N-Bit Saturated Math Carry Look-ahead Combinational Adder Design in VHDL

N-Bit Saturated Math Carry Look-ahead Combinational Adder Design in VHDL

Features

The following design topics are covered:

• Methodology and logic behind look ahead carry combinational adders and saturated mathematics
• Using the *generate* statement to scale the logic slices up with a structural combinational approach
• Benchmarking for performance with a generic FPGA supplier's native toolset
• Simulation of the N-Bit Saturated Math Carry Look-ahead Combinational Adder using a test-bench

Introduction

This VHDL module uses the basic Boolean equations derived from a binary carry look-ahead adder and structurally builds the equations. The VHDL generate statement and a generic variable are used to scale the slices up to the user's desired word width.

The saturation part of the adder ensures glitch free performance when used in DSP and control related designs. The adder will saturate the output by identifying an overflow condition and assigning the maximum positive or negative full scale value to the output.

A Xilinx Spartan 6 development kit along with the ISE v13.1 development environment was used to develop the adder, although one is not limited to this particular toolset. A multitude of development kits from Xilinx, Microsemi, Lattice, and Altera are available from the Digi-Key Corporation website at the following link found here.

Background

The carry look-ahead adder works by evaluating the two words being added to identify carry generate and carry propagate bits. These bits will determine all the places where a carry will occur in a combinational fashion. The addition is done at the same time, slice by slice, as each carry bit is available. The saturation is determined by an overflow flag and the most significant bit of the temporary sum.

Application

Building the Circuit

A VHDL generate statement is the perfect solution for building the Boolean equations for each slice. This allows scaling according to the user's requirements. A generic variable is used in the VHDL module to determiene the addend word width and the generate statement builds the structural logic accordingly. The initial carry-in is initialized to '0' and subsequent carries are dependant on the each bit of the two addends from least significant bit to most significant bit. Each slice is pieced together to build the combinational circuit with speed being limited by only gate delays. The basic carry look-ahead adder slice can be seen below in figure 1.

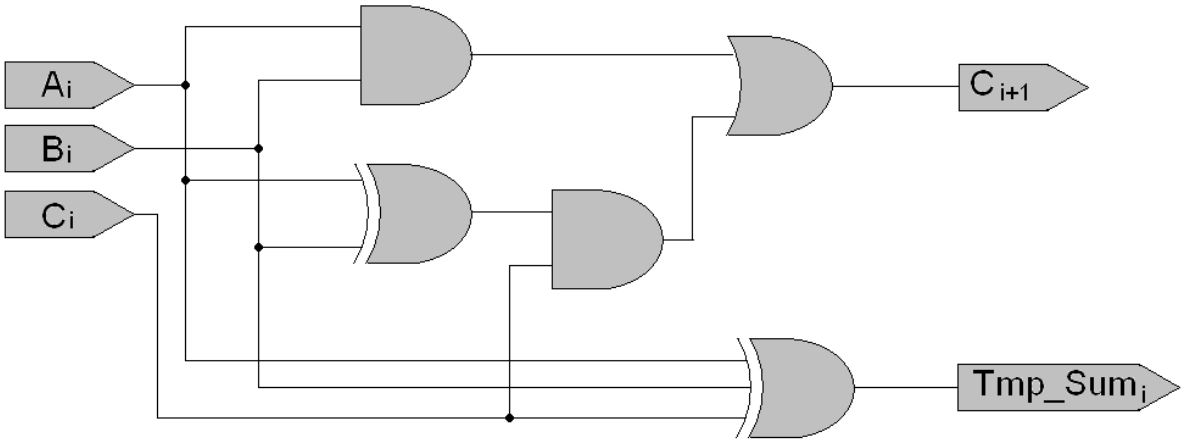

Figure 1 Basic Carry Look-Ahead Adder Slice

The saturation is determined by an overflow condition. This condition is determined by evaluating the exclusive or of the carry-in and carry-out of the most significant bit, the two most significant bits found in the carry array. If an overflow occurs, the most significant bit of the temporary sum will determine if the final sum should be saturated for the maximum positive value or maximum negative value. A '1' would indicate the overflow caused a positive value to flip negative, so we saturate the final sum value positive. A '0' would indicate the opposite, so we saturate the final sum value negative. A 3-slice implementation of the adder can be seen in figure 2 below. The output sum saturation can be seen in figure 3 below.

