Overview

From time-to-time telecommunications stations on the surface of Mars must be inspected by a team of qualified technicians. This happens in conditions conducive to movement on the surface by people.

In the station, the most important data on its condition is displayed on seven-segment displays, so technicians have a quick overview of the situation.

The station failure scenario assumes failure of the circuit that controls seven-segment displays, and more precisely, the conversion of numbers from binary to BCD (binary-coded decimal) code.

FPGA in the station has an implementation of this algorithm, but it turns out that it is faulty. Therefore, this task assumes fixing a bug in the provided RTL code.

Theoretical background

BCD is a number system in which the successive digits of the decimal digital representation of a number are written in binary code. For example, let us convert the number 26 to BCD:

$$26_{10} = \underset{2}{0010}\quad\underset{6}{0110}$$

This notation uses only 10 of the 16 states that the four bits used to represent each digit can take, so it is not used for storing or transmitting information. However, it is used in applications such as seven-segment displays.

The algorithm for converting a binary number to BCD consists of as many steps as the bit length of the number in BCD format. Each iteration checks whether any BCD digit is greater than or equal to 5 (0101 in binary code). If it is, that digit is incremented by 3. A bit shift is then performed so that the MSB of the binary number is inserted into the LSB of the BCD number. Table 1 shows the calculation steps for converting the number 243_{10}.

Table 1 243_{10} to BCD conversion algorithm.

HUNDREDS	TENS	ONES	BINARY	OPERATION
0000	0000	0000	11110011	Initialization
0000	0000	0001	11100110	Shift
0000	0000	0011	11001100	Shift
0000	0000	0111	10011000	Shift
0000	0000	1010	10011000	Add 3 to ONES, since it was 7
0000	0001	0101	00110000	Shift
0000	0001	1000	00110000	Add 3 to ONES, since it was 5
0000	0011	0000	01100000	Shift
0000	0110	0000	11000000	Shift
0000	1001	0000	11000000	Add 3 to TENS, since it was 6
0001	0010	0001	10000000	Shift
0010	**0100**	**0011**	00000000	Shift

Goal

Your goal is to **debug** the *binary2bcd* module. Analyze its operation, make necessary code changes, or implement additional parts of a module to fix it. You are allowed to make changes only in the *binary2bcd* module.

Example of a correct answer:

Input value: $01000011_2 = 67_{10}$

Output BCD value: 01100111_{BCD}

Provided module

You can edit only the *binary2bcd.sv* file.

Module ***binary2bcd*** is supposed to realize conversion from binary to **2 digits** BCD code with the help of FSM. It has the following inputs and outputs:

Table 2 List of inputs and outputs of the binary2bcd module.

Signal name	Signal type	Bit length
i_clk	Input	1 bit
i_rst	Input	1 bit
i_binary	Input	8-bit
i_busy_input	Input	1 bit
i_busy_output	Input	1 bit
i_empty_input	Input	1 bit
i_full_output	Input	1 bit
o_BCD	Output	8-bit
o_valid_output	Output	1 bit
o_req_input	Output	1 bit

The FSM diagram of this module is shown in Figure 1.

