

JK LAKSHMIPAT UNIVERSITY

DIGITAL CIRCUIT AND SYSTEMS  
(EE1120)

Activity 10

4 – Bit Carry Look Ahead Adder using VHDL language.

Date : 14th March 2024

Name : Bobby Sharma

Roll no. = 2023BTECH023

# AIM: Design and Simulation of 4-bit carry look ahead adder (Structural Modelling) using VHDL language using Xilinx ISE Tool.

SOFTWARE REQUIRED: Xilinx ISE tool in your device.

THEORY: A 4-bit Carry Look-Ahead (CLA) adder is a type of digital circuit used to add two 4-bit binary numbers together. It's designed to reduce the time delay inherent in ripple carry adders by generating the carry signals for multiple stages simultaneously, rather than propagating them sequentially. This results in faster addition of numbers compared to traditional ripple carry adders.

* Binary Addition: Before diving into the CLA adder, it's important to understand binary addition. In binary addition, we add two binary numbers bit by bit, starting from the least significant bit (LSB) and propagating carries to higher-order bits as needed.
* Carry Propagation Delay: In a ripple carry adder, each stage waits for the carry out from the previous stage before it can produce its own carry out. This introduces a propagation delay which increases linearly with the number of bits.
* Carry Look-Ahead Logic: In a Carry Look-Ahead adder, the carry out for each stage is calculated independently of the previous stage's carry out. This is achieved using a set of logic gates that predict the carry based on the inputs, rather than waiting for the carry to propagate.
* Generate and Propagate Signals: The Carry Look-Ahead logic typically involves two key signals: Generate (G) and Propagate (P). The Generate signal indicates whether a carry must be generated at a particular stage (i.e., whether both input bits are 1), while the Propagate signal indicates whether a carry from a previous stage can propagate to the current stage (i.e., whether at least one input bit is 1).
* Carry Generation: The Generate signal for each stage is calculated by taking the logical AND of the input bits at that stage. This indicates whether a carry must be generated at that stage.
* Carry Propagation: The Propagate signal for each stage is calculated by taking the logical OR of the input bits at that stage. This indicates whether a carry from a previous stage can propagate to the current stage.
* Carry Look-Ahead Logic: Once the Generate and Propagate signals are computed for each stage, the carry out for each stage can be determined using these signals and the carry in. This is typically done using logical operations such as AND, OR, and XOR.
* Sum Generation: The sum bit for each stage is computed using the XOR operation between the input bits and the carry in.
* Final Sum and Carry Out: The final sum bits and the carry out from the most significant stage are obtained by propagating the carry generated by the individual stages.

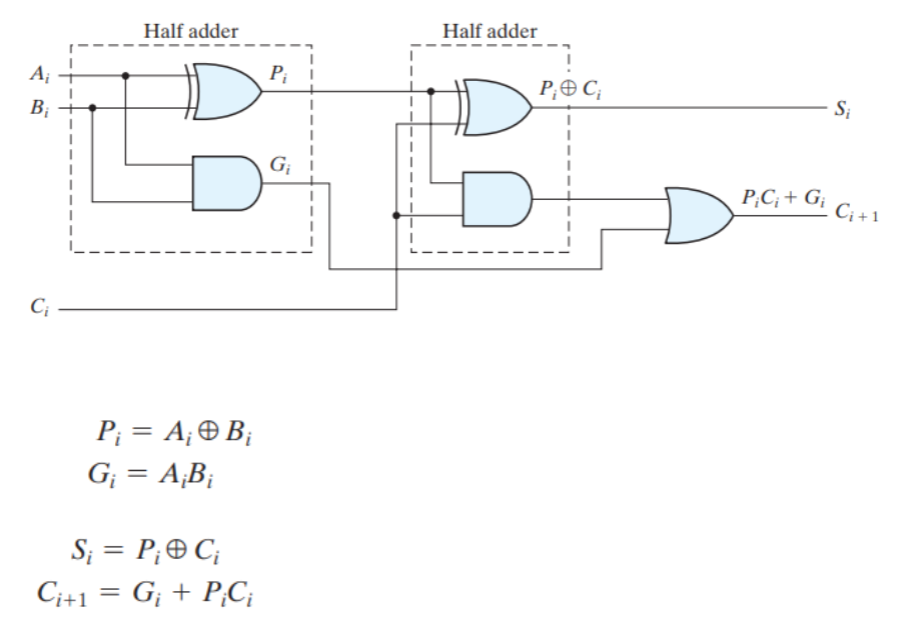


Figure 1

OBSERVATION: The observed outputs of all the basic gates are as follows:

* FULL ADDER:

VHDL Code: RTL Diagram:

