Verilog modeling using the `always` and `initial` blocks

Proposed problems

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

October 9, 2024

Problem 1

Design a Verilog module named *msd*, using always blocks, having one input i on 5 bits and one output o on 4 bits. Output o is the value of the most significant decimal digit in input i's unsigned number decimal representation. Example: $i = 23 \rightarrow o = 2$; $i = 9 \rightarrow o = 9$.

Note: For implementing this problem download source file ▸ *msd.v* and, respectively, script file ▸ *run_msd.txt*, only modifying the Verilog source file.

Problem 2

Construct a module named *div3*, having an input *i*, on 4 bits and an output *o*, on the minimum number of bits so that at the output to provide the quotient of dividing by 3 the unsigned number connected at the input *i*. Do not use the "/" Verilog division operator for this exercise and implement the module using an always block. Example: $i = 10 \rightarrow o = 3$; $i = 2 \rightarrow o = 0$.

Note: For implementing this problem download source file ▸ *div3.v* and, respectively, script file ▸ *run_div3.txt*, only modifying the Verilog source file.

Problem 3

Construct a device for counting the number of bits of 1 for the unsigned number connected at the input *i*. The module, named *cnt1s*, has an input *i*, on 6 bits and an output *o* on the minimum required number of bits. Build the module using always blocks.

Note: For implementing this problem download source file ▸ *cnt1s.v* and, respectively, script file ▸ *run_cnt1s.txt*, only modifying the Verilog source file.

Problem 4

Design a device named *seq3b* having an input i on 4 bits and an output o on a single bit. The output is active if the binary number connected at the input contains a sequence of 3 or more consecutive bits with the same value. Example: $i = 14 \rightarrow o = 1$; $i = 9 \rightarrow o = 0$.

Note: For implementing this problem download source file ▸ *seq3b.v* and, respectively, script file ▸ *run_seq3b.txt*, only modifying the Verilog source file.

Problem 5

Construct a module named *mul5bcd* having a 4-bit input i and two ouptuts d and u, both on 4 bits. At input i it receives a BCD digit (on 4 bits) and the module provides at the output the result of multiplying by 5, in BCD, the input digit: the output d represents the decimals' figure while the output u represents the units' figure of the result. Example: $i = 3(0011) \rightarrow d = 1(0001)$, $u = 5(0101)$; $i = 9 \rightarrow d = 4(0100)$, $u = 5(0101)$.

Note: For implementation download source file ▸ *mul5bcd.v* and, respectively, script file ▸ *run_mul5bcd.txt*, only modifying the Verilog source file.

Problem 6

Design module *text2nibble*, having input *i* on 8 bits and output*o* on 4 bits. Input *i* receives one ASCII character. If the character is a decimal digit ('0' to '9'), provided at output the value of the digit, otherwise provide value 15.

ASCII char	ASCII char value (radix 16)	Output o
'0'	30 (00110000)	0
'1'	31 (00110001)	1
'2'	32	2
'3'	33	3
'4'	34	4
'5'	35	5
'6'	36	6
'7'	37	7
'8'	38	8
'9'	39 (00111001)	9

Problem 6 (contd.)

Note: For implementation download source file ▸ *text2nibble.v* and, respectively, script file ▸ *run_text2nibble.txt*, only modifying the Verilog source file.

Problem 7

Build a register on 4 bits, named *r4b*. The register can perform a parallel load of the value on 4-bit input *d* into its content and it can also right-shift its content by 1 position, in which case the value of the most significant bit is provided at the 1-bit input *sh_in*. The register has 2 synchronous command inputs: command line *ld* – which activates the parallel loading of *d* and command line *sh* – which triggers the right-shift. The register has a 4-bit output *q*, representing its current content. For implementation modify the Verilog source file ▸ *r4b.v*.

Note: The synchronous sequential devices, like the register above, will have an implicit input for the clock signal, *clk*, and an implicit asynchronous, active low, reset line, *rst_b*, unless otherwise stated.