life.augmented

STMicroelectronics

Computer Arithmetic Training

Nov.2024

LAB 1

Designing and Testing Fast Adders

Instructor:
Ahmed Abdelsalam

Due to:

Sat 14 Dec.2024,11:59PM

Objective

The objective of this lab is to ensure we have a good understanding of the architecture and operation of various types of fast adders, and to compare them in terms of cost and delay using Verilog/VHDL. Additionally, we will verify their functionality through a testbench.

Prerequisites

- Good understanding of digital logic design.
- Familiarity with VHDL or Verilog HDL.
- Good Understanding for CLA and PP Adders.

Notes for submission

1. Upload Verilog/VHDL files to GitHub (or compress them as Zip file and upload to Drive)
2. Report (PDF format) contains all the requirements specified below

Requirements

- **RTL** the Verilog/VHDL code implements the assigned question and comment your design well upload all the code files and attach the link to the report
- **Simulation** modify the Testbench from Lab0 to fit with the new width of I/O and run simulation attach screenshots to show the output or waveform
- **FPGA** run synthesis and Implementation using VIVADO Toolchain targeting VCU118 board and report the Utilization and critical path delay calculation
- **[Bonus] Parametrized RTL** write generic code can be used to generate N-bit adder (where N is Parameter/Generic) of type similar to the Type in the Assigned Question
- **[Bonus] ASIC** run Synthesis and PnR for the design using any technology you have access to it. You need to specify the technology used in report. You also need to report Area, Delays and power information and attach screenshots to report.

Question1

You are asked to design a 64-bit hierarchical Carry look ahead adder using 4-bit building unit through the following the steps below and attach all the required captures and codes

Step 1: 4-bit CLA (carry network)

Definition: design a 4-bit Carry look ahead carry network which use generate propagate signals in addition to carry in and compute the all the carry signals C_1 to C_4.

The circle below describes how to compute generate, propagate signals from x and y inputs

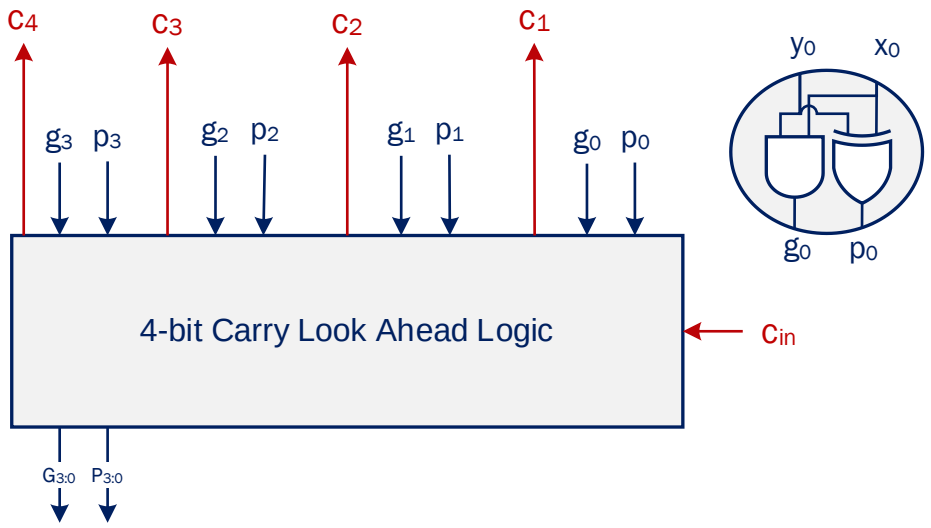

Figure 1

- **Inputs** { $c_{in,}$ g_0, p_0, g_1, p_1, g_2, p_2, g_3, p_3 }
- **Outputs** {c_1, c_2, c_3, $c_{4,}$ $G_{0:3}$, $P_{0:3}$ }

Step 2: 64-bit hierarchical Carry look ahead adder using 4-bit CLA logic as building block

Definition: Hierarchical Carry Look Ahead Adder uses smaller size of carry look Ahead logic nested to compute all he carry locations and avoid the growth in the gate sizes and number of inputs in the figure below a full description how to build 16-bit adder using 2 levels of 4-bit carry look ahead module to build carry network then use the carry signals with propagate signal to compute the sum trace back the schematic below and convince yourself about how the operation done then use the same idea one more level to build 64-bit adder.

