Electronic System Level Design

Assignment 2, 2017-5-23

Design 1: Half and Full Adders

Abstract

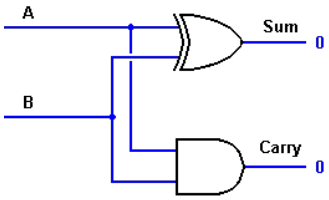
Implement a half-adder, then use two (2) half-adders to implement a full adder. Then synthesize the full adder using Cadence CTOS.

Please read carefully. All outputs required are described in the text. Five (5) points will be taken for each bug, missing required output and behavior.

The half-adder `SC_METHOD` module

Description

1. A half-adder schematic is given below

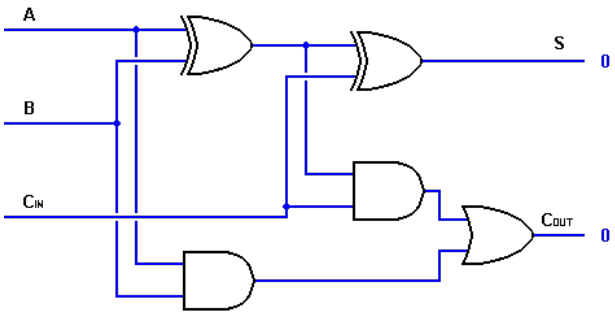


2. Use above schematic as the specification and implement a `SC_MODULE` with a `SC_METHOD` process. All input and output ports are named exactly the same as in the schematic.

The full-adder `SC_METHOD` module

Description

1. A full-adder schematic is given below



2. Use above schematic as the specification and implement a `SC_MODULE` with a `SC_METHOD` process. You must instantiate two (2) half-adders developed above to implement this full-adder. Again all input and output ports are named exactly the same as in the schematic. C_{in} should be named `Cin` and C_{out} should be named `Cout`.

sc_main

Description

1. Create a test suite, i.e. `sc_main`, that
   - Instantiate both half-adder and full-adder modules
   - Provide all possible input combinations to these modules, i.e. 4 input vectors to the half-adder and 8 input vectors to the full-adder.
   - Create a trace file named `RESULT.vcd`. And trace ports in following order:
     - Half-adder `A`
     - Half-adder `B`
     - Half-adder `Sum`
     - Half-adder `Carry`
     - Full-adder `A`
     - Full-adder `B`
     - Full-adder `Cin`
     - Full-adder `S`
     - Full-adder `Cout`

Makefile

A `makefile`, or `Makefile`, must be provided

Synthesize with CTOS

A `fa.tcl` CTOS script file is provided. Please use the script to make sure C2Silicon can successfully synthesize your code.

NOTE:

1) `#ifndef HA_H` class redefine prevention technique must be used in the header file.
2) Each design uses one directory to not mingle two designs together.

Design 2: Finite Impulse Response Digital Filter

Abstract

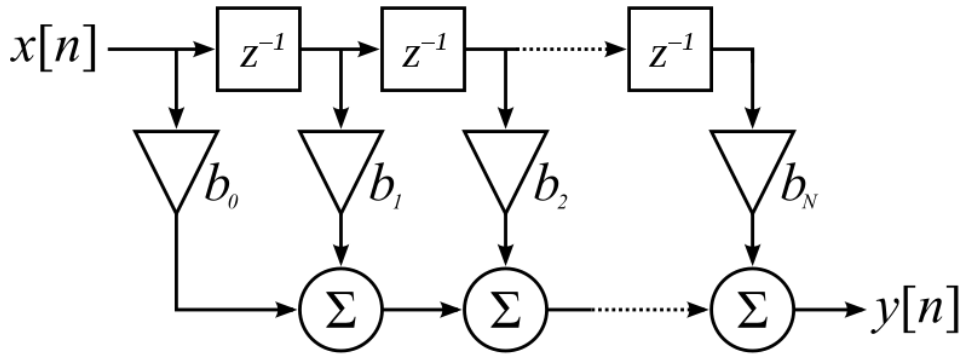
Implement a 16th-order discrete-time Finite Impulse Response (FIR) digital filter, in 50MHz clock. Then synthesize the filter using Cadence CTOS.

Please read carefully. All outputs required are described in the text. Five (5) points will be taken for each bug, missing required output and behavior.

The 16th-order FIR module with a `SC_CTHREAD` process

Description

3. A schematic of N^{th}-order discrete-time FIR filter is given below



4. Use above schematic as the specification and implement a non-pipeline 16th-order FIR filter, in `SC_MODULE` with a `SC_CTHREAD` process
5. The input port is named `x` and its data type is `sc_uint<32>`
6. The output port is named `y` and its data type is `sc_uint<32>`
7. The clock port is named `clk`. The reset pin is named `rst`
8. Let us implement a *Moving Average Filter*, a.k.a. *boxcar filter*, where all b_i = 1/(N + 1). Therefore in this case $b_i = 1/17 \approx 0.05882353$
9. Let us use a fixed point system of `wl=32` and `iwl=16` for all computations. Then b_i = 0x00000F0F. However, this is just to explain how you going to set b_i values in the module. Notice that in your code there is no need to use fixed point data types.
10. You need to implement the asynchronous reset behavior, in which all delays are reset to 0.

sc_main

Description

2. Create a test suite, i.e. `sc_main`, that
   - Instantiate the FIR module
   - Provide 64 data inputs to the module
   - Create a trace file named `RESULT.vcd`. And trace ports in following order:
     - `clk`
     - `rst`
     - `x`
     - `y`

makefile

A `makefile`, or `Makefile`, must be provided

Synthesize with CTOS

A `filter.tcl` CTOS script file is provided. Please use the script to make sure C2Silicon can successfully synthesize your code.

NOTE: please tar two designs into one compressed file to turn in.

Please turn in the source codes only and `makefile`. Do not turn in the executable.

Due date

17:00, June 6, 2017

Score weight (towards the final grade) 10%