents absence of signal (silence).
`dot`	out	'1' to indicate a *dot* is detected, '0' otherwise. Only valid when `valid` is asserted.
`dash`	out	'1' to indicate a *dash* is detected, '0' otherwise. Only valid when `valid` is asserted.
`lg`	out	'1' to indicate a *letter gap* is detected, '0' otherwise. Only valid when `valid` is asserted.
`wg`	out	'1' to indicate a *wordgap* is detected, '0' otherwise. Only valid when `valid` is asserted.
`valid`	out	'1' for 1 cycle to indicate a valide symbol is detected. '0' otherwise.
`clr`	in	Asynchrons clear to initial state.
`clk`	in	Input clock at 48 kHz.

Principle of Operation

Figure B.1 shows a sample input and the corresponding expected output:

As shown in the figure, the main task for `symdet` is to determine if a dot, dash, letter gap, or a word gap has been detected. If so, the corresponding signal, together with the `valid` signal will be asserted for 1 cycle.

There are 2 main technical challenges for designing `symdet`:

(a) The exact duration of a base unit (called u for now) is unknown.

(b) The actual detected duration of each Morse code symbol is not exact and will vary from time to time.

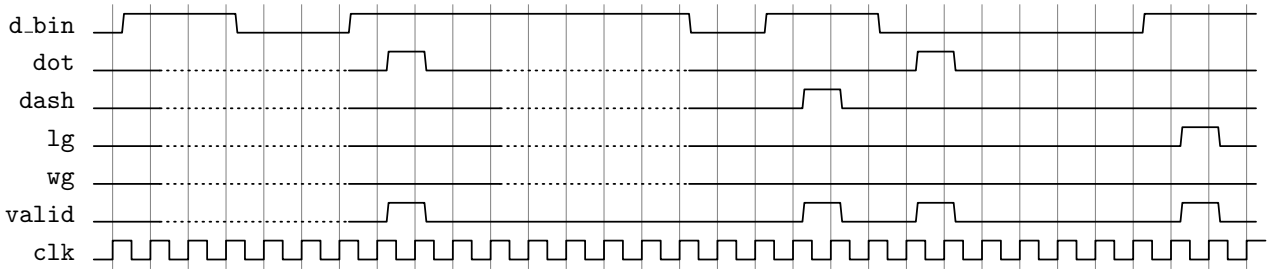


Figure B.1: Sample timing diagram of `symdet`

Because of (a) above, your `symdet` must estimate the duration (in terms of clock cycle) of a base unit u. Once that is determined, then it knows a dot should be 1 unit of '1', and a dash will have 3 unit of '1'. A letter gap will have 3 units of '0's and a word gap will have 7 units of '0's.

For instance, in Figure B.1, each unit u is above 3 actual cycles. Therefore, you can observe that a dash is about 9 cycles.

Because of (b) above, the actual value of u may change over time, and/or the length of the actual symbol may vary from time to time. For instance, in the above example, even though the expected length of a dash is 9 consecutive '1', in reality, your circuit may encounter only 8, or 11 '1's. As a result, your circuit must be a bit more flexible when detecting a symbol. Here is where you can be creative.

Finally, there are dotted lines in the above timing diagram because there is no fixed latency requirement between when a symbol is detected and when that corresponding signal is asserted. The only requirement is that the *sequence* of symbols is preserved, i.e., for instance in the above diagram, the dash will not be asserted before the first dot.

B.1.2 UART The second module your group needs to design is an UART for communication with the host computer. You will use this connection to display the alphabet or number received. A basic introduction to serial communication UART can be found here: `https://learn.sparkfun.com/tutorials/serial-communication/all`

For this project, you only need the tramsmission (TX) part of a standard UART. Also, your UART does not need to be flexible, and should only communicate with the following configuration:

Baud Rate	9600
Data Bit	8
Parity	N
Stop Bit	1

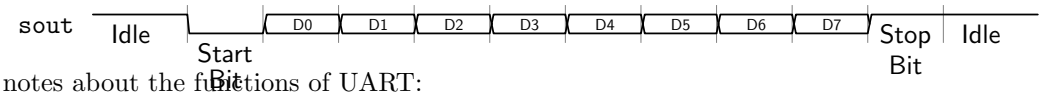
Your UART has the follwing I/I ports:

Name	Direction	Description
`d[7:0]`	in	8-bit data input.
`wen`	in	Asserted for 1 cycle to write data in `d` to transmit.
`sout`	out	Serial data to be transmitted.
`clr`	in	Asynchrons clear to initial state.
`clk`	in	Input clock at 48 kHz.

Principle of Operation

With the above restrictions, the only task your UART needs to perform is to convert an input 8-bit data `din` into a data packet in the 8-N-1 format for serial communication output at the `sout` port. Specifically,

given an 8-bit input `D[7:0]`, where `D0` is the lease significant bit (LSB) and `D7` is the most significant bit (MSB), your UART should produce the following waveform:



A few notes about the functions of UART:

- When idle, `sout` to be held at '1'
- Transmission begins by pulling `sout` to '0' for 1 baud.
- Transmission may begins after the stop bit.
- Since baud rate is 9600, the start bit, each data bit, and the stop but should be held for the entire duration of the baud, i.e, 1/9600 seconds.
- Input clock is 48 kHz.

B.1.3 What to Submit For this part, submit a brief report about your design. Make sure you include the following in the report:

- Names of the students in the group.
- Top level block diagram of each of your design (`symdet` and `UART`).
- Description of the functions.
- Screenshot of simulation waveform to help illustrate how it works.

Submit the report and all related vhdl files on Moodle.