Nothing will change in *SUM* or *G,P generation logic* only the carry network will change by adding one more level

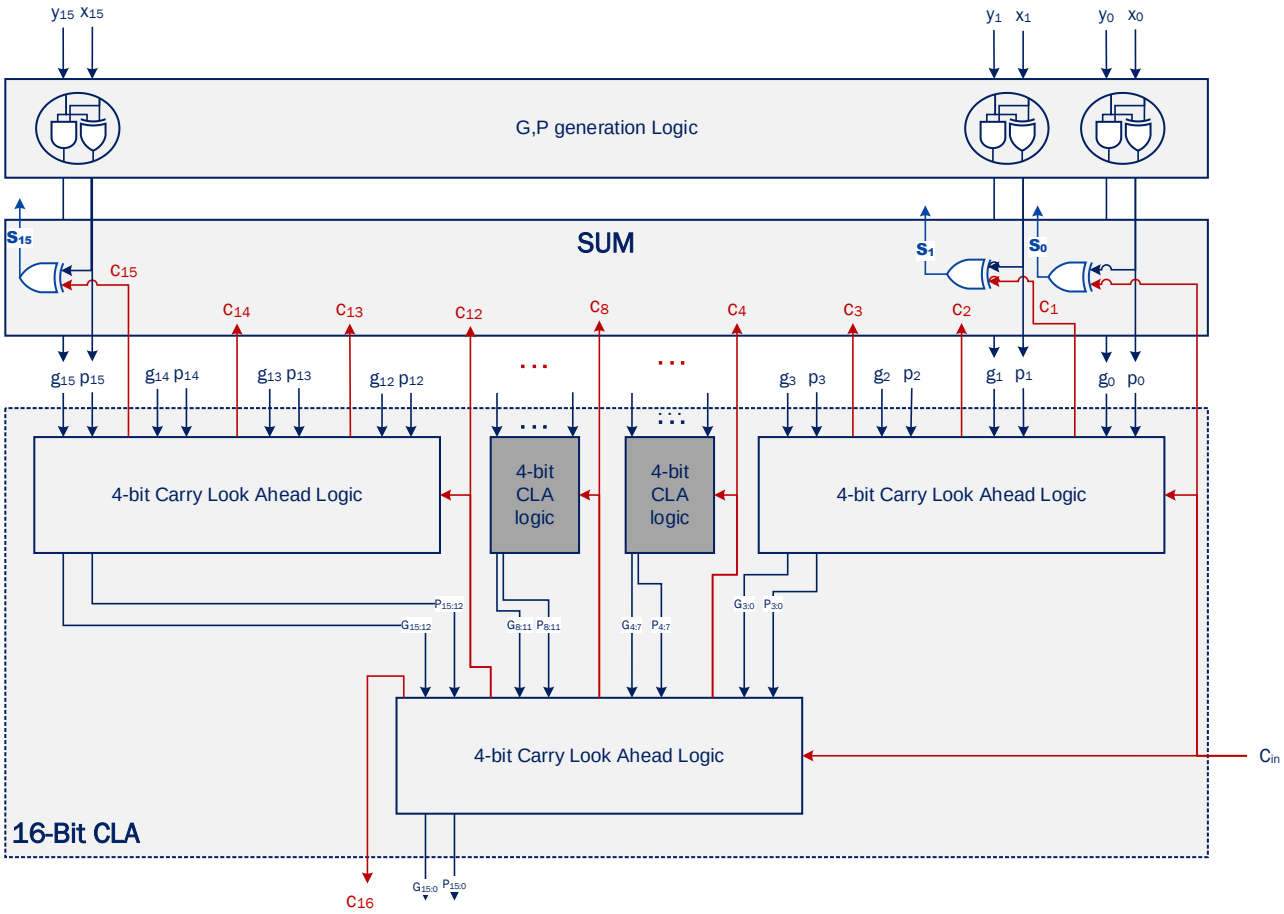

Figure 2

- **Inputs** {*A*, *B*, *Cin*} **(*A*,*B* are 64-bits inputs and *Cin* is 1-bit)**
- **Outputs** {*Sum*, *Cout*} **(*sum* is 64-bits output and *Cout* is 1-bit)**

Question 2

You are asked to design a 64-bit parallel prefix adder using Brent-kung Carry network through the following the steps below and attach all the required captures and codes

Fig.3 below show an example how to implement a size-8 prefix sum network using brent-kung Architecture each layer is encoded with different color to help you extract the pattern

