Abstract

This report describes the design and evalation of a 32-bit Carry Select Adder (CSA), evaluating how it compares to the conventional Carry Ripple Adder (CRA) in timing, complexity and implementation area.

1 Circuits and Coding

Carry Ripple Adders (CRA) are an example of iterative logic where the carry out of each full adder comprises the intercell signal. Iterative logic circuits are easy to implement and test, and subsequently the first adders were built this way. A downside of iterative logic is the time for the intercell signal to propagate to the last iterative unit.

Carry Select Adders (CSA) are fast adders which aim to reduce propagation delay by performing much of the computation in parallel. A schematic of a 32-bit CSA is detailed in Figure 1.

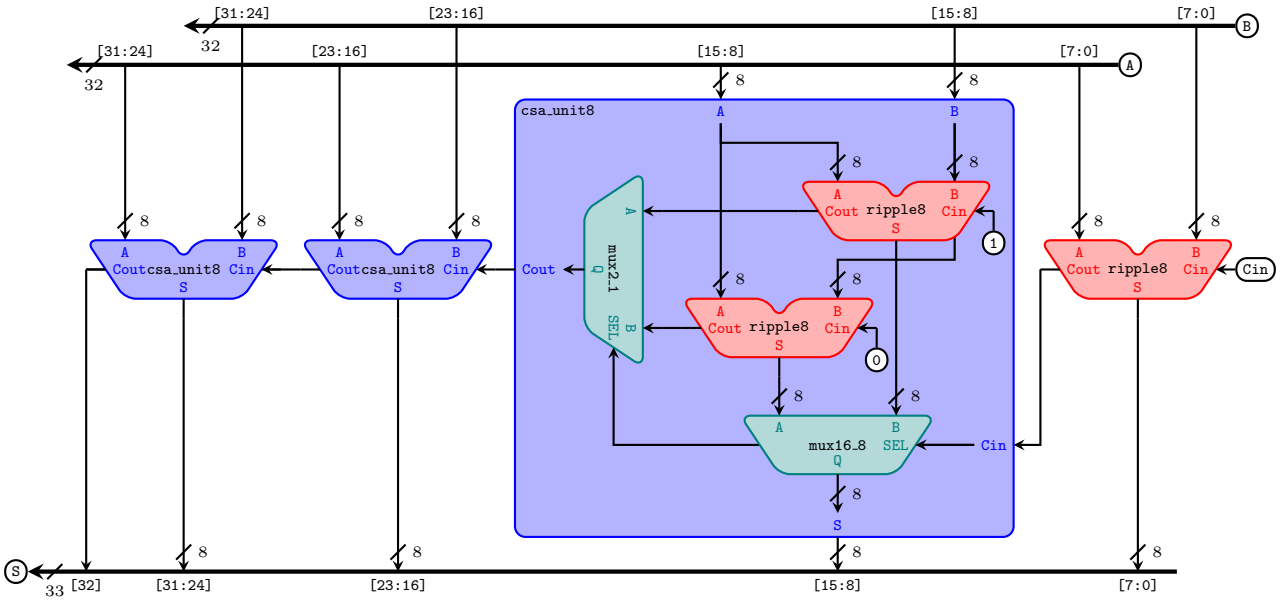


Figure 1: Diagram of a 32-bit carry select adder, with 8-bit select units

Figure 2 illustrates the hierarchy of files used to build the 32-bit CSA in Figure 1 and a 32-bit carry ripple adder for comparison. Both The CSA and CRA were implemented in Altera® Quartus® II and shared as much VHDL as possible to avoid discrepencies in implementation efficiency. Only the top-level files used the Quartus® II block schematic file format.

Listings 1 and 2 include the logic expression used to implement the `full_adder` and `mux2_1` design entities.

```
17 ARCHITECTURE full_adder_architecture OF full_adder IS
18 BEGIN
19     Cout <= ((A xor B) and Cin) or (A and B);
20     S <= (A xor B) xor Cin;
21 END;
```

Listing 1: Architecture declaration in `full_adder.vhd`

```
15 -- architecture
16 ARCHITECTURE mux2_1_architecture OF mux2_1 IS
17 BEGIN
18     Q <= (A and not SEL) or (B and SEL);
19 END mux2_1_architecture;
```

Listing 2: Architecture declaration in `mux2_1.vhd`

The 16-to-8 multiplexer was constructed using eight parallel `mux2_1` components, mapped to the elements of `std_logic_vector(7 downto 1)` type ports. A similar approach was used to create the 8-bit ripple adder out of `full_adder` components, although the carry in and carry out were mapped to elements of `std_logic_vector(6 downto 0)` type intercell signals. Signals