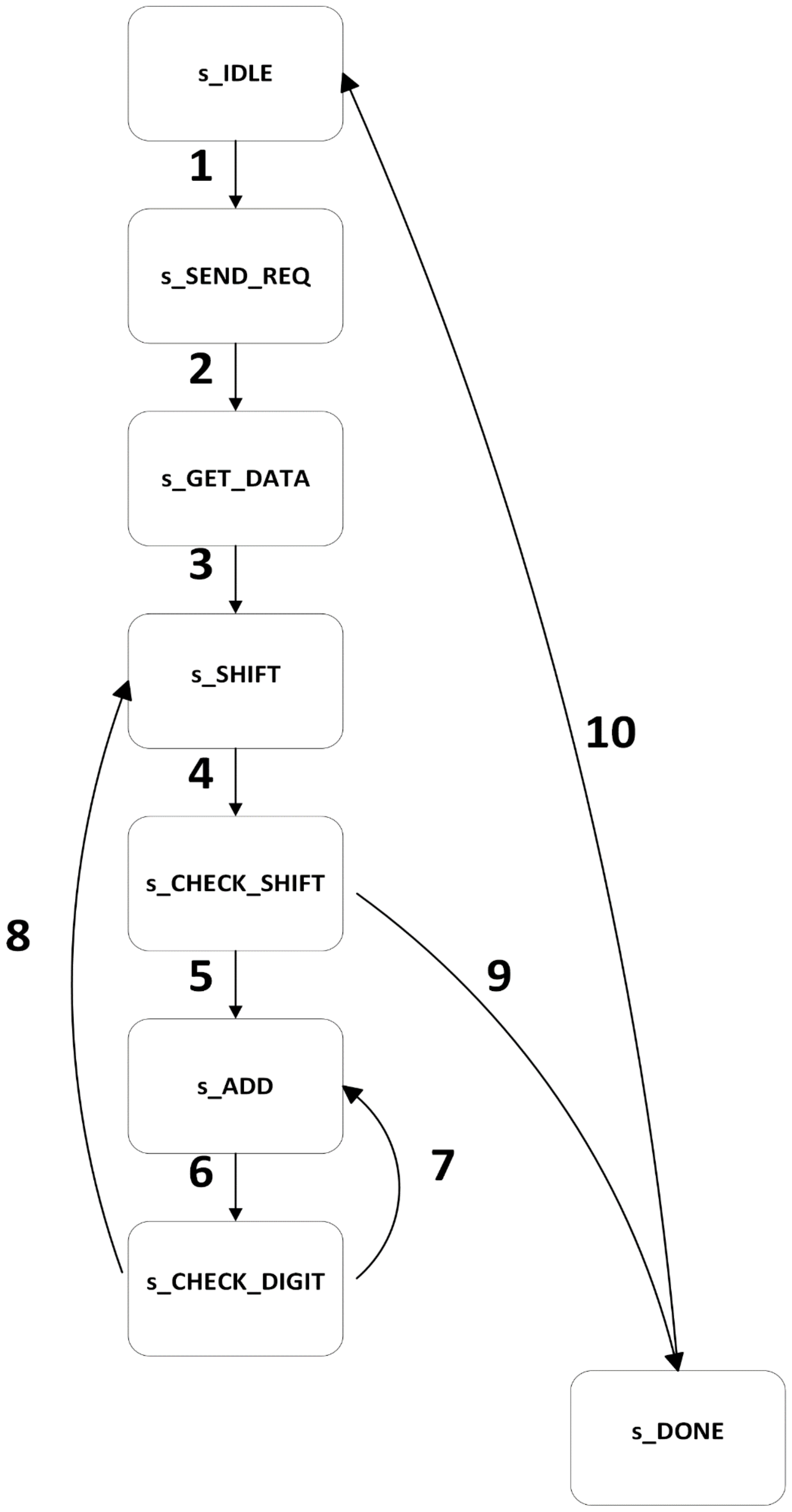

Figure 1 Finite State Machine Diagram for Task 5.

Table 3 List of state changes of the binary2bcd module.

#	Current state	Next state	Condition
1	s_IDLE	s_SEND_REQ	i_busy_input==0 && i_empty_input==0
2	s_SEND_REQ	s_GET_DATA	1 (always)
3	s_GET_DATA	s_SHIFT	1 (always)
4	s_SHIFT	s_CHECK_SHIFT	1 (always)
5	s_CHECK_SHIFT	s_ADD	r_loop_count != BINARY_WIDTH-1
6	s_ADD	s_CHECK_DIGIT	1 (always)
7	s_CHECK_DIGIT	s_ADD	r_digit_index != DECIMAL_DIGITS-1
8	s_CHECK_DIGIT	s_SHIFT	r_digit_index == DECIMAL_DIGITS-1
9	s_CHECK_SHIFT	s_DONE	r_loop_count == BINARY_WIDTH-1
10	s_DONE	s_IDLE	i_full_output==0 && i _busy_output==0

Where:

DECIMAL_DIGITS = 2

BINARY_WIDTH = 8

binary_2_bcd module gets its data from the *task_5_input* module and sends it to the *task_5_output* module. **DO NOT CHANGE THOSE MODULES.** Assume that they work correctly.

Input and output interfaces

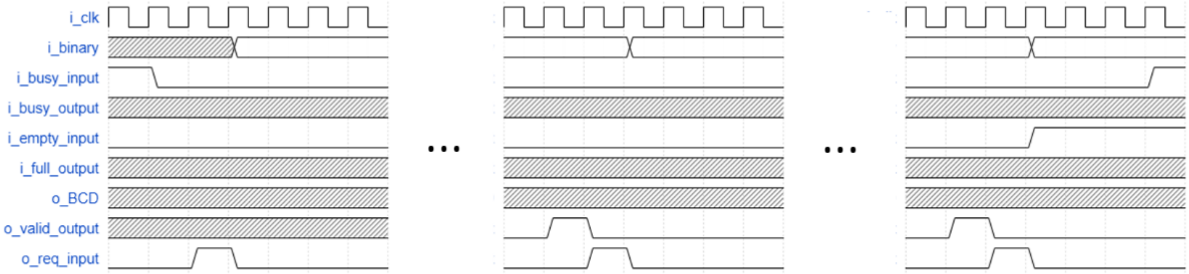

Figure 2 Waveforms showing how the task receives data. The first sample, one in the middle of a packet and the last one from a packet.

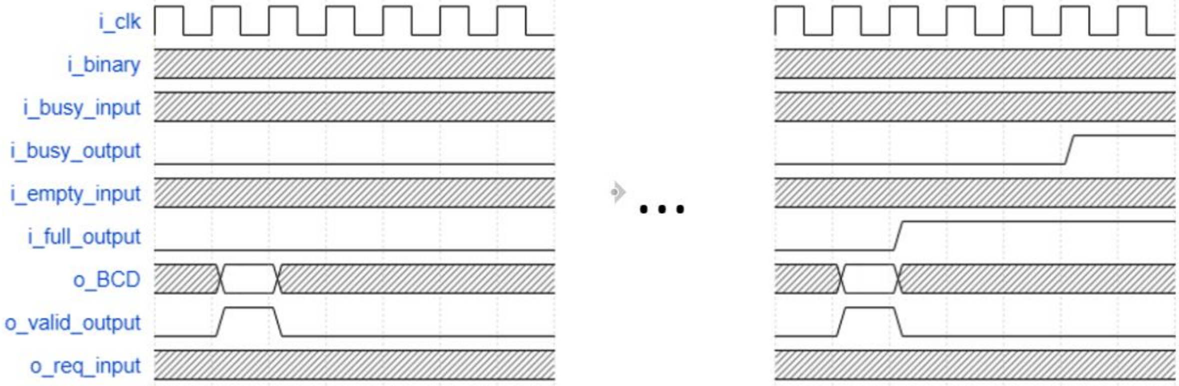

Figure 3 Waveforms *showing how the module should send data*. The sample was calculated based on the data from the middle of a packet and the last from a packet.

Evaluating the task

For the task to be considered correctly performed, for each number received in binary format task must send its BCD form.