Digital Systems Electronics Laboratory 03

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 occurring. 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 (*fourbit_rca_tb.vhd*) 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.

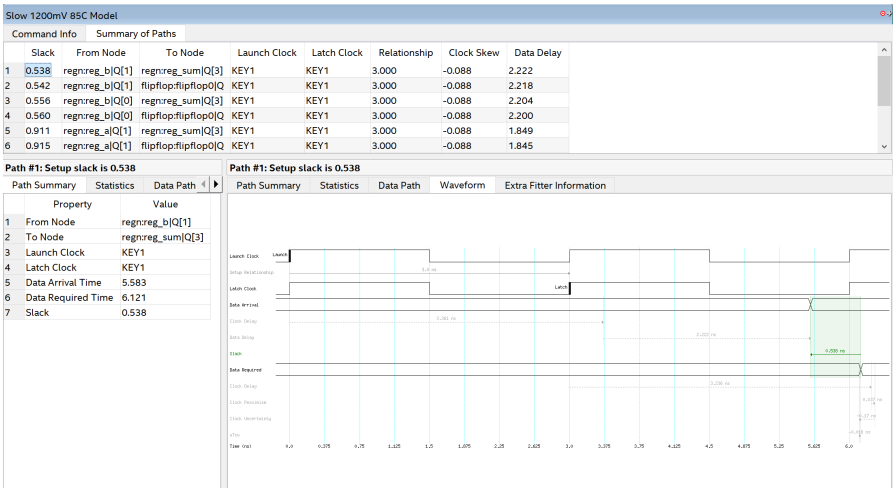

Figure 1: Internal delays

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 MHz. 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.

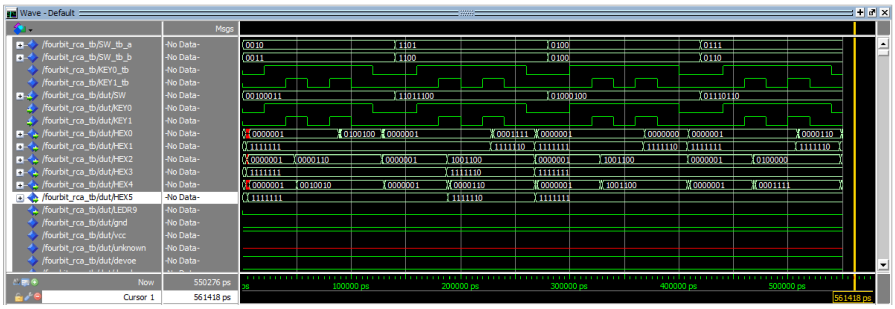

Figure 2: Overall testbench

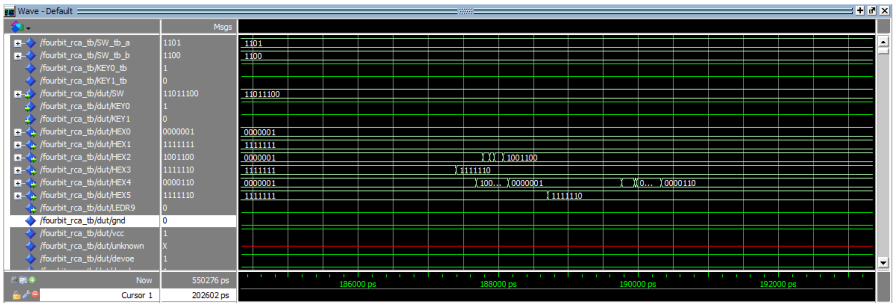

Figure 3: Particular delayed transition

3 4-bit Sequential Adder/Subtractor

3.1 Design entry

In this second exercise we have to slightly modify the previous circuit in order to implement also the subtraction. In particular:

- We allocate a new switch SW_8 to decide if we want to execute an addition (logic value 0) or a subtraction (logic value 1).
- If we want to add two numbers the circuit will work exactly as the previous one.
- If we want to subtract two numbers we have to complement the second operand and introduce a carry in equal to 1 for the LSB. In practice we are making a sum between the first number and the second number with the sign changed which is the exact concept of the subtraction.

In practice the only components being changed from the previous exercise will be *ripple_adder.vhd* and the top entity *fourbit_adder_subtractor.vhd*.

3.2 Functional Simulation

As for the previous exercise we implement a testbench with respect to the clock and reset inpunts as already stated. We will implement four different configurations:

- An addition between two positive numbers without overflow: (+2) + (+3).

