Lecture 13

ECE 0201: Digital Circuits and Systems

Binary Adders (4.5)

Announcements

- Assignments due Saturday Oct. 10
  - Homework 6
  - Quiz 6
  - Lab 6 report
- Start Lab 7 Friday Oct. 9
  - In-Person Lab appointments on the Canvas calendar

Questions for Today

How do we design digital systems?

How can we perform arithmetic with combinational logic?

Binary Half Adder

A combinational circuit that adds two 1-bit numbers is called a binary **half-adder.**
For Z = X + Y,

Inputs: X, Y$\in \{\mathbf{0}, \mathbf{1}\}$

Output: Z $\in \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$

Binary Half Adder

A combinational circuit that adds two 1-bit numbers is called a binary **half-adder.**
For Z = X + Y,

Inputs: X, Y$\in \{\mathbf{0}, \mathbf{1}\}$

Output: Z $\in \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 2

Binary Half Adder

A combinational circuit that adds two 1-bit numbers is called a binary **half-adder.** For Z = X + Y,

Inputs: X, Y$\in \{\mathbf{0}, \mathbf{1}\}$

Output: Z $\in \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$

The **LSB** of the result is referred to as the **sum (s)**, the **MSB** of the result is known as the **carry (c)**.

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 2

x	y	c	s
0	0		
0	1		
1	0		
1	1		

Binary Half Adder

A combinational circuit that adds two 1-bit numbers is called a binary **half-adder.** For Z = X + Y,

Inputs: X, Y$\in \{\mathbf{0}, \mathbf{1}\}$

Output: Z $\in \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$

The **LSB** of the result is referred to as the **sum (s)**, the **MSB** of the result is known as the **carry (c)**.

