Lab Session 2

HDL based design and implementation of combinational logic (n-bit adder)

Introduction

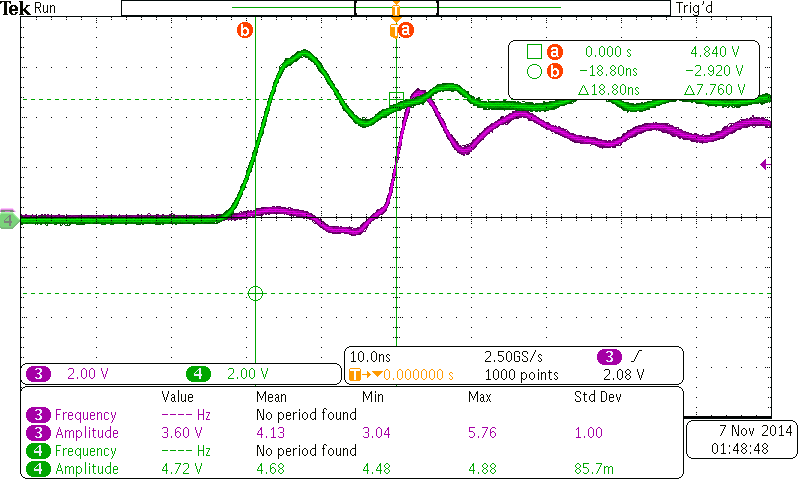
In this lab session, we endeavor to simulate an N-bit adder operation using VHDL, Xlinx. Following lab tasks 2.1 to 3.2 we design and show a working model of an 8-bit ripple adder demonstrating the functional principle, the included delay and implementation to hardware via Xlinx of our VHDL project.

**Lab task2.1**

*Table 1*

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C\_IN** | **S** |
| 0 | 1 | 0 | 1 |
| 0 | 2 | 0 | 2 |
| 2 | 3 | 1 | 6 |
| 3E | 94 | 0 | D2 |
| 65 | 51 | 0 | 106 |
| 78 | 2C | 0 | C4 |
| FF | FF | 1 | 1FF |
| 0 | 0 | 0 | 0 |
| FF | 1 | 0 | 100 |

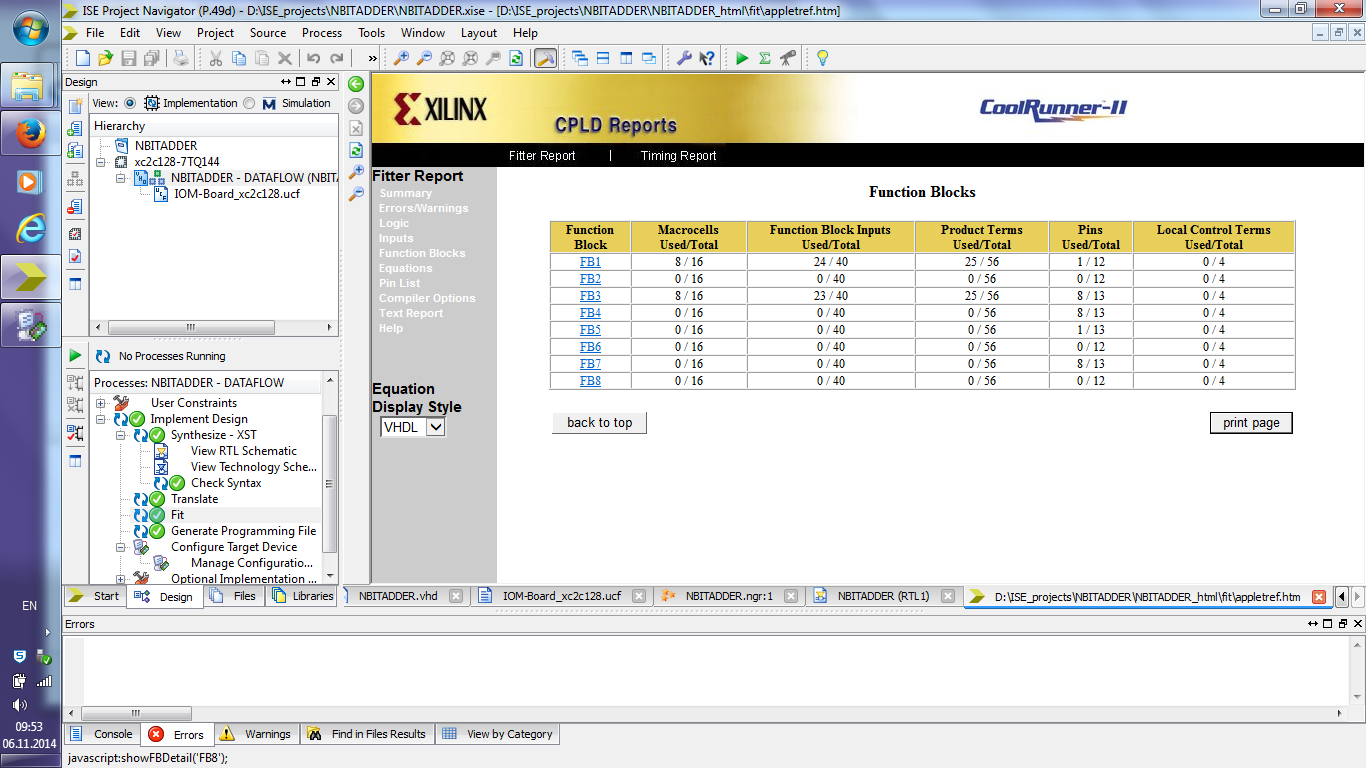
In Table 1, we verified using manual switches that the given results and the test cases implemented in our lab preparation are exactly identical.



