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 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 are a fast adder that aims to reduce the propagation delay by performing much of the computation in parallel. A schematic of a 32-bit CSA is detailed in Figure 1.

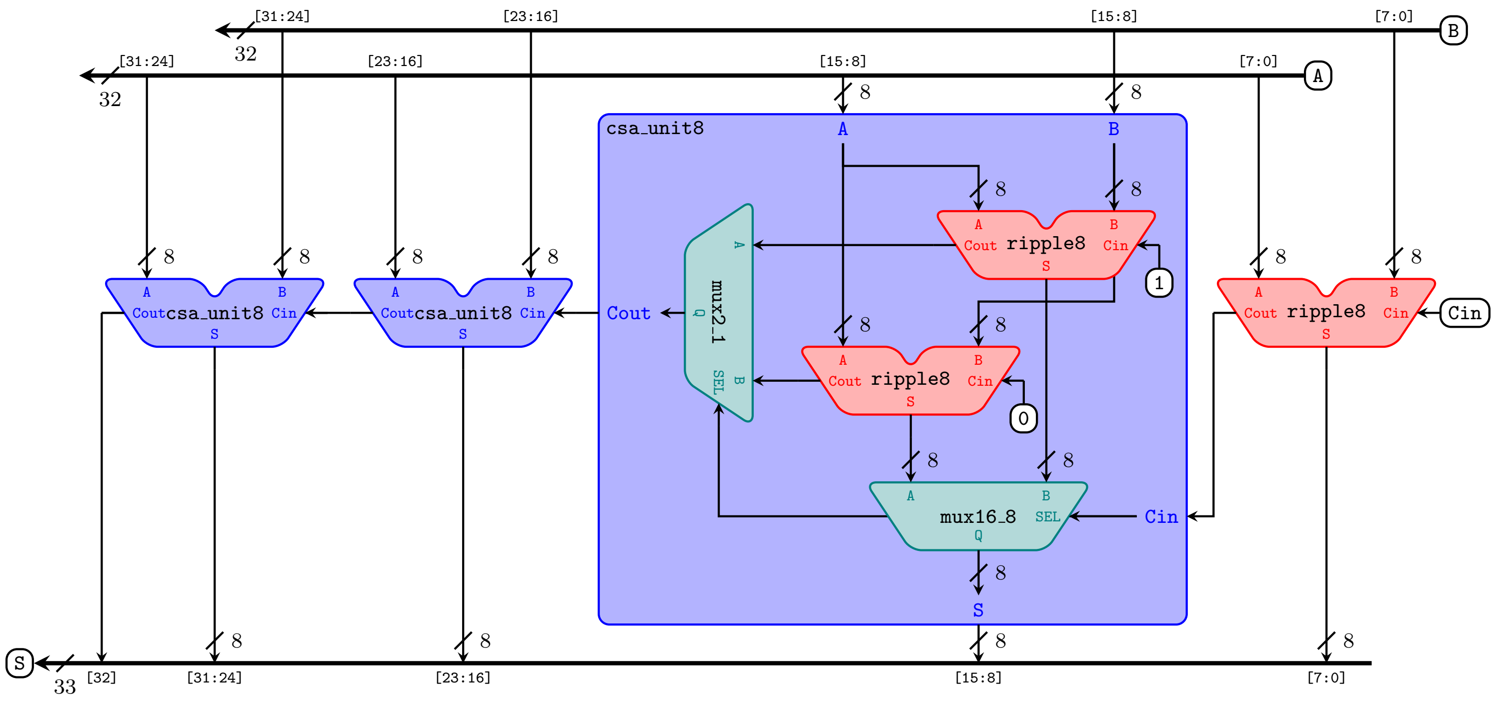


FIGURE 1: Diagram of a 32-bit carry select adder.

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

The propagation delay of the carry select adder 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 carry select adder compared to the carry ripple adder include the increased complexity, size and power requirements due to duplicating most of the full adders and including multiplexors.

```vhdl
ARCHITECTURE full_adder_architecture OF full_adder IS
BEGIN
    Cout <= ((A xor B) and Cin) or (A and B);
    S <= (A xor B) xor Cin;
END;
```

LISTING 1: `full_adder_architecture` declaration in `full_adder.vhd`

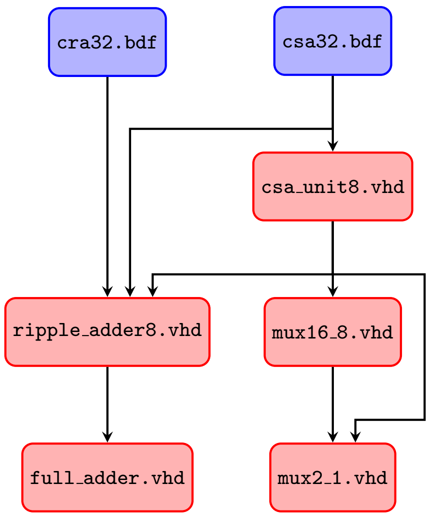


FIGURE 2: Hierarchy of entity declarations.

```vhdl
ARCHITECTURE mux2_1_architecture OF mux2_1 IS
BEGIN
    Q <= (A and not SEL) or (B and SEL);
END mux2_1_architecture;
```

