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.

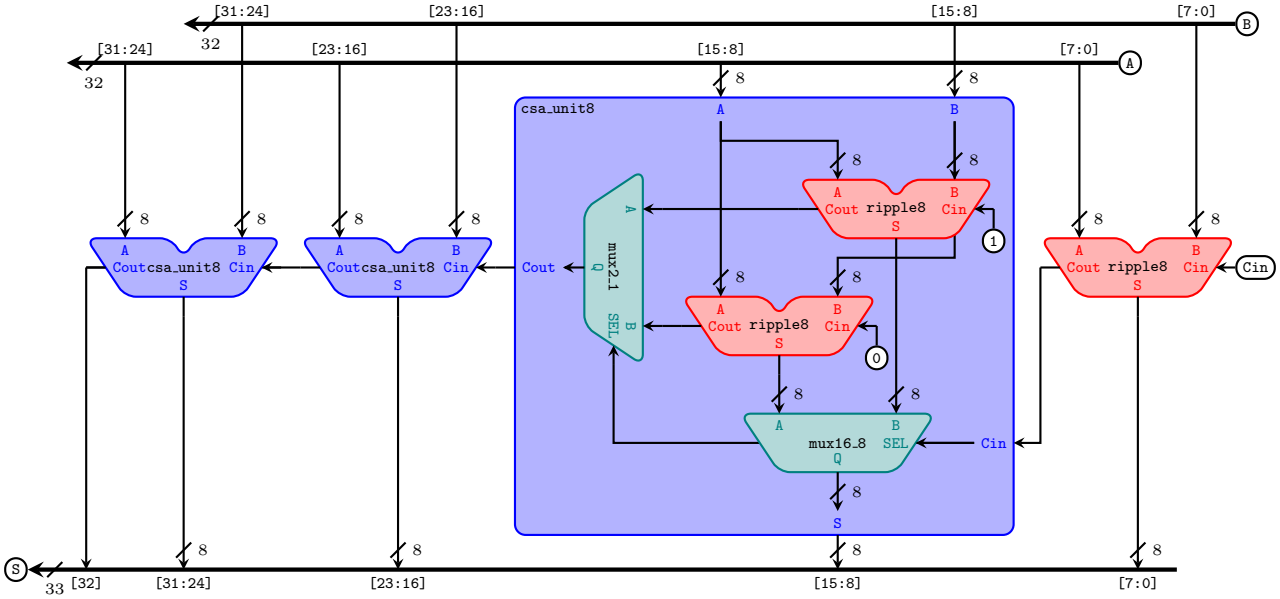


Figure 1: Diagram of a 32-bit carry select adder.

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`

```
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 multiplexor was constructed using eight parallel `mux2_1` components, mapped to the elements of `std_logic_vector(7 downto 1)` type ports. A similar approach in Listing 3 shows how intercell signals were used to connect `full_adder` components together to build the 8-bit ripple adder design entity.

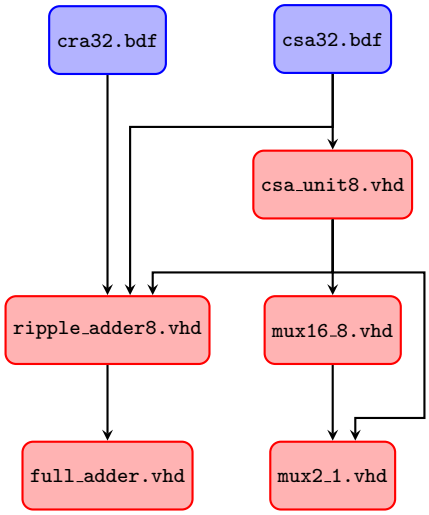


FIGURE 2: Hierarchy of entity declarations.

Signals were also used to represent the candidate sum and carry out values from the two 8-bit ripple adders detailed in the unit cells in Figure 1. The VHDL implementation of the CSA unit cell is described in Listing 4.

