Materia:

Diseño de Circuitos Integrados Digitales CMOS II

Nombre de la Tarea:

Carry-lookahead Adder

Nombre del estudiante:

Ariel Gonzales Diaz

Nombre del Profesor:

Dr. José Roberto Reyes Barón

Fecha:

23 de Agosto de 2023

Introduction.

In this homework, we have a practice of a Carry-lookahead Adder. With the objective of creating this system in the program Quartus Prime and explain how it works in our FPGA.

Theoretical Framework.

Program a Carry-lookahead Adder in the Terasic DE10-Lite board to ensure that in the future, we can integrate this knowledge in a bigger project with this board.

Apply the knowledge acquired in the past semester in the class, translate an circuit to VHDL language for the FPGA and Make the typical development of a code.

Development.

A carry-lookahead adder (CLA) or fast adder is a type of electronics adder used in digital logic. A carry-lookahead adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, ripple-carry adder (RCA), for which the carry bit is calculated alongside the sum bit, and each stage must wait until the previous carry bit has been calculated to begin calculating its own sum bit and carry bit. The carry-lookahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger-value bits of the adder.

Already in the mid-1800s, Charles Babbage recognized the performance penalty imposed by the ripple-carry used in his Difference Engine, and subsequently designed mechanisms for anticipating carriage for his never-built Analytical Engine. Konrad Zuse is thought to have implemented the first carry-lookahead adder in his 1930s binary mechanical computer, the Zuse Z1. Gerald B. Rosenberger of IBM filed for a patent on a modern binary carry-lookahead adder in 1957.

Two widely used implementations of the concept are the Kogge–Stone adder (KSA) and Brent–Kung adder (BKA).

Carry-lookahead depends on two things:

1. Calculating for each digit position whether that position is going to propagate a carry if one comes in from the right.
2. Combining these calculated values to be able to deduce quickly whether, for each group of digits, that group is going to propagate a carry that comes in from the right.

Supposing that groups of four digits are chosen. The sequence of events would go like this:

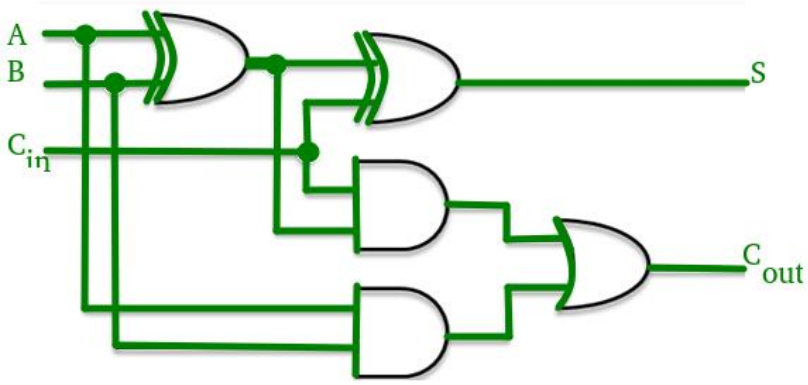
1. All 1-bit adders calculate their results. Simultaneously, the lookahead units perform their calculations.
2. Assuming that a carry arises in a particular group, that carry will emerge at the left-hand end of the group within at most five gate delays and start propagating through the group to its left.
3. If that carry is going to propagate all the way through the next group, the lookahead unit will already have deduced this. Accordingly, *before the carry emerges from the next group*, the lookahead unit is immediately (within one gate delay) able to tell the *next* group to the left that it is going to receive a carry – and, at the same time, to tell the next lookahead unit to the left that a carry is on its way.

The net effect is that the carries start by propagating slowly through each 4-bit group, just as in a ripple-carry system, but then move four times as fast, leaping from one lookahead-carry unit to the next. Finally, within each group that receives a carry, the carry propagates slowly within the digits in that group.

The more bits in a group, the more complex the lookahead carry logic becomes, and the more time is spent on the "slow roads" in each group rather than on the "fast road" between the groups (provided by the lookahead carry logic). On the other hand, the fewer bits there are in a group, the more groups have to be traversed to get from one end of a number to the other, and the less acceleration is obtained as a result.

Deciding the group size to be governed by lookahead carry logic requires a detailed analysis of gate and propagation delays for the particular technology being used.