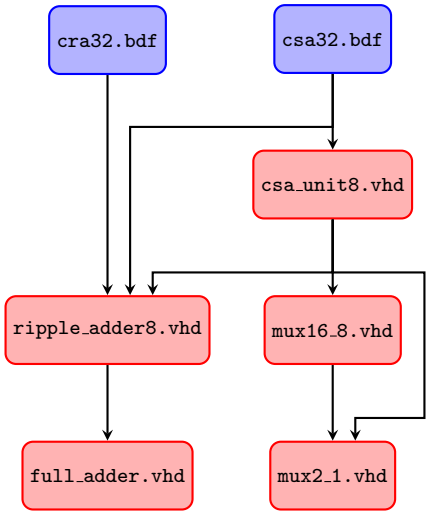
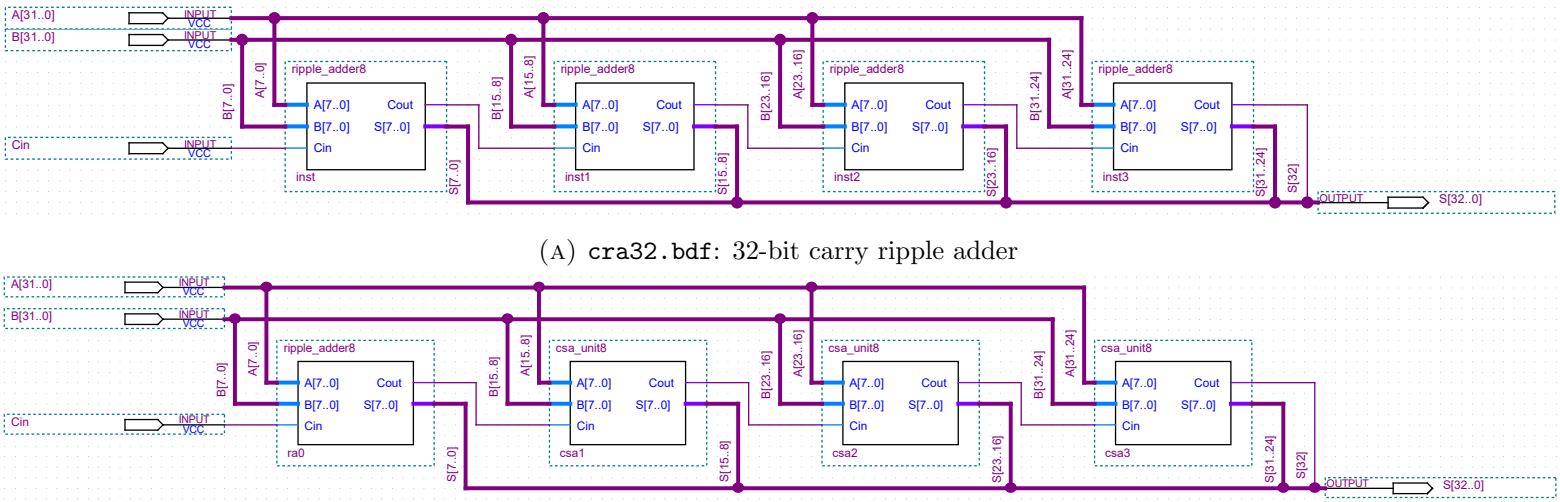


Figure 2: Hierarchy of entity declarations

were also used to represent the candidate sum and carry out values from the two 8-bit ripple adders detailed in the `csa_unit8` cells in Figure 1.

The top-level design entities shown in the schematics in Figure 3 were built out of `ripple_adder8` and `csa_unit8` components.