*Figure 1: Oscilloscope screenshot, delay measurement*

After synthesizing the VHDL model of the 8-bit carry adder, we have measured the delay of 18.8ns between C\_IN and S(7). (see Figure 1)

**Lab task 2.2**



*Figure 2: Screenshot of the Function blocks in Xlinx*

Total usage of Macrocells: 16/128 (13%)

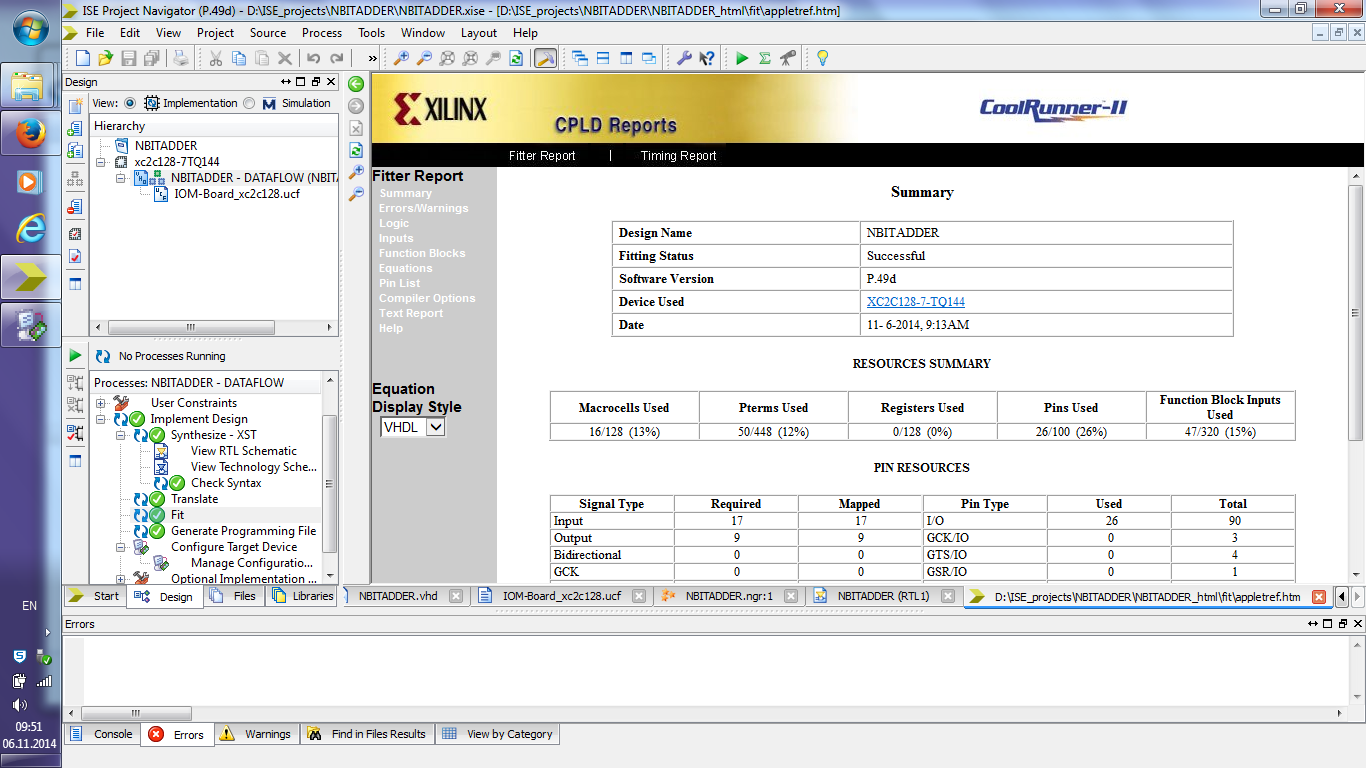
Total usage of Function blocks: 47/320 (15%)