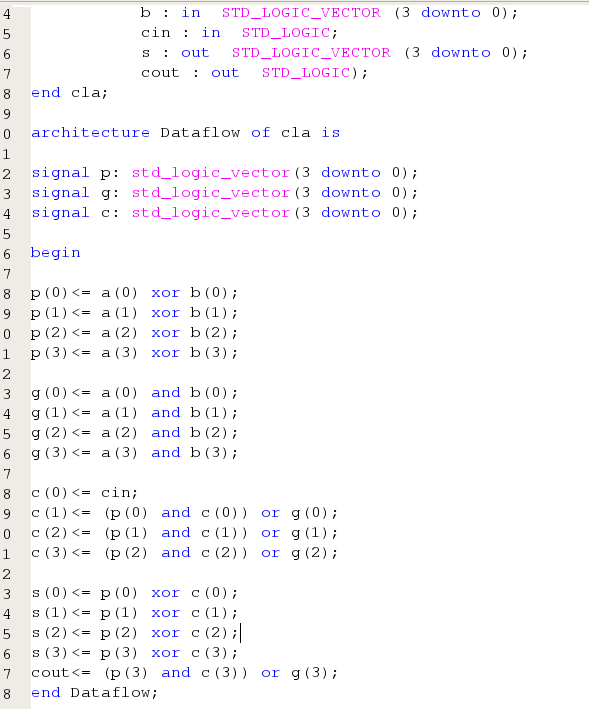
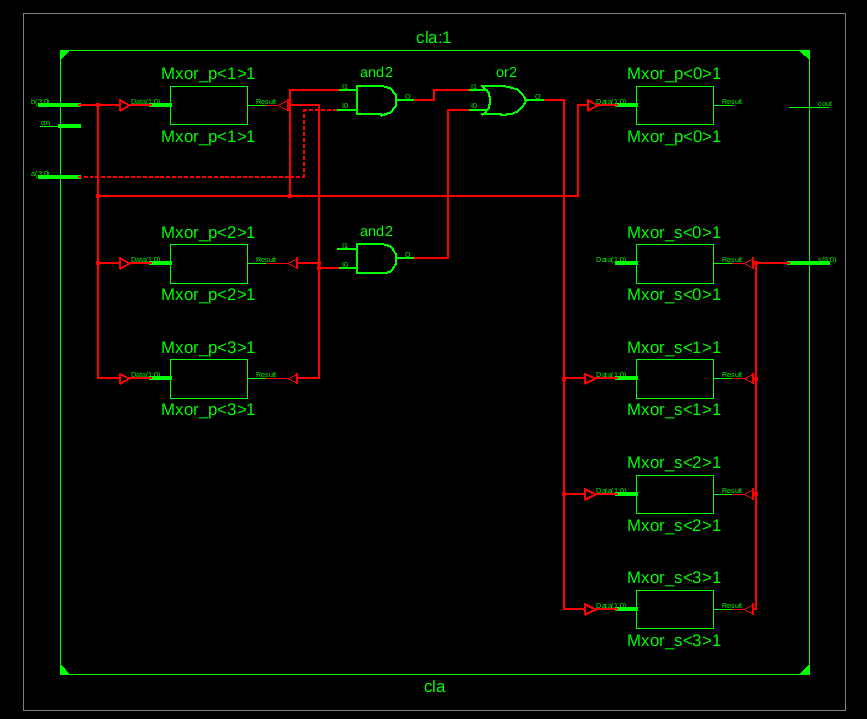
 

Figure 2 Figure 3

Test Bench Code:

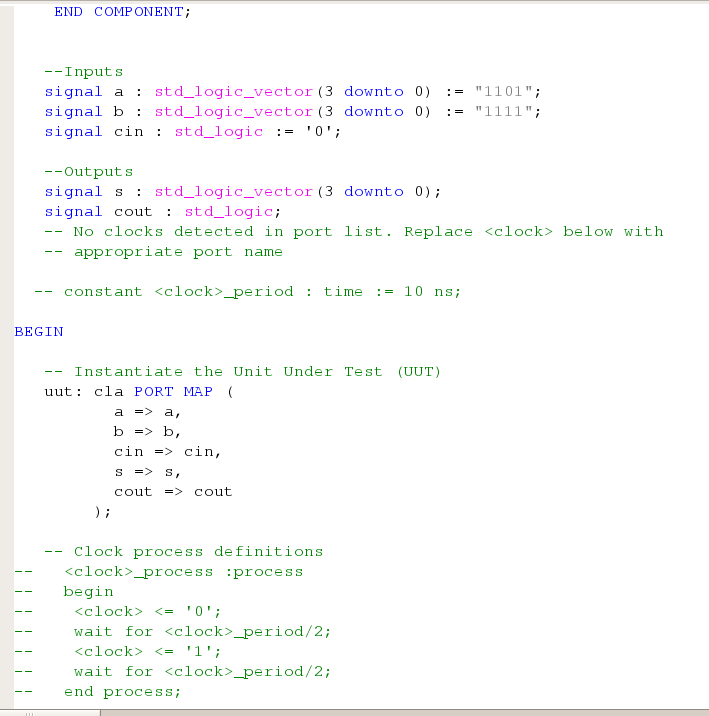


Figure 4

Waveform:

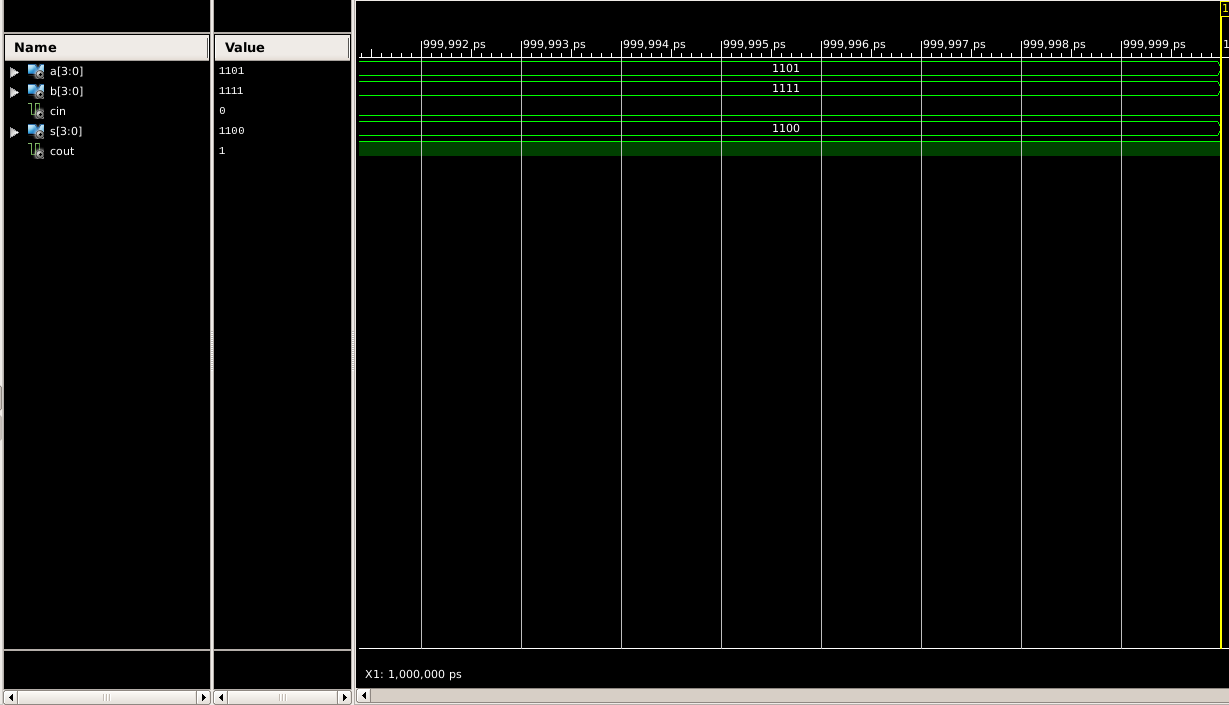


Figure 5

Here the yellow line in figure 5 represents the input (a=1111, b=1101) and output(sum = 1100 and carry = 1 ). The truth table for a 4-bit adder lists all possible combinations of inputs (A, B, and Cin) and the corresponding outputs (Sum and Cout).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | Cin | S | Cout |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Table 1

# RESULT: We have concluded the truth table of 4-bit carry look ahead adder using VHDL language in Xilinx ISE Tool.

APPLICATION IN DAILY LIFE:

* **Computers and Mobile Devices**: Inside the CPU (Central Processing Unit) of computers and mobile devices, arithmetic and logic units (ALUs) utilize adders for performing arithmetic operations. When you add numbers in a spreadsheet, play a game, or browse the web, you're indirectly using the adders within the CPU.
* **Financial Transactions**: Banking systems, payment gateways, and financial institutions use digital circuits for processing financial transactions. These systems involve arithmetic operations such as addition for calculating balances, transaction amounts, etc.
* **Communications**: In communication systems, adders are used for error correction, data compression, encryption, and various signal processing tasks. For instance, in wireless communication, signal processing circuits use adders to process incoming signals for decoding and analysis.
* **Digital Media**: Digital signal processing (DSP) in devices like cameras, audio recorders, and music players heavily rely on arithmetic operations. Adders are used in these devices for tasks such as image processing, audio filtering, and compression/decompression.