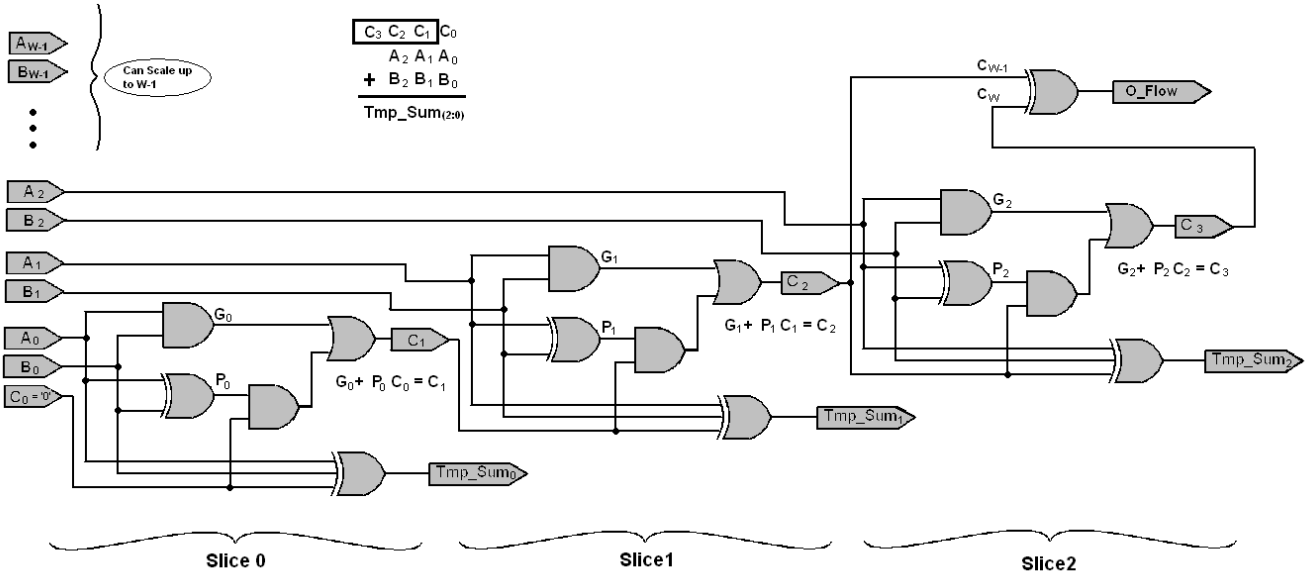

Figure 2 Three-Slice Implementation with Optional Scaling up to W-bit Addend Width

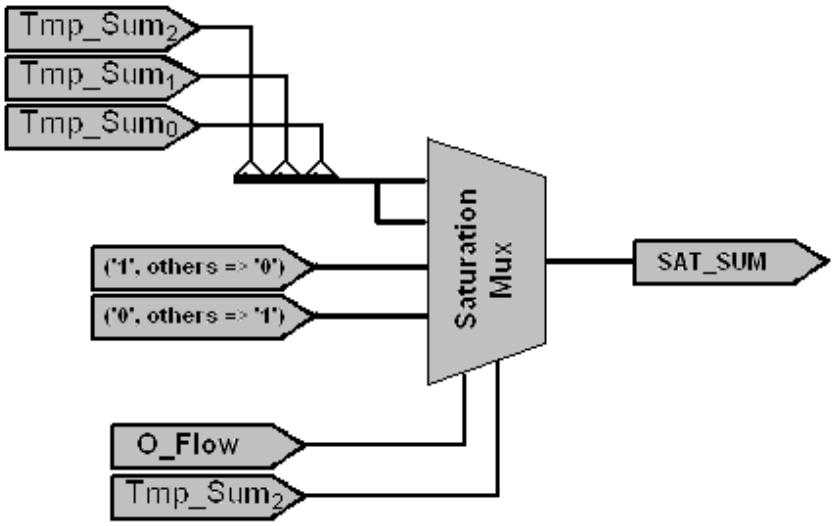

Figure 3 Three-Slice Implementation with Optional Scaling up to W-bit Addend Width

Benchmarking

To benchmark the design we can drive the combinational adder between two n-bit clocked flip-flops. This will determine how fast we can run the adder, given a constant addend word width. At 64 bit, this design was able to run at 100 MHz using a Spartan 6 device at a -3 speed grade, each result was given in less than 10 nano seconds. The design was constrained at 10 ns with no further optimizations selected. One will experience different results given a different FPGA or even the same FPGA at a different speed grade. Optimizations can also be made using a given FPGA supplier's native toolset by properly constraining the design and optimizing for speed over area. A top level diagram for the test schematic can be seen below in figure 4.