(a) `cra32.bdf`: 32-bit carry ripple adder

(b) `csa32.bdf`: 32-bit carry select adder, 8-bit select units

Figure 3: Block diagram schematics for the top-level entities of each project

2 Speed and Operation

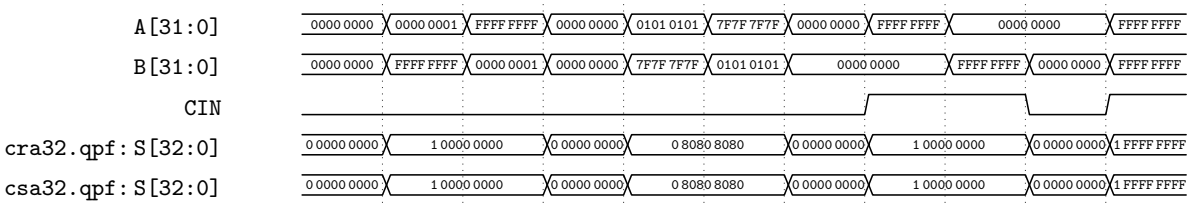


Figure 4: Functional simulation output, 100 % coverage

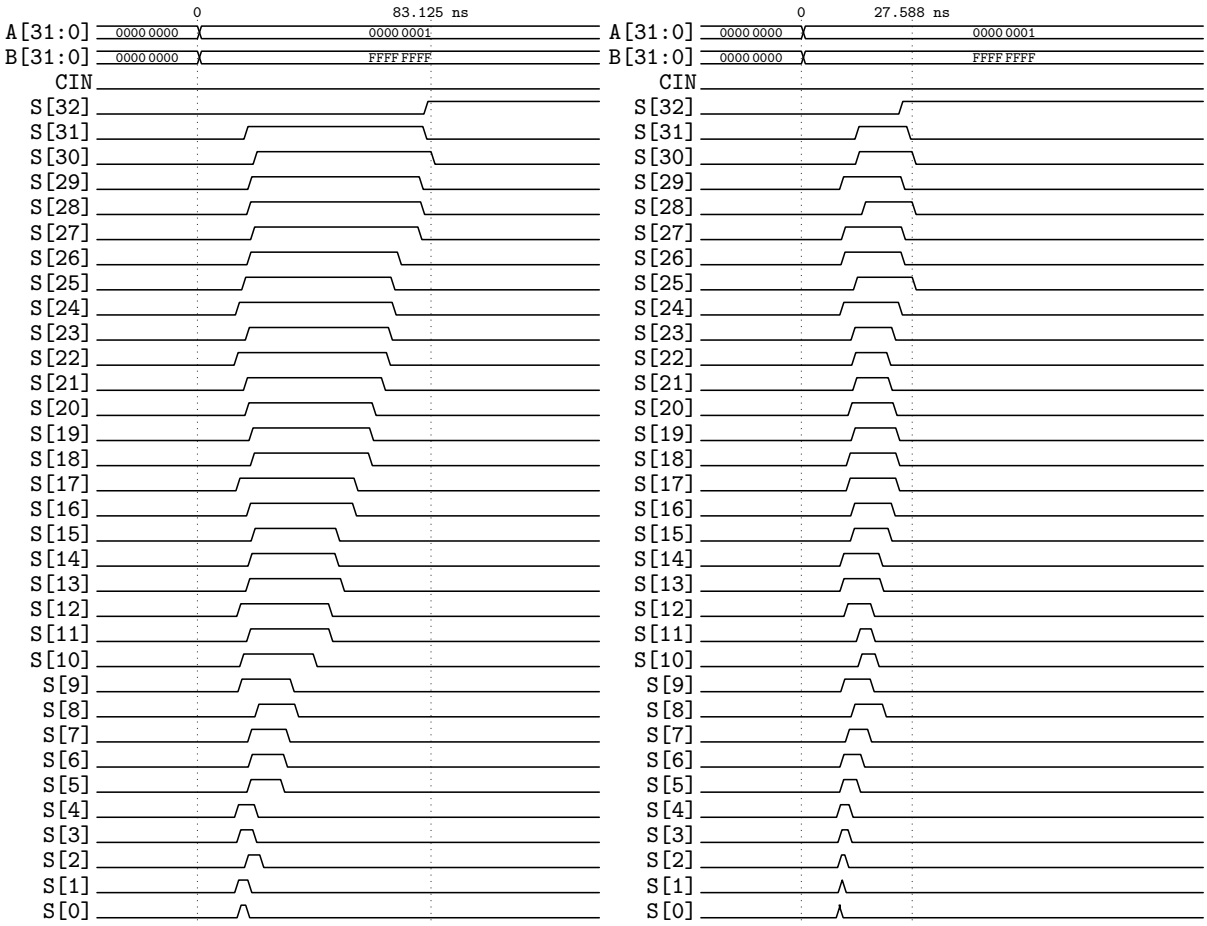


FIGURE 5: Worst case timing simulation output, compared

The 32-bit addition is split into 8-bit units which are executed in parallel. In each carry select unit, short ripple adders compute sum candidates for both possible carry in values. Simultaneously, the first intercell carry signal is computed by the inital adder. The intercell carry signals then propagate through the multiplexors selecting the correct sum and carry out candidates, rather than through the adders.

The propagation delay of the CSA is only as long as the 8-bit ripple adder and 3 multiplexors, much faster in theory than the propagation delay of a 32-bit ripple adder.

Apparent drawbacks of the CSA compared to the CRA include the increased complexity, size and power requirements due to duplicating most of the full adders and including multiplexors.

3 FPGA Implementation