- A subtraction between two negative numbers without overflow: (-3) - (-4).
- A subtraction between two positive numbers without overflow: (+6) - (+4).
- An addition between two positive numbers with overflow: (+7) + (+6).

3.3 Synthesis

In the synthesis we want to test once again the correct functioning of our circuit trough the wave analysis on ModelSim as well as the internal delays introduced by the memory elements.

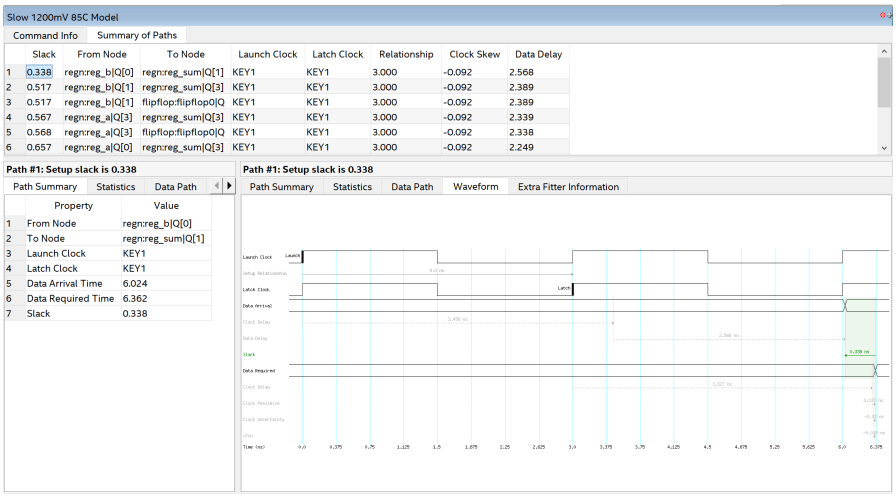

Figure 4: Internal delays

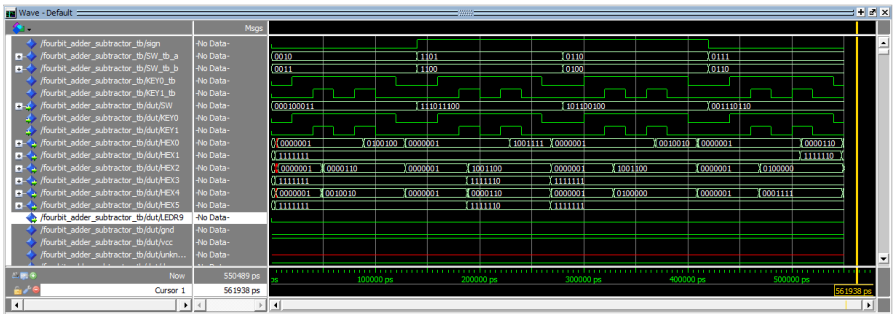

Figure 5: Overall testbench

In Figure 4 we can see that this circuit has a maximum operating frequency $f_{\max}$ of 375.66 MHz while once again the longest path in terms of delay is the one going from *reg_b* (one of the two sum/subtraction terms registers) to *reg_sum* (the register where we memorize the current result value).

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

4.1 16-bit RCA

1. **Design entry**
   The design resumes that of section 2, but it is in this case required the implementation of a 16-bit RCA instead of a 4-bit RCA, as can be seen from *sixteenbit_rca* where 16 different full adders are arranged one in series to another in order to transfer the carry signal. The implementation scheme and the functioning observed by the components (that have simply undergone dimensional adjustments) are the same as for the previous project. The only big difference is that 16-bit input signals can't properly fit the board and for this reason the project is limited to the use of arbitrary input and output signals that do not match the board.

2. **Functional Simulation**
   The implemented testbench has the same structure as the one previously implemented in section 2.2. As can be seen from the *testbench_rca.vhd* file, arbitrary test signals such as *a_tb, b_tb, Clk_tb* and *Rst_tb* are created. These signals correspond to those passed by the top entity and, unlike the 4-bit RCA exercise, don't have correspondeces on the board.

3. **Synthesis** The time analysis is performed by taking into account the worst case delay as shown in the image below.