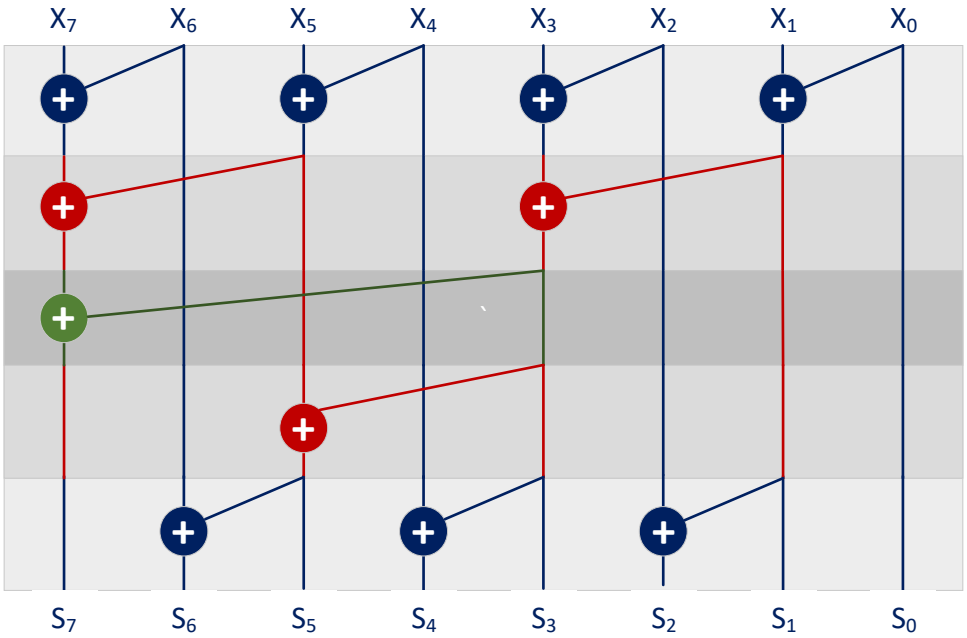

Figure 3

Fig.4 show the steps how the example in fig.3 is build step by step first layer one which contains (8-1 adders) and change the problem from 8-input problem to a smaller one with only 4-inputs then we apply the same idea by using (4-1 adders) and split the problem to half please spend some time to convince yourself and make sure you understand the pattern very well then start to map this into a carry network of 64-bit adder by removing each + sign with a carry operator you will find the implementation of carry operator below in fig.4

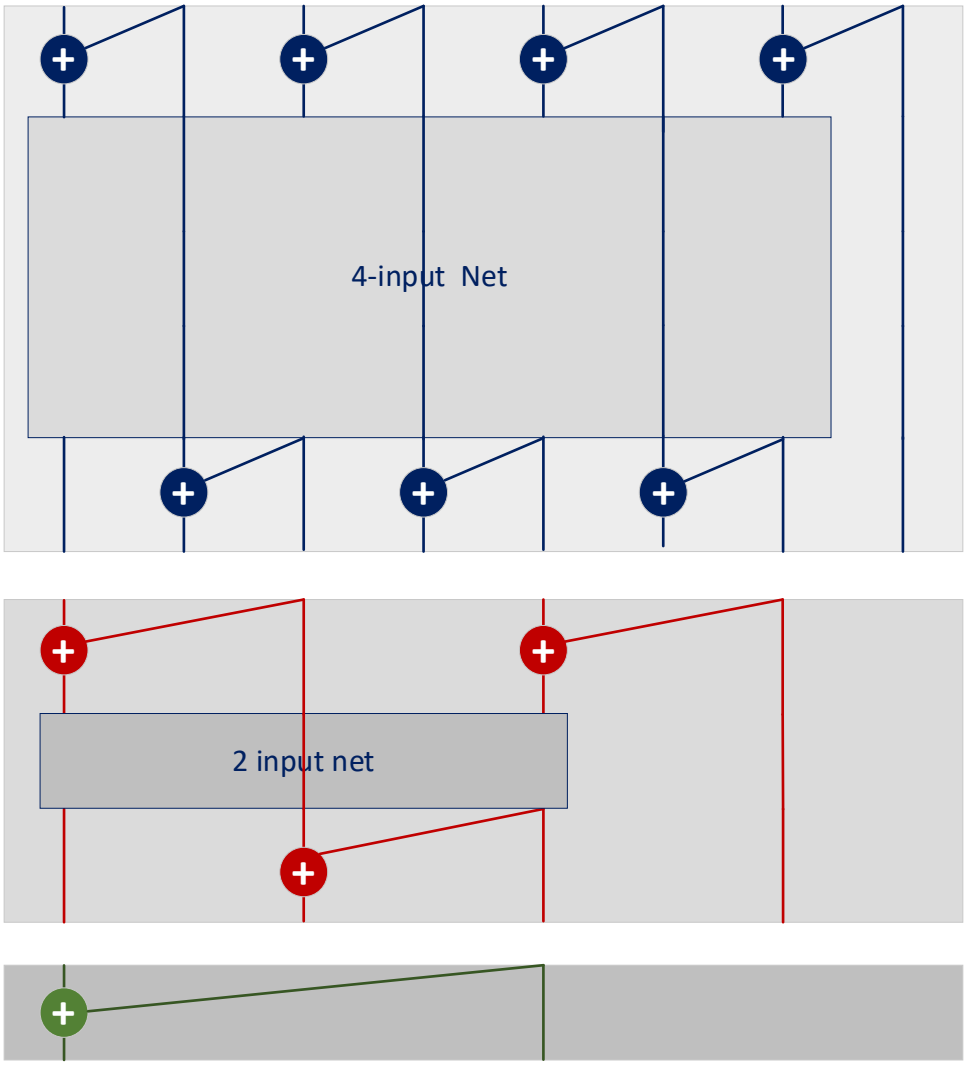
4-input Net
2 input net
g1 p1
g0
p0
g1:0
p1:0
g4:7
p4:7
g0:3
p0:3
g7:0
p7:0