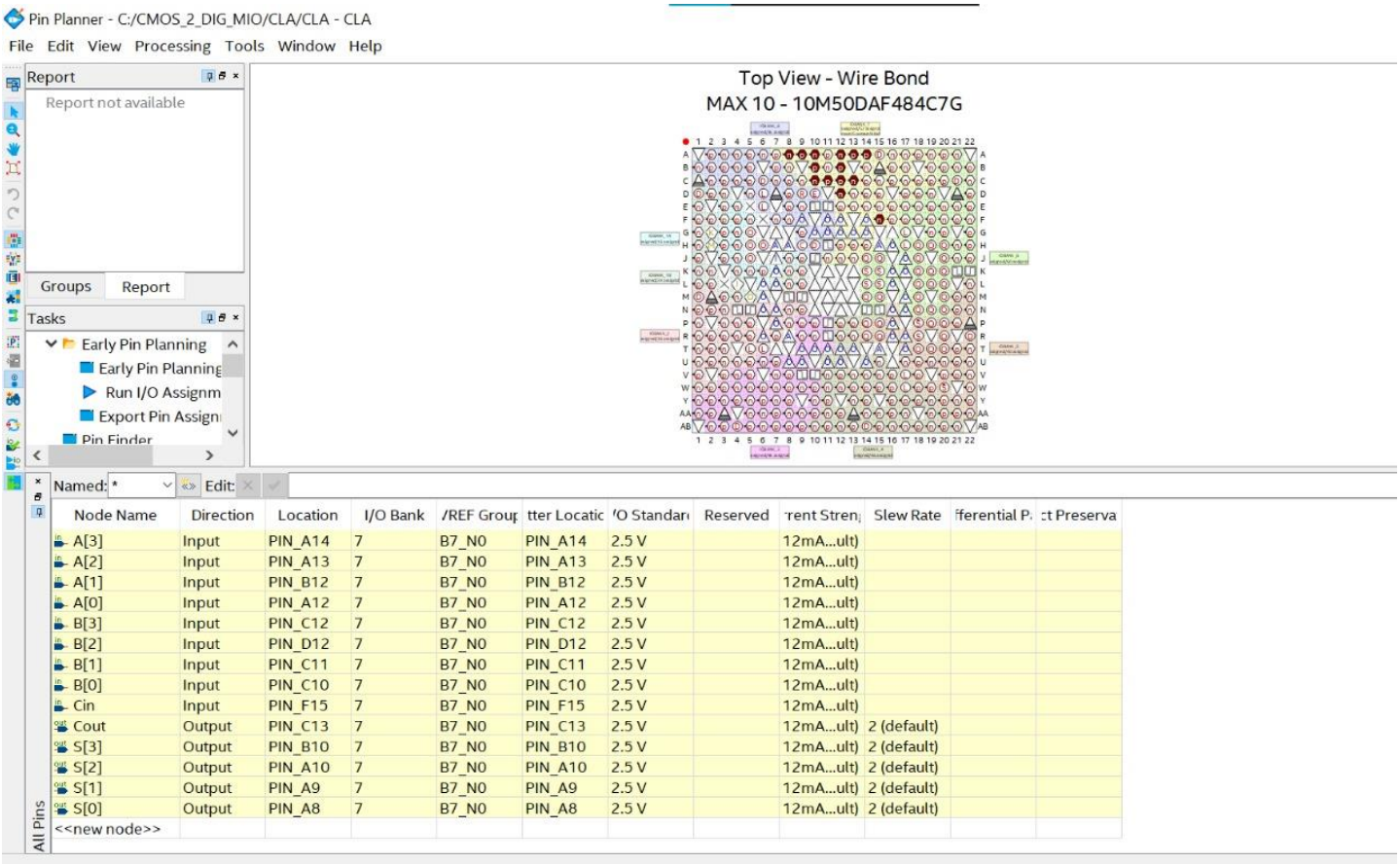
A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic.