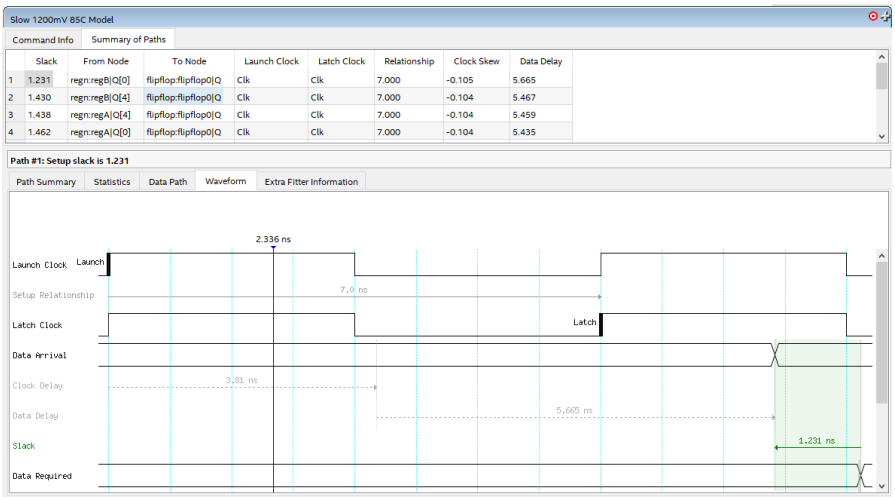

Figure 6: Internal delays

The maximum operating frequency f_{max} is equal to 173.34 MHz.
Starting from the testbench implemented in the *testbench_rca.vhd* file we can appreciate the correct functioning of the project observing that the result of the sum of two 16-bit numbers is correct and does not present, as expected, the overflow phenomenon.

4.2 16-bit CBA

1. **Design entry** The Carry Bypass Adder (CBA) is a summer that uses propagate signals

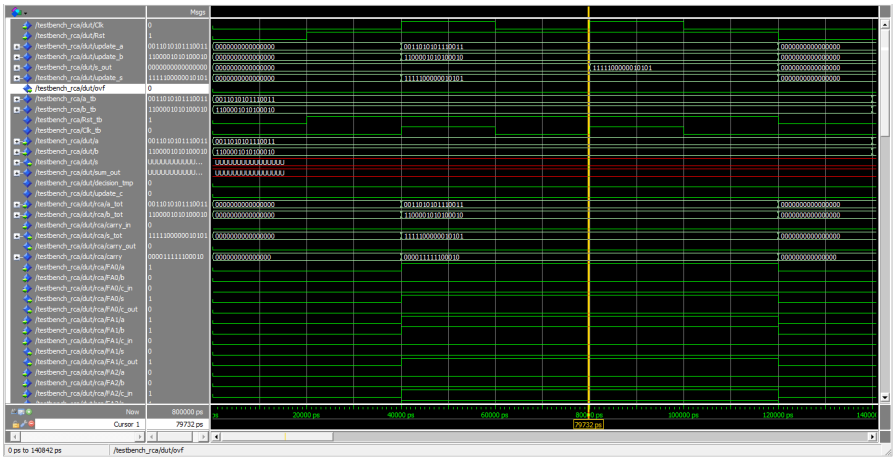

Figure 7: Overall testbench - 16-bit RCA

p to decide the *carry_out* signal with the help of a multiplexer. The design for a 16-bit CBA consists of 4 blocks (described in the *carry_bypass_adder_4bit.vhd*), each of which includes the 4 full adders (*full_adder_4bit.vhd*) and the multiplexer (*mux_2to1.vhd*). Therefore the circuit as a whole includes a total of 16 full adders and 4 multiplexers that are all enclosed within the *cba_16bit.vhd* file.

2. **Functional Simulation** The implemented testbench has the same structure as the one previously implemented in section 2.2 as can be seen from the *cba_tb.vhd* file.

3. **Synthesis** The time analysis is performed by taking into account the worst case delay as shown in the image below.