Total usage of Product terms: 50/448 (12%)

Total usage of Pins: 26/100 (26%)

Total usage of Registers: 0/128 (0%)

(see Figure 2 and 3)



*Figure 3: Screenshot of the Resource summary in Xlinx*

**Lab Task 2.3**

Equation for S(0), S(1) and S(2) using Boolean symbols of the 8-bit adder

S(0) <= ((B(0) ^ ((⌐ C\_IN ^ ⌐ B(0)) v (⌐ C\_IN ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0)))) v (A(0) ^ ((⌐ C\_IN ^ ⌐ B(0)) v (⌐ C\_IN ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0)))) v (C\_IN ^ B(0) ^ A(0)) v (C\_IN ^ ⌐ B(0) ^ ⌐ A(0)))

S(1) <= B(1) ~~v~~ ((((⌐ C\_IN ^ ⌐ B(0)) v (⌐ C\_IN ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0))) ^ A(1)) v (((⌐ C\_IN ^ ⌐ B(0)) v (⌐ C\_IN ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0))) ^ ⌐ A(1)))

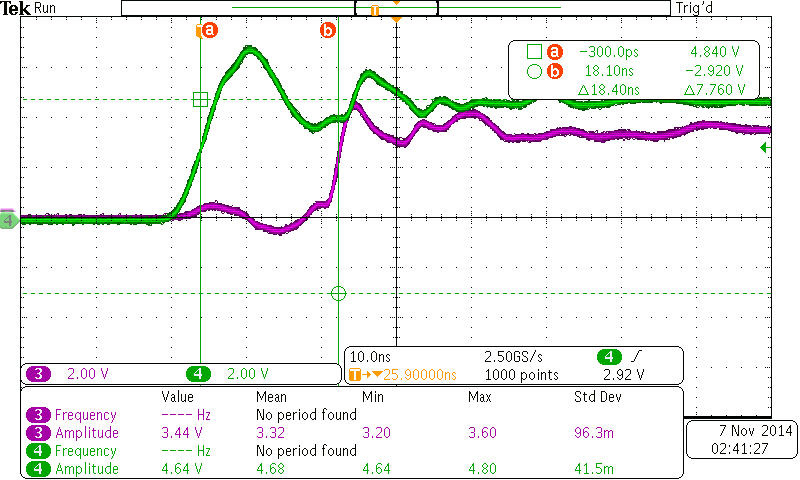
S(2) <= ⌐ (((⌐ B(1) ^ ⌐ A(1)) v (((⌐ C\_IN ^ ⌐ B(0)) v (⌐ C\_IN ^ ⌐ A(0)) v (⌐B(0) ^ ⌐ A(0))) ^ B(1) ^ ⌐ A(1)) v (((⌐ C\_IN ^ ⌐ B(0)) v (⌐ C\_IN ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0))) ^ ⌐ B(1) ^ A(1))))