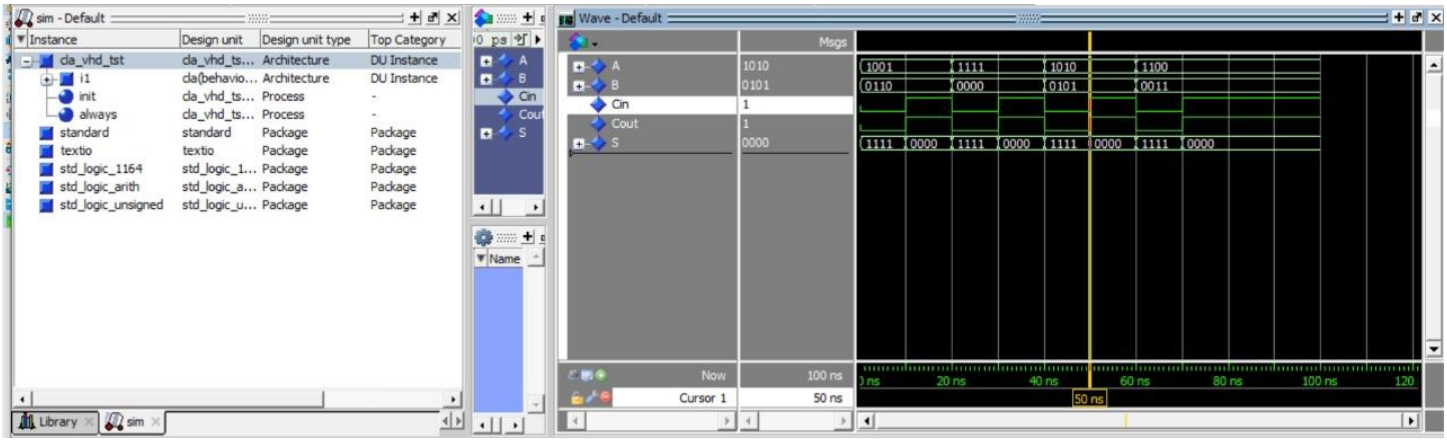


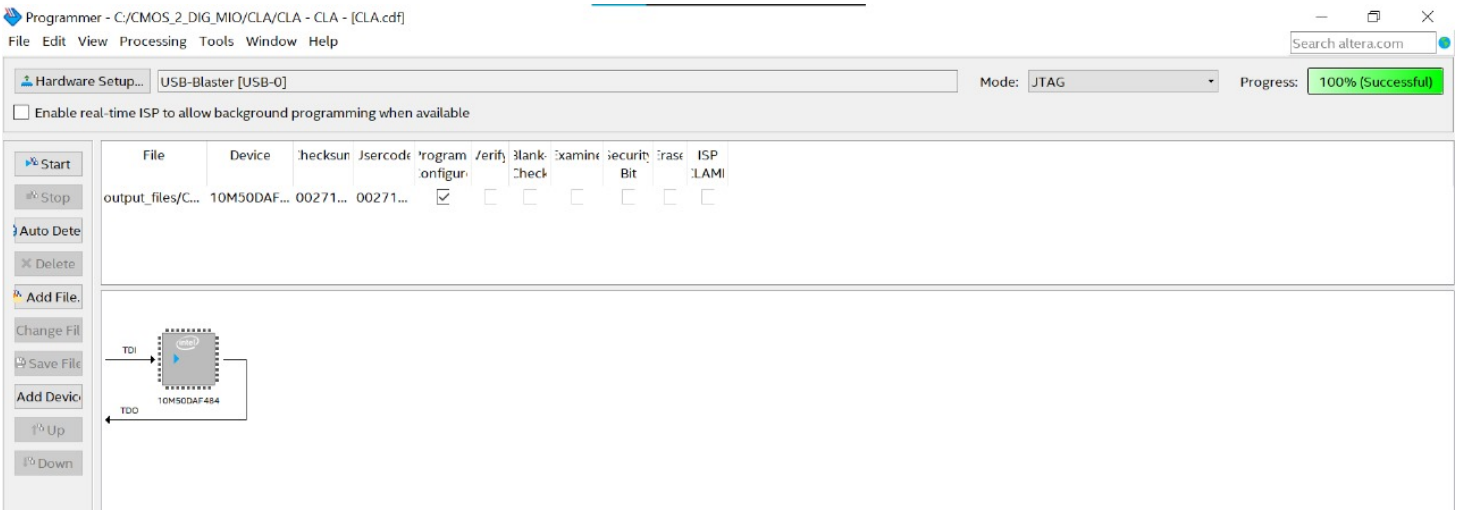
A	B	C	C +1	Condition
0	0	0	0	No Carry Generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No Carry Propogate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry Generate
1	1	1	1	

Two circumstances serve as the foundation for each lookahead operation:

- To determine whether a carry bit is propagated from the right location, calculate each digit position.
- The output for each pair of digits where the group creates a propagation bit that originates from the proper place is then created by combining the computed values.