LISTING 2: `mux2_1_architecture` declaration in `mux2_1.vhd`

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

```vhdl
BEGIN
    fa0 : full_adder PORT MAP ( A => A(0),
                                B => B(0),
                                Cin => Cin,
                                Cout => C(0),
                                S => S(0));
    fa1 : full_adder PORT MAP ( A => A(1),
                                B => B(1),
                                Cin => C(0),
                                Cout => C(1),
                                S => S(1));
```

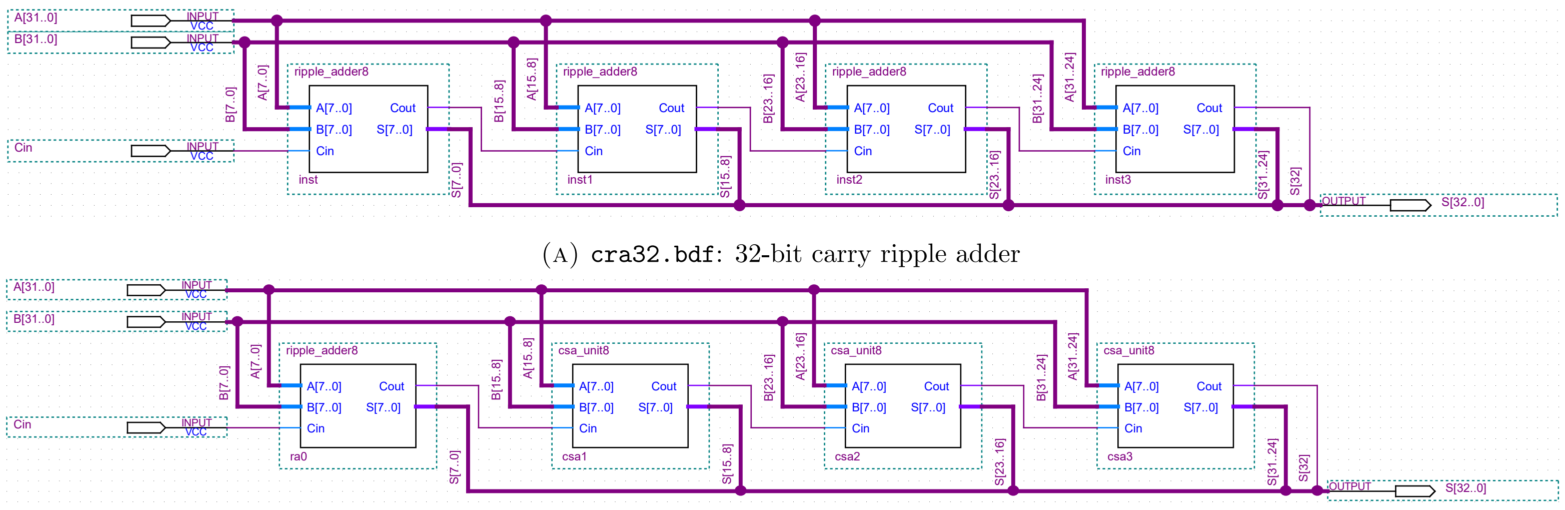
```vhdl
    fa7 : full_adder PORT MAP ( A => A(7),
                                B => B(7),
                                Cin => C(6),
                                Cout => Cout,
                                S => S(7));
END;
```

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

```vhdl
    SIGNAL S0 : std_logic_vector(7 downto 0);
    SIGNAL Cout0 : std_logic;
    SIGNAL S1 : std_logic_vector(7 downto 0);
    SIGNAL Cout1 : std_logic;
```

```vhdl
BEGIN
    ra0 : ripple_adder8 PORT MAP (  A => A,
                                    B => B,
                                    Cin => '0',
                                    Cout => Cout0,
                                    S => S0);
    ra1 : ripple_adder8 PORT MAP (  A => A,
                                    B => B,
                                    Cin => '1',
                                    Cout => Cout1,
                                    S => S1);
    smux : mux16_8 PORT MAP (A => S0,
                            B => S1,
                            SEL => Cin,
                            Q => S);
    cmux : mux2_1 PORT MAP ( A => Cout0,
                            B => Cout1,
                            SEL => Cin,
                            Q => Cout);
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.



(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

3 FPGA Implementation