**Lab Task 3.1**

*Table 2*

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **CIN** | **S** |
| 0 | 1 | 0 | 1 |
| 0 | 2 | 0 | 2 |
| 2 | 3 | 1 | 6 |
| 3E | 94 | 0 | D2 |
| 65 | 51 | 0 | 106 |
| 78 | 2C | 0 | C4 |
| FF | FF | 1 | 1FF |
| 0 | 0 | 0 | 0 |
| FF | 1 | 0 | 100 |

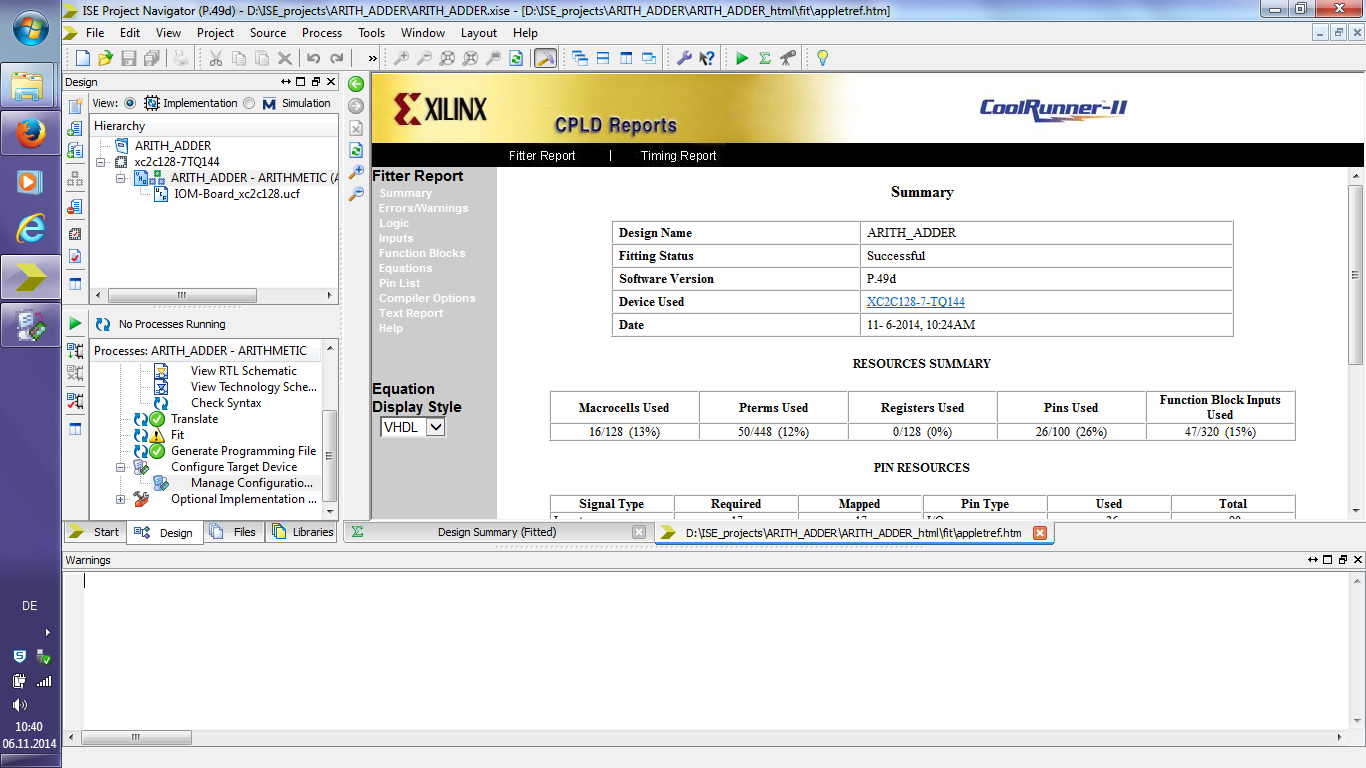
In Table 2, we verified using manual switches that the given results and the test cases implemented in our lab preparation are exactly identical.