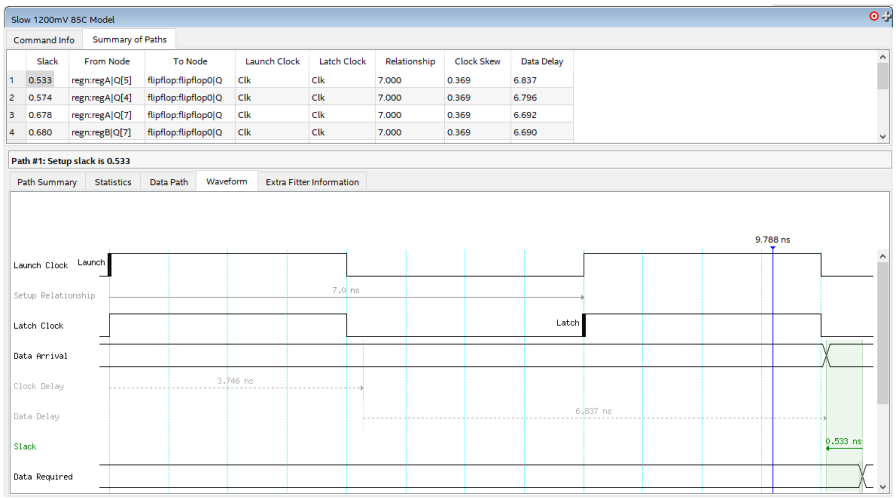

Figure 8: Internal delays

The maximum operating frequency f_{max} is equal to 154.63 MHz.
Starting from the testbench implemented in the *cba_tb.vhd* file we can appreciate the correct

functioning of the project observing that the result of the sum of two 16-bit numbers is correct and does not present, as expected, the overflow phenomenon.

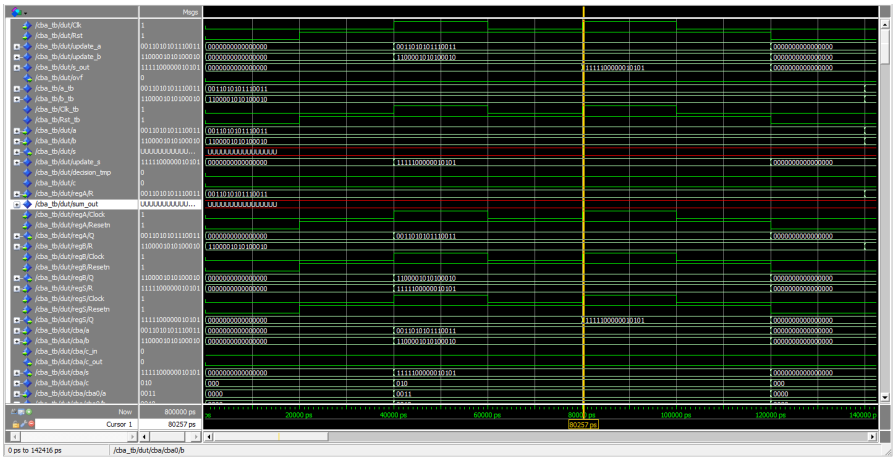

Figure 9: Overall testbench - 16-bit CBA

4.3 16-bit CSA

1. **Design entry** The 16-bit Carry Select Adder (CSA) is a summer that uses 7 different ripple carry adder (*rca_4bit.vhd*, for a total of 4 full adders for each RCA) and multiplexers and that is fully described in *csa.vhd* file. To realize the project, two types of multiplexer are implemented in the design:

   - 3 4bit-2to1-multiplexers (*mux_4bit.vhd*) useful for determining the sum result;
   - 3 1bit-2to1-multiplexers (*mux_2to1.vhd*) useful to determine the value of carry and transmit it to the next block.

   For each block, except for the first one where the carry is definitely '0', 2 RCAs are described with the respective input logic signals '1' and '0' representing the input carry.

2. **Functional Simulation** The implemented testbench has the same structure as the one previously implemented in section 2.2 as can be seen from the *csa_tb.vhd* file.

3. **Synthesis** The time analysis is performed by taking into account the worst case delay as shown in the image below.

   The maximum operating frequency f_{max} is equal to 284.74 MHz.
   Starting from the testbench implemented in the *csa_tb.vhd* file we can appreciate the correct functioning of the project observing that the result of the sum of two 16-bit numbers is correct and does not present, as expected, the overflow phenomenon.

5 Multiplier

5.1 Design entry

In this last exercise we want to implement a 4-bit multiplier also known as *array multiplier*. As shown in the circuit schematic we can implement this function by only using AND gates