```
26      SIGNAL C : std_logic_vector(6 downto 0);
```

```
27  BEGIN
28      fa0 : full_adder PORT MAP ( A => A(0),
29                                  B => B(0),
30                                  Cin => Cin,
31                                  Cout => C(0),
32                                  S => S(0));
33      fa1 : full_adder PORT MAP ( A => A(1),
34                                  B => B(1),
35                                  Cin => C(0),
36                                  Cout => C(1),
37                                  S => S(1));
```

```
63      fa7 : full_adder PORT MAP ( A => A(7),
64                                  B => B(7),
65                                  Cin => C(6),
66                                  Cout => Cout,
67                                  S => S(7));
68  END;
```

LISTING 3: Snippet showing one of eight chained `full_adder` components mapped in `ripple_adder8.vhd` architecture declaration.

```
44      SIGNAL S0 : std_logic_vector(7 downto 0);
45      SIGNAL Cout0 : std_logic;
46      SIGNAL S1 : std_logic_vector(7 downto 0);
47      SIGNAL Cout1 : std_logic;
```

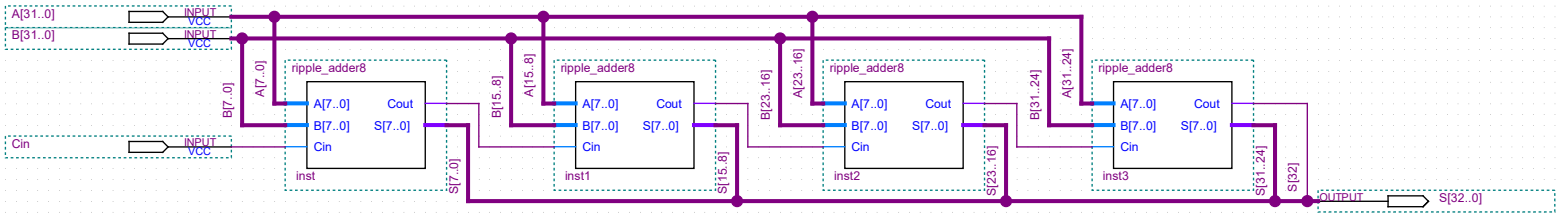
```
48  BEGIN
49      ra0 : ripple_adder8 PORT MAP (  A => A,
50                                      B => B,
51                                      Cin => '0',
52                                      Cout => Cout0,
53                                      S => S0);
54      ra1 : ripple_adder8 PORT MAP (  A => A,
55                                      B => B,
56                                      Cin => '1',
57                                      Cout => Cout1,
58                                      S => S1);
59      smux : mux16_8 PORT MAP (A => S0,
60                               B => S1,
61                               SEL => Cin,
62                               Q => S);
63      cmux : mux2_1 PORT MAP ( A => Cout0,
64                               B => Cout1,
65                               SEL => Cin,
66                               Q => Cout);
67  END;
```

LISTING 4: Snippet showing how the two ripple adders were mapped to the intermediate candidate signals before being selected using multiplexors and the carry in port in `csa_unit8.vhd` architecture declaration.

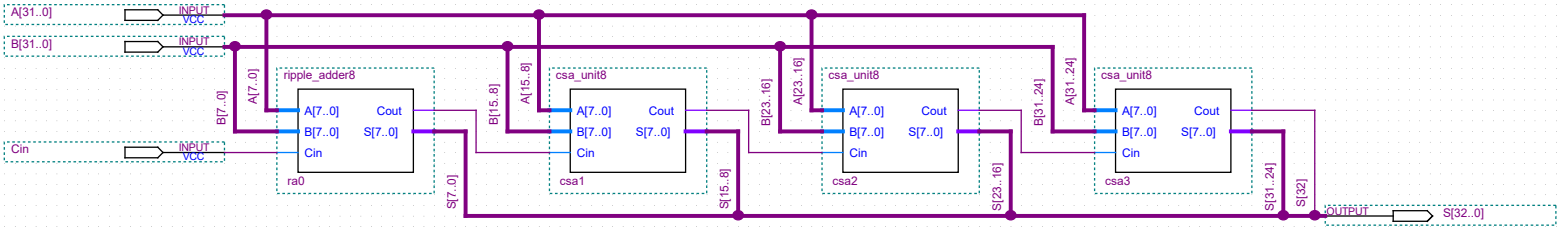
The top level design entities are shown in schematic in Figure 3

2 Speed and Operation

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,



(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.

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