*Figure 4: Oscilloscope screenshot, delay measurement*

The delay we measured between C\_IN and S(7) is 18.40ns. (see Figure 4)

**Lab task 3.2**

****

*Figure 4: Screenshot of the Resource summary in Xlinx*

Total usage of Macrocells: 16/128 (13%)

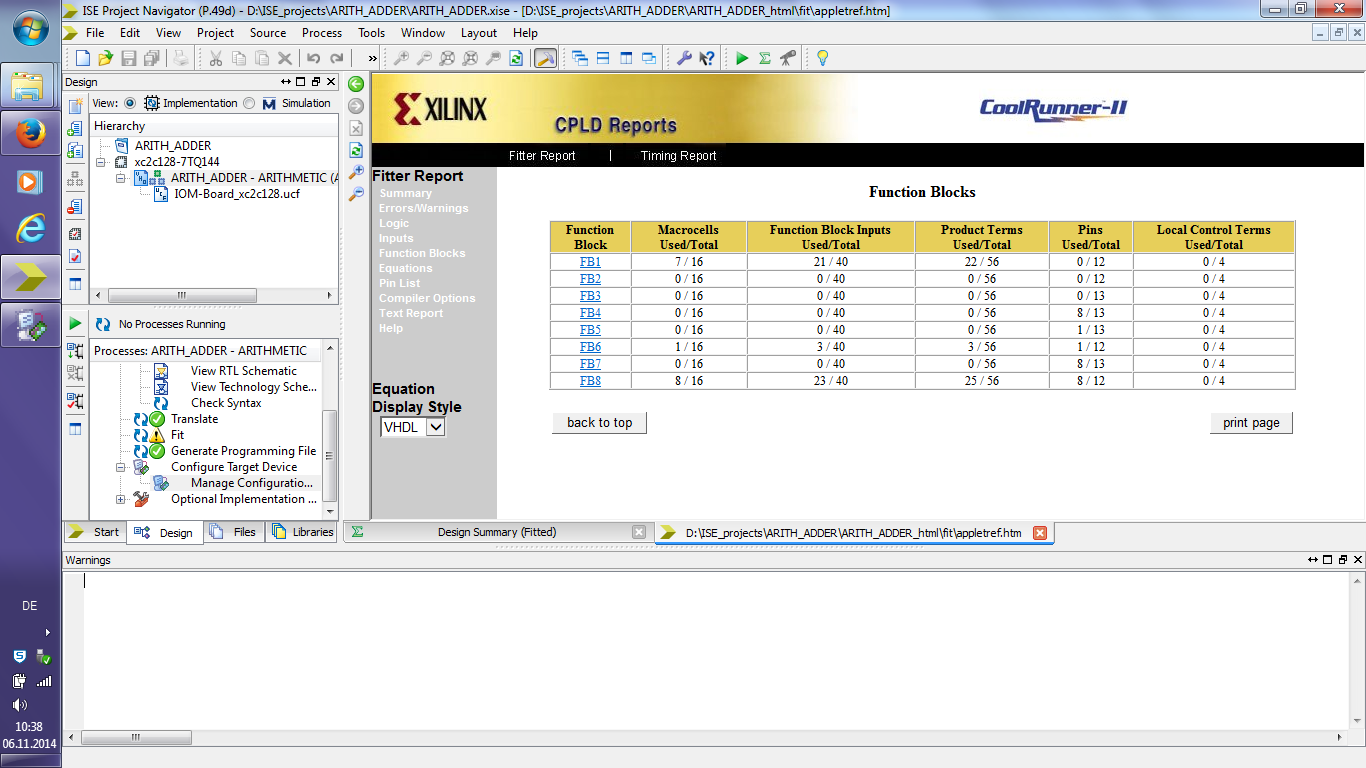
Total usage of Function blocks: 47/320 (15%)

Total usage of Product terms: 50/448 (12%)

Total usage of Pins: 26/100 (26%)

Total usage of Registers: 0/128 (0%)

(see Figure 5 and 6)



*Figure 5: Screenshot of the Function blocks in Xlinx*

*Table 3*

|  |  |  |
| --- | --- | --- |
| **Resources** | **N-bit adder** | **Arithmetic adder** |
| Macrocells | 13% | 13% |
| Product terms | 12% | 12% |
| Registers | 0% | 0% |
| Pins | 26% | 26% |
| Function blocks | 47% | 47% |
| Propagation delay | 18.80ns | 18.40ns |

In Table 3 we compared the two adders with respect to resources and propagation delay. The results we got are identical.