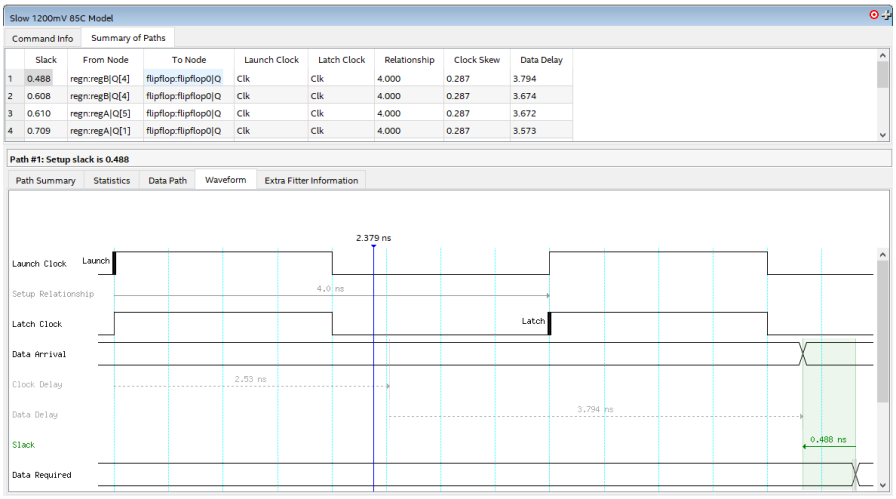

Figure 10: Internal delays

and ripple-carry adders. We won't spend much words about the design for the exact fact that the schematic is already given and the functioning of the ripple-carry adder has already been analyzed. The idea is that at every stage (except for the first one) we will pass one bit b_x of the multiplication term B that will be the input to four AND gates while the other four inputs of the gates are given by the result of the previous ripple-carry adder. The top entity is described in the fil *multiplier.vhd* and a display of the result obtained as always trough the 7-segmend displays is given in the file *display_hexadecimal.vhd*. Please notice that we failed in understanding how to represent all 256 different output values on the displays without writing hundreds of line of code for every case so we implemented the display only for numbers from 0 to 30 to check the correct functioning of the circuit.

5.2 Functional Simulation

In the functional simulation we implemented a testbench with some configurations to see if the two multiplication terms as well as the result are correctly displayed. The only defines two test signal to simulate switches $SW_{7\text{-}4}$ and $SW_{3\text{-}0}$ on the FPGA.

5.3 Synthesis

In Figure 11 we can see the wave analysis of our testbench: notice that as required HEX_0 and HEX_1 are used to display the two terms of the sum while HEX_2 and HEX_3 are used for the two multiplication terms. In the Figure we can see the two configurations displayed in hexadecimal values are $2 \times 3 = 6$ and $1 \times 14 = 14$.

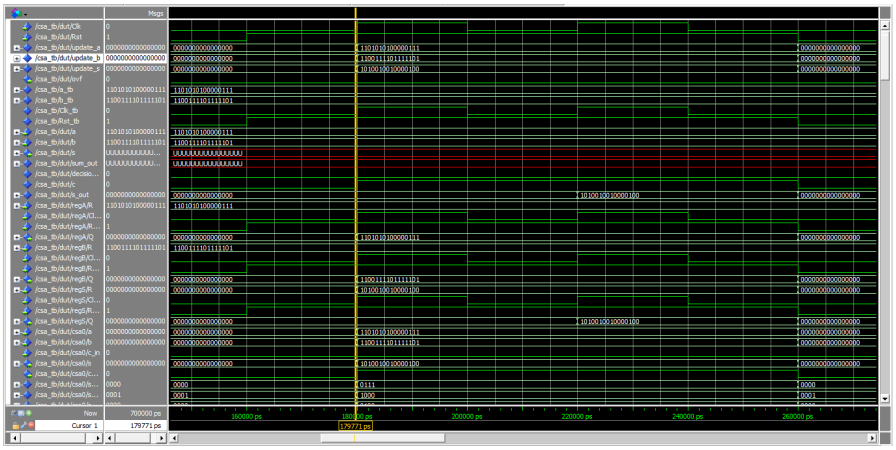

Figure 11: Overall testbench - 16-bit CSA

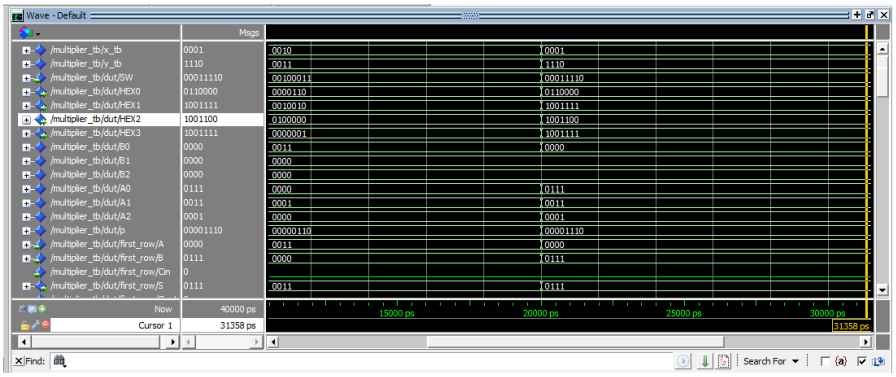

Figure 12: Overall testbench