Digital Systems Electronics Laboratory 02

Group 13

s295391 Giorgio Zoccatelli
s295567 Vittorio Macripò

Due Date:	April 8, 2024
Delivery date:	March 10, 2024
Instructor:	Professor Guido Masera

Politecnico di Torino
Accademic Year 2023/24

Contents

1 Introduction

The purpose of this laboratory experience is to implement different circuits that can execute the basic operations of addition, subtraction and multiplication between binary numbers. We will also go trough a more depth time analysis pointing out the different working frequencies of these circuits and finding out the critical paths for each case. Please notice that for the further descriptions, although we double checked that the overflow operation worked correctly on the singular testbenches, it seems to stop working when including the *.vho* and *.sdo* files in ModelSim.

2 4-bit Sequential RCA

2.1 Design entry

For the first exercise we want to describe a sequential 4-bit ripple-carry adder that supports signed numbers in 2's-complement: to do so we need the following designs:

- Three registers that will be updated every rising clock edge occurrence. Two of them will store the terms of the sum given by switches $SW_{7\text{-}4}$ and $SW_{3\text{-}0}$ and the third will store the result of the sum. This design is implemented in *regn.vhd.*
- A ripple-carry adder which is the heart of our circuit will take as an input the values to sum and will give as an output the result and the carry out (in this example we are not yet implementing the subtraction so the carry in will be always set to 0). To correctly sum two 4-bit signed numbers know from the theory that we can split the circuit in four connected full adders so for this purpose we will create two designs: *full_adder.vhd* and *ripple_adder.vhd.*
- After the sum operation we want to know if an overflow is occuring. We know from the theory that a 4-bit number in 2's-complement can correctly represent values from -8 to 7 and more precisely we can notice that the overflow occurs only when the carry out of the MSB is different from the MSB of the sum and at the same time the carry out must be different from the carry in of the LSB. This operation is described in *overflow_check.vhd.*
- We want to visually see on the FPGA whenever an overflow is occuring so we store the result in a D-type Flip Flop that will trigger the $LEDR_9$. The Flip Flop is described in *flipflop.vhd.*
- We want to visually see on the FPGA the sum displayed following the hexadecimal systems so as we did for the previous laboratory activity we need a 7-segment decoder described in *display_hexadecimal.vhd.*

All these designs are hierarchically described in a top entity *fourbit_rca.vhd* where a unique clock signal driven by KEY_1 and a uniqe reset signal KEY_0 are assigned to all the memory elements.

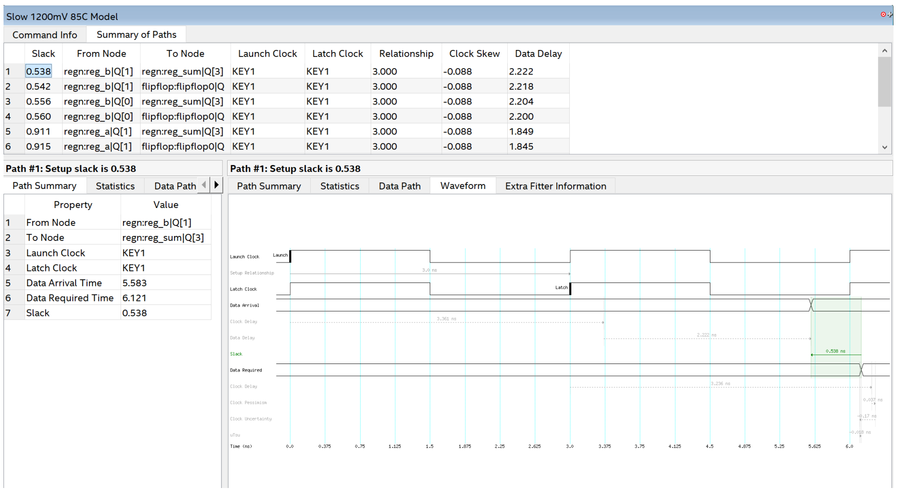
2.2 Functional Simulation

In this section we want to implement a testbench to correctly simulate the behavior of our circuit. The rules to apply the testbench signals are well known but we want to point out how to correctly implement the clock and reset signals:

- KEY_0 and KEY_1 are initially set to 0 which means by convention that the reset signal is high and the clock signal is down.
- After giving a starting input configuration to the switches we set KEY_0 to 1 (reset low) and KEY_1 to 1 (clock high) so that the two registers of the sum will change their output following the switches configuration and the ripple-carry adder can properly sum the two values. This combinational circuit will then give the result of the sum and the carry out to the third register and to the Flip Flop after the overflow is checked.
- Without changing the state of KEY_0 we set KEY_1 to 0 and then again to 1 so that at the second rising edge of the clock we can change the output of the third register as well as the output of the Flip Flop triggering both the HEX displays and the $LEDR_9$.
- We can finish a test cycle by setting both KEY_1 and KEY_0.

2.3 Synthesis

With the synthesis we want to test the correct functioning of our circuit and we want to analyse the time constricts regarding the clock period and the worst case delay trough memory elements.