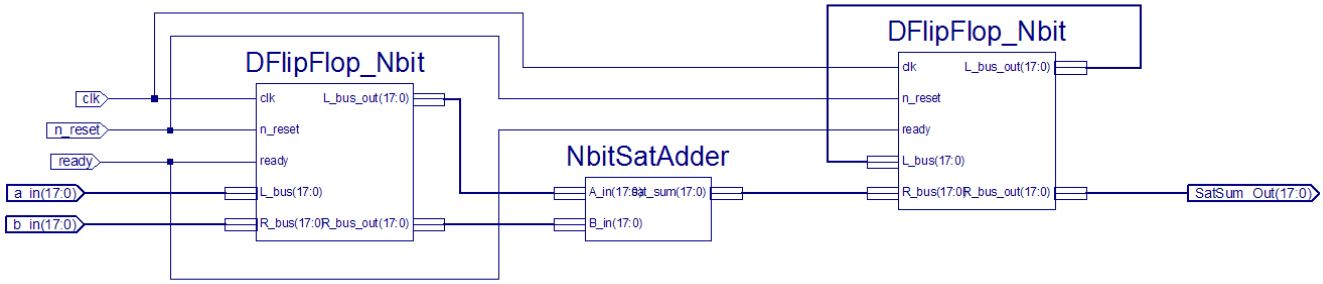

Figure 4 Top Level Test Schematic for Benchmarking the Combinational Adder

Simulating the Circuit

Various HDL simulators, such as ModelSim or Xilinx's Isim, can be used in verifying the circuit's output. Isim was used in this particular application. A test-bench was created in the Xilinx ISE v13.1 development environment. The test-bench has two processes, one for the clock and one for the input signals. The output can be seen in figure 5 below. The red and blue circles identify where the output has been saturated due to an overflow condition. The test-bench for the design is included with the other design files.

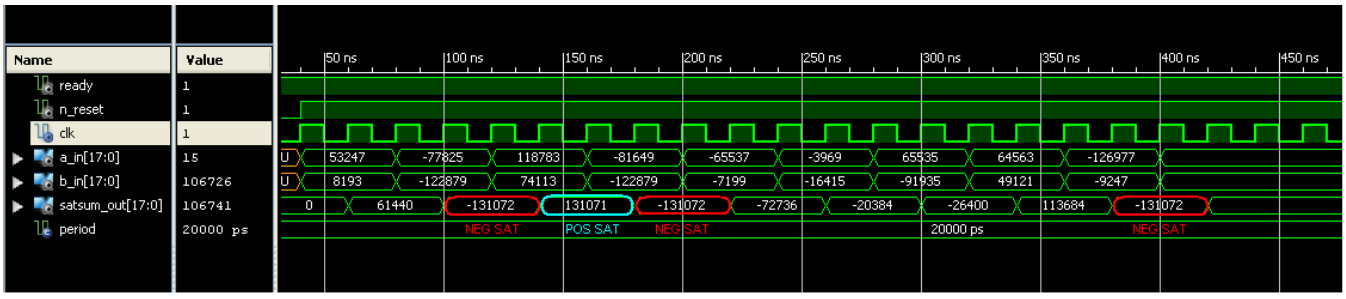

Figure 5 Isim Test-Bench Output

Capabilities, Limitations, and Alterations

This design can be used with any FPGA development platform with varying performance. As the word width gets large the amount of hardware used also gets quite large. The Xilinx ISE v13.1 was used in conjunction with a Spartan 6 development platform to develop and benchmark this VHDL module. The internal Xilinx Isim simulation software was used to simulate and verify the design.

Conclusion

The NbitSatAdder.vhd file can be modified and instantiated in any design where speed and output saturation would be of importance. This module works well when used in cascaded adder circuits such as in a parallel FIR filter circuit. When used in slower FPGAs, an N-bit, clocked, flip-flop works well to synchronize the circuit when attempting to meet timing requirements. A multitude of development kits from Xilinx, Microsemi, Lattice, and Altera are available from the Digi-Key Corporation website at the following link found here.

Additional Information

Further design support, product tutorials, application notes, users guides and other documentation can be found on any of the four major FPGA suppliers' websites:

Xilinx applications at http://www.xilinx.com/support/index.htm#nav=sd-nav-link-156334&tab=tab-sd

Microsemi cSOC group's website at http://www.actel.com/techdocs/default.aspx.

Lattice at http://www.latticesemi.com/dynamic/index.cfm?fuseaction=view_category&source=topnav

Altera support at http://www.altera.com/support/spt-index.html

Appendix:

The complete LookAheadCarryStaurationAdder.xise project can be downloaded from the Digi-Key, eewiki.net website under the Programmable Logic section

VHDL for Adder => NbitSatAdder.vhd
VHDL for Flip Flop => NBitDFF.vhd
VHDL for Test Bench = > Adder_test.vhd
Schematic File for Xilinx ISE v13.1 => Top_Level.sch

Feedback for Our Sponsor

Please take a few seconds to help us justify the continued development and expansion of the eewiki.

Click on one of our Digi-Key links on your way to search for or purchase electronic components.

Is the eewiki helpful? Comments, feedback, and questions can be sent to eewiki@digikey.com.