Figure 4

Question 3

You are asked to design a 64-bit parallel prefix adder using Ladner–Fischer Carry network through the following the steps below and attach all the required captures and codes

In fig.5 below you will find the generic pattern how to build a Ladner-Fischer sum prefix network accumulate the most sum from the lower half to all the sums in the upper half then you transformed the issue into 2 smaller prefix sum networks of half the size and repeat until reach prefix sum of size 2 study this image and try to apply this pattern with the size 64 to build a carry network.

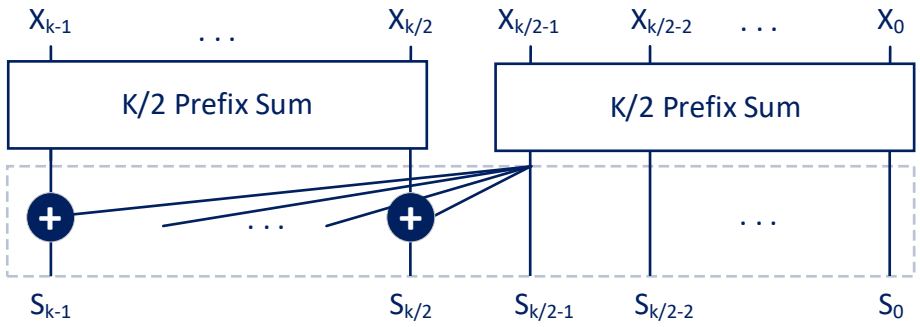

Figure 5

Fig. 6 below shows an example of Ladner–Fischer prefix sum network with size of 8 you need to study this example and convince yourself before applying to the given problem.

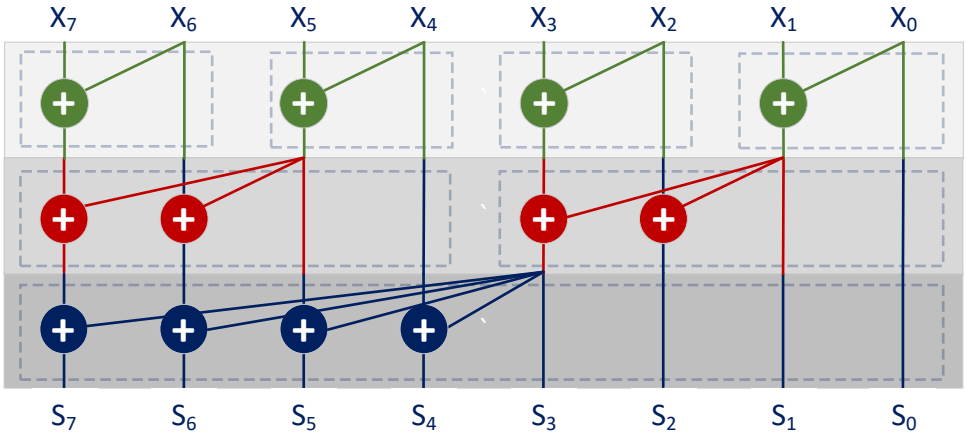

Figure 6