Slow 1200mV 85C Model

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.538	regn:reg_b|Q[1]	regn:reg_sum|Q[3]	KEY1	KEY1	3.000	-0.088	2.222
2	0.542	regn:reg_b|Q[1]	flipflop:flipflop0|Q	KEY1	KEY1	3.000	-0.088	2.218
3	0.556	regn:reg_b|Q[0]	regn:reg_sum|Q[3]	KEY1	KEY1	3.000	-0.088	2.204
4	0.560	regn:reg_b|Q[0]	flipflop:flipflop0|Q	KEY1	KEY1	3.000	-0.088	2.200
5	0.911	regn:reg_a|Q[1]	regn:reg_sum|Q[3]	KEY1	KEY1	3.000	-0.088	1.849
6	0.915	regn:reg_a|Q[1]	flipflop:flipflop0|Q	KEY1	KEY1	3.000	-0.088	1.845

Path #1: Setup slack is 0.538

Path Summary | Statistics | Data Path

	Property	Value
1	From Node	regn:reg_b|Q[1]
2	To Node	regn:reg_sum|Q[3]
3	Launch Clock	KEY1
4	Latch Clock	KEY1
5	Data Arrival Time	5.583
6	Data Required Time	6.121
7	Slack	0.538

Path #1: Setup slack is 0.538

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

Figure 1: Circuit delays

In Figure 1 we can see that using a clock signal with a period of 3 ns we obtain a circuit with a maximum operating frequency $f_{\max}$ equal to 406.17 Hz. The same figure also points out that our longest path trough the circuit in terms of delay is the one going from *reg_b* (one of the two register for the sum terms) to *reg_sum* (the register storing the result of the sum). By importing our testbench on ModelSim as well as the *.vho* and *.sdo* files we can then analyse the testbench with respect to all the internal signals delays.

In Figure 2 we can see an overall timing evolution of the circuit while in Figure 3 we can see as a prove of the correct delay representation the transition between the reset and the display of the sum terms -3 and -4 occurring in the second testbench configuration.

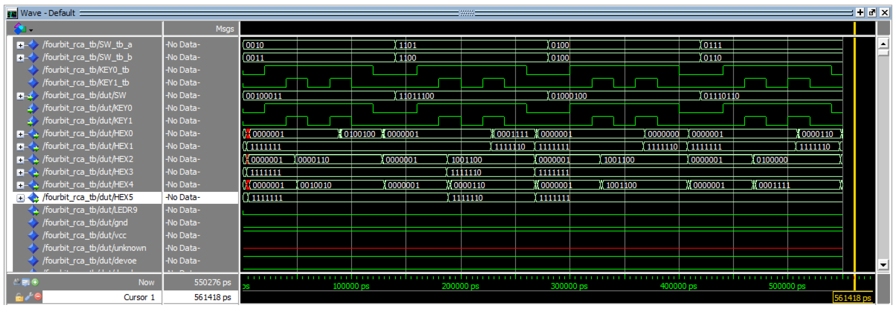


Figure 2: Overall testbench

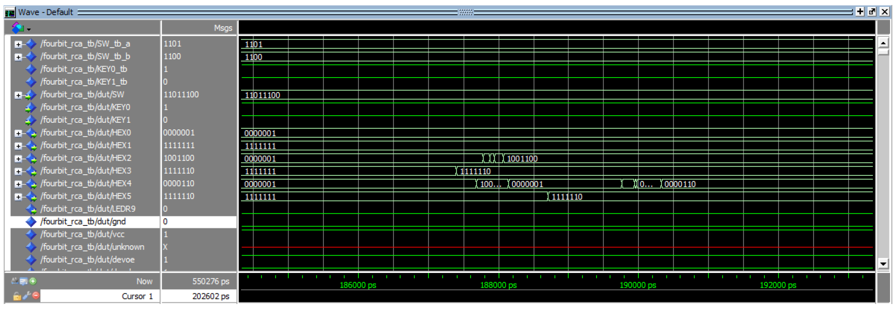

Wave - Default
Msgs
/fourbit_rca_tb/SW_tb_a 1101
/fourbit_rca_tb/SW_tb_b 1100
/fourbit_rca_tb/KEY0_tb 1
/fourbit_rca_tb/KEY1_tb 0
/fourbit_rca_tb/dut/SW 11011100
/fourbit_rca_tb/dut/KEY0 1
/fourbit_rca_tb/dut/KEY1 0
/fourbit_rca_tb/dut/HEX0 0000001
/fourbit_rca_tb/dut/HEX1 1111111
/fourbit_rca_tb/dut/HEX2 1001100
/fourbit_rca_tb/dut/HEX3 1111110
/fourbit_rca_tb/dut/HEX4 0000110
/fourbit_rca_tb/dut/HEX5 1111110
/fourbit_rca_tb/dut/LEDR9 0
/fourbit_rca_tb/dut/gnd 0
/fourbit_rca_tb/dut/vcc 1
/fourbit_rca_tb/dut/unknown X
/fourbit_rca_tb/dut/devoe 1
Now 550276 ps
Cursor 1 202602 ps

Figure 3: Particular delayed transition

3 4-bit Sequential Adder/Subtractor

3.1 Design entry

3.2 Functional Simulation

3.3 Synthesis

4 16-bit RCA, Carry-Bypass Adder and Carry-Select Adder

4.1 Design entry

4.2 Functional Simulation

4.3 Synthesis

5 Multiplier

5.1 Design entry

5.2 Functional Simulation

5.3 Synthesis