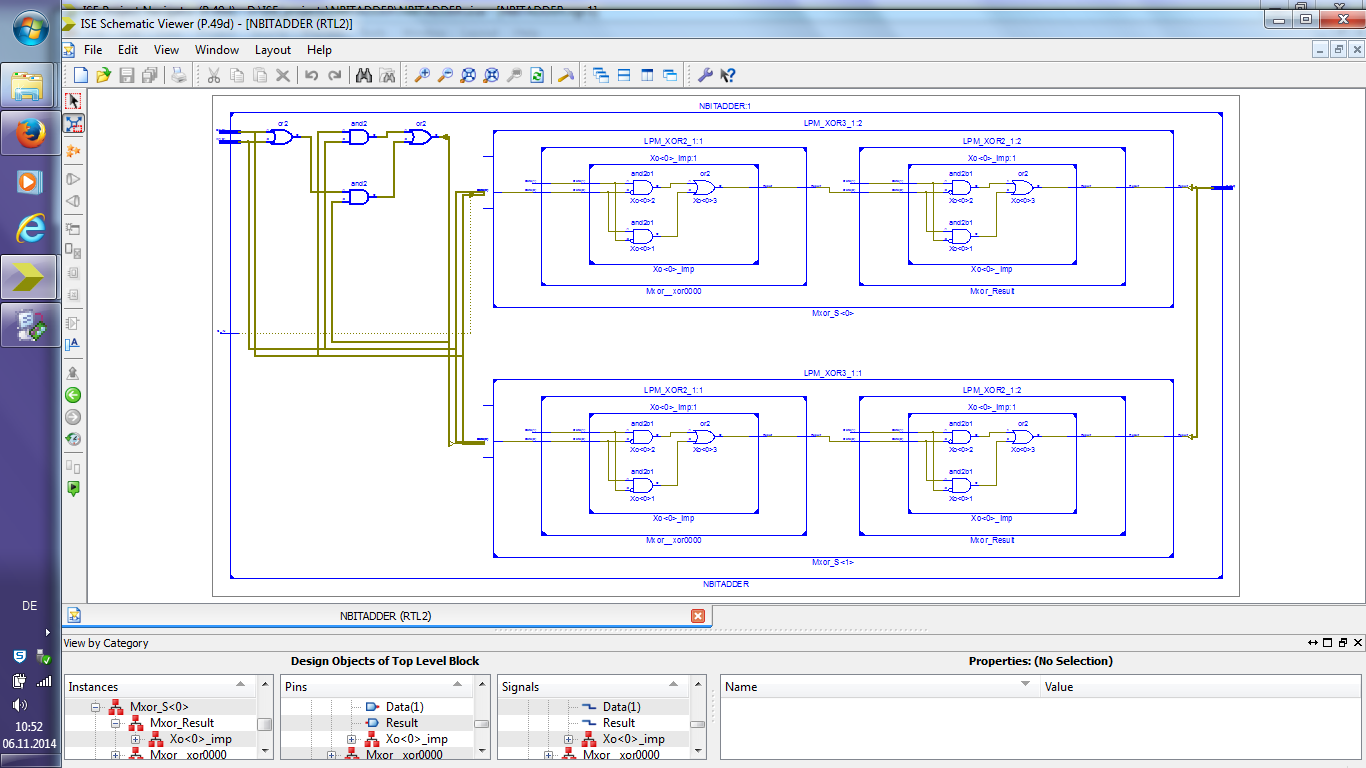
**Extra task:**

Equation for S(0), S(1) and S(2) using Boolean symbols of the arithmetic adder

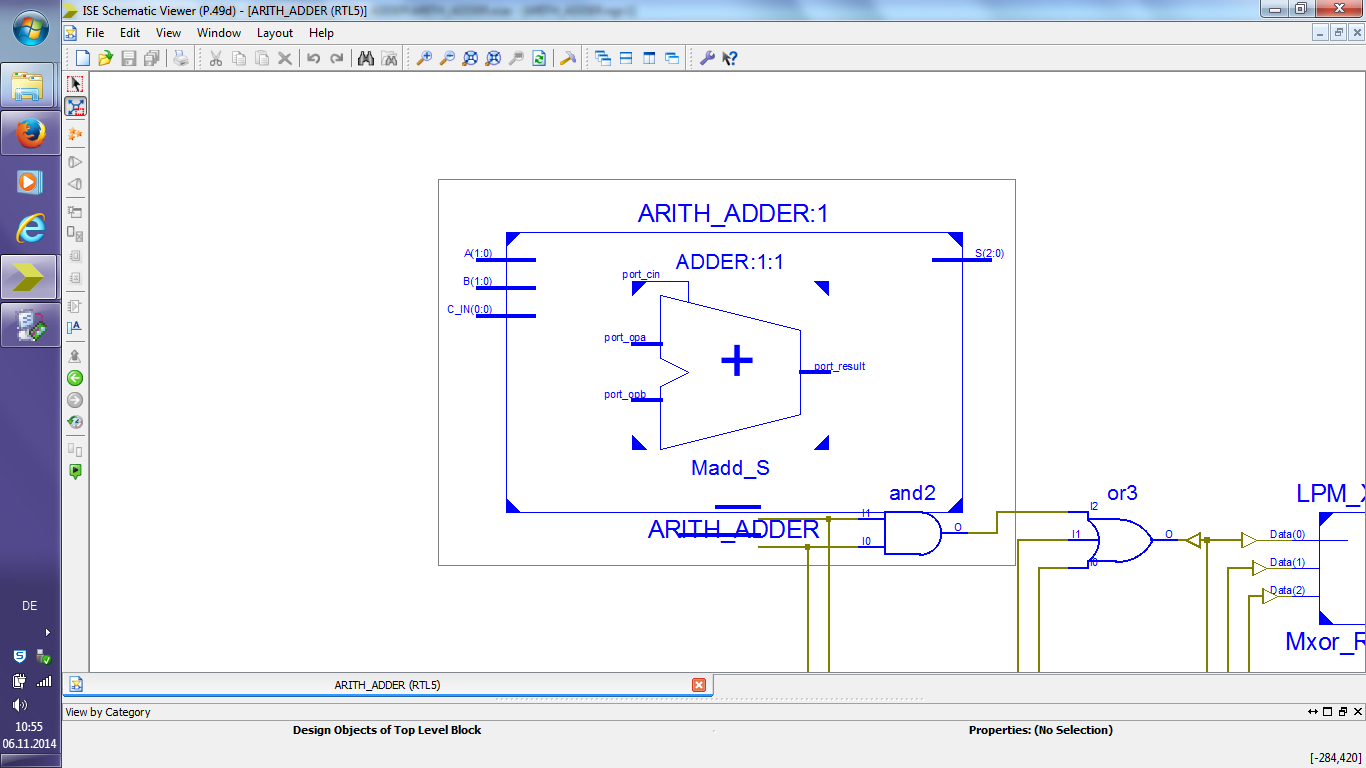
S(0) <= ((B(0) ^ ((⌐ C\_IN(0) ^ ⌐ B(0)) v (⌐ C\_IN(0) ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0)))) v (A(0) ^ ((⌐ C\_IN(0) ^ ⌐ B(0)) v (⌐ C\_IN(0) ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0)))) v (C\_IN(0) ^ B(0) ^ A(0)) v (C\_IN(0) ^ ⌐ B(0) ^ ⌐ A(0)))

S(1) <= B(1) ~~v~~ ((((⌐ C\_IN(0) ^ ⌐ B(0)) v (⌐ C\_IN(0) ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0))) ^ A(1)) v (⌐ ((⌐ C\_IN(0) ^ ⌐ B(0)) v (⌐ C\_IN(0) ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0))) ^ ⌐ A(1)))

S(2) <= (((⌐ B(1) ^ ⌐ A(1)) v (((⌐ C\_IN(0) ^ ⌐ B(0)) v (⌐ C\_IN(0) ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0))) ^ B(1) ^ ⌐ A(1)) v (((⌐ C\_IN(0) ^ ⌐ B(0)) v (⌐ C\_IN(0) ^ ⌐ A(0)) v (⌐ B(0) ^ ⌐ A(0))) ^ ⌐ B(1) ^ A(1))))



*Figure 6: Screenshot of the RTL schematic of the 8-bit adder*

**

*Figure 7: Screenshot of the RTL schematic of the Arithmetic adder*

**Remark:** Sadly we could not zoom in to see the full schematic due to the software configuration. (see Figure 7)