Pin Planner - C:/CMOS_2_DIG_MIO/CLA/CLA - CLA
File Edit View Processing Tools Window Help
Report
Report not available
Groups Report
Tasks
Early Pin Planning
Early Pin Planning
Run I/O Assignm
Export Pin Assign
Pin Finder
Top View - Wire Bond
MAX 10 - 10M50DAF484C7G
Named: *
Edit:
Node Name | Direction | Location | I/O Bank | /REF Group | tter Locatic | 'O Standar | Reserved | rent Stren | Slew Rate | fferential P | ct Preserva
A[3] | Input | PIN_A14 | 7 | B7_N0 | PIN_A14 | 2.5 V | | 12mA...ult) | | |
A[2] | Input | PIN_A13 | 7 | B7_N0 | PIN_A13 | 2.5 V | | 12mA...ult) | | |
A[1] | Input | PIN_B12 | 7 | B7_N0 | PIN_B12 | 2.5 V | | 12mA...ult) | | |
A[0] | Input | PIN_A12 | 7 | B7_N0 | PIN_A12 | 2.5 V | | 12mA...ult) | | |
B[3] | Input | PIN_C12 | 7 | B7_N0 | PIN_C12 | 2.5 V | | 12mA...ult) | | |
B[2] | Input | PIN_D12 | 7 | B7_N0 | PIN_D12 | 2.5 V | | 12mA...ult) | | |
B[1] | Input | PIN_C11 | 7 | B7_N0 | PIN_C11 | 2.5 V | | 12mA...ult) | | |
B[0] | Input | PIN_C10 | 7 | B7_N0 | PIN_C10 | 2.5 V | | 12mA...ult) | | |
Cin | Input | PIN_F15 | 7 | B7_N0 | PIN_F15 | 2.5 V | | 12mA...ult) | | |
Cout | Output | PIN_C13 | 7 | B7_N0 | PIN_C13 | 2.5 V | | 12mA...ult) | 2 (default) | |
S[3] | Output | PIN_B10 | 7 | B7_N0 | PIN_B10 | 2.5 V | | 12mA...ult) | 2 (default) | |
S[2] | Output | PIN_A10 | 7 | B7_N0 | PIN_A10 | 2.5 V | | 12mA...ult) | 2 (default) | |
S[1] | Output | PIN_A9 | 7 | B7_N0 | PIN_A9 | 2.5 V | | 12mA...ult) | 2 (default) | |
S[0] | Output | PIN_A8 | 7 | B7_N0 | PIN_A8 | 2.5 V | | 12mA...ult) | 2 (default) | |
<<new node>>
All Pins


sim - Default
Instance | Design unit | Design unit type | Top Category
da_vhd_tst | da_vhd_ts... | Architecture | DU Instance
i1 | da(behavio... | Architecture | DU Instance
init | da_vhd_ts... | Process | -
always | da_vhd_ts... | Process | -
standard | standard | Package | Package
textio | textio | Package | Package
std_logic_1164 | std_logic_1... | Package | Package
std_logic_arith | std_logic_a... | Package | Package
std_logic_unsigned | std_logic_u... | Package | Package
Library
sim
Wave - Default
Msgs
A
B
Cin
Cout
S
1010
0101
1
1
0000
Now
100 ns
Cursor 1
50 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120



Conclusions.

In this practice we were able to observe the behavior of our type of adder, whose advantage is that it is faster because it has a shorter propagation time than other types and we were able to observe why this is so, in addition to learning a new type of adder to the previous ones seen.

References.

dev, andre. (2021). COMO HACER UN SUMADOR CON ACARREO ANTICIPADO (LogicCircuit) [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=bW_FbHchkW0

Wikipedia Contributors. (2023, July 11). *Carry-lookahead adder*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Carry-lookahead_adder

Prepbytes. (2023, February 20). *Carry Look Ahead Adder*. PrepBytes Blog. https://www.prepbytes.com/blog/computer-network/carry-look-ahead-adder/

(2018, June 19). Gatevidyalay.com. https://www.gatevidyalay.com/carry-look-ahead-adder/

Carry Look Ahead Adder. (2017, December). GeeksforGeeks; GeeksforGeeks. https://www.geeksforgeeks.org/carry-look-ahead-adder/