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 2

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

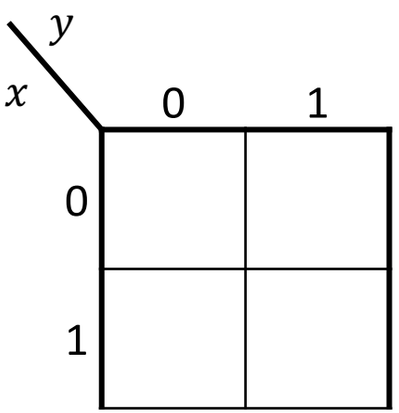
Binary Half Adder

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

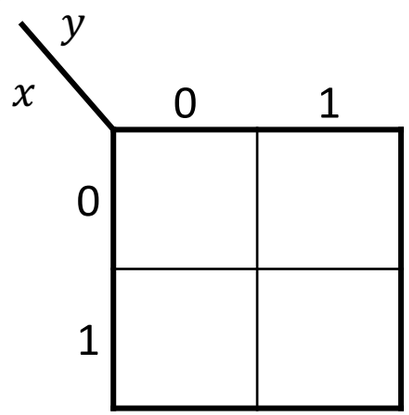
Binary Half Adder

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

c



s



Binary Half Adder

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$\boldsymbol{c}$

x \ y	0	1
0	0	0
1	0	1

$$\boldsymbol{c = xy}$$

$\boldsymbol{s}$

x \ y	0	1
0	0	1
1	1	0

$$\boldsymbol{s = x\bar{y} + \bar{x}y}$$

Binary Half Adder

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$\boldsymbol{c}$

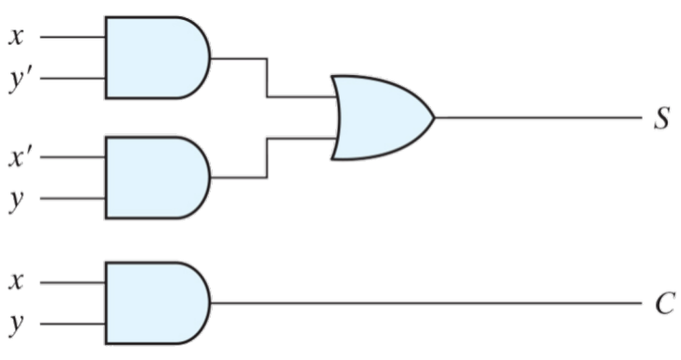
x \ y	0	1
0	0	0
1	0	1

$\boldsymbol{c = xy}$

$\boldsymbol{s}$

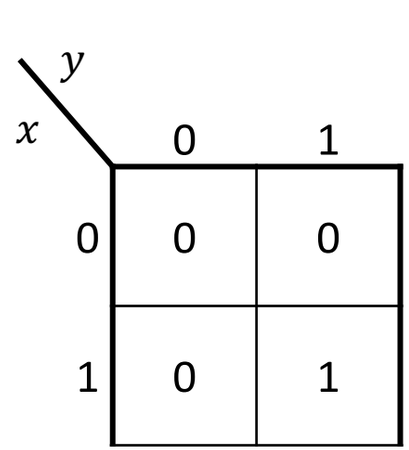
x \ y	0	1
0	0	1
1	1	0

$\boldsymbol{s = x\bar{y} + \bar{x}y}$

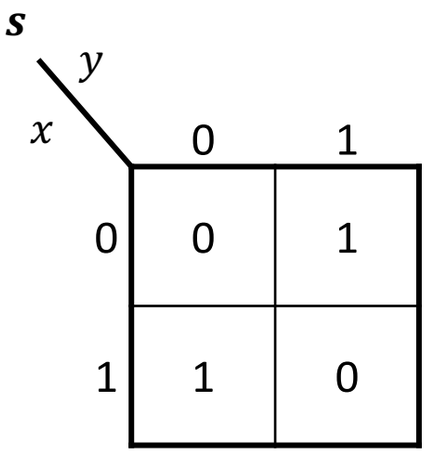


Binary Half Adder

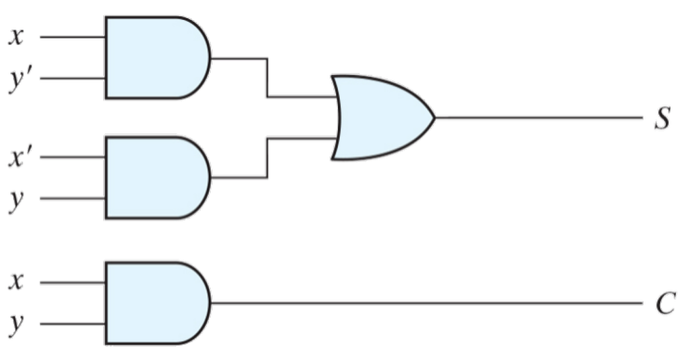
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

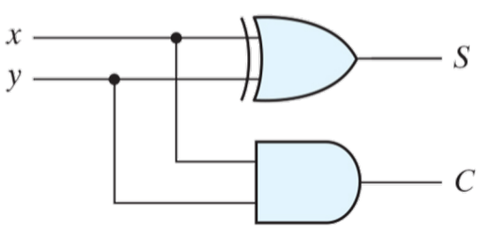


$$\boldsymbol{c = xy}$$



$$s = x\bar{y} + \bar{x}y$$
$$\boldsymbol{s = x \oplus y}$$





Binary Half Adder

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

c

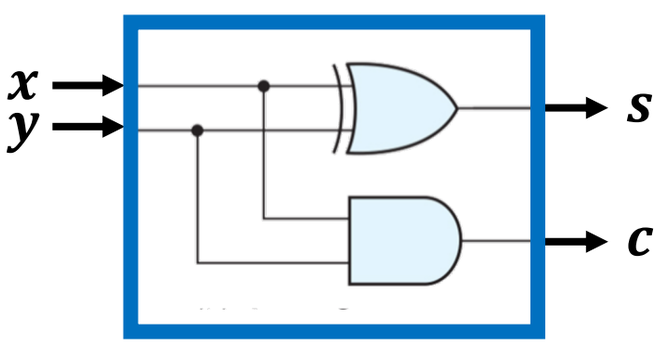
x \ y	0	1
0	0	0
1	0	1

$$c = xy$$

s

x \ y	0	1
0	0	1
1	1	0

$$s = x\bar{y} + \bar{x}y$$
$$s = x \oplus y$$



Binary n-bit Addition

To add numbers of more than 1 bit, we carry out pairwise addition of the bits in each bit position.

If the result of the addition is greater than 1, a **carry-out** bit is generated and is included in the addition of the next bit position **(carry-in)**.

Binary n-bit Addition

To add numbers of more than 1 bit, we carry out pairwise addition of the bits in each bit position.

If the result of the addition is greater than 1, a **carry-out** bit is generated and is included in the addition of the next bit position **(carry-in)**.

1001 1101

+ 0011 1111

Binary n-bit Addition

To add numbers of more than 1 bit, we carry out pairwise addition of the bits in each bit position.

If the result of the addition is greater than 1, a **carry-out** bit is generated and is included in the addition of the next bit position **(carry-in)**.

$$\begin{array}{rr} & \mathbf{1001\ 1101} \\ \mathbf{+} & \mathbf{0011\ 1111} \\ \hline & 1101\ 1100 \end{array}$$

Binary n-bit Addition

To add numbers of more than 1 bit, we carry out pairwise addition of the bits in each bit position.

If the result of the addition is greater than 1, a **carry-out** bit is generated and is included in the addition of the next bit position **(carry-in)**.

Inputs: X, Y, $C_{IN} \in \{0, 1\}$
Output: Z $\in \{0, 1, 2, 3\}$

1001 1101
+ 0011 1111
1101 1100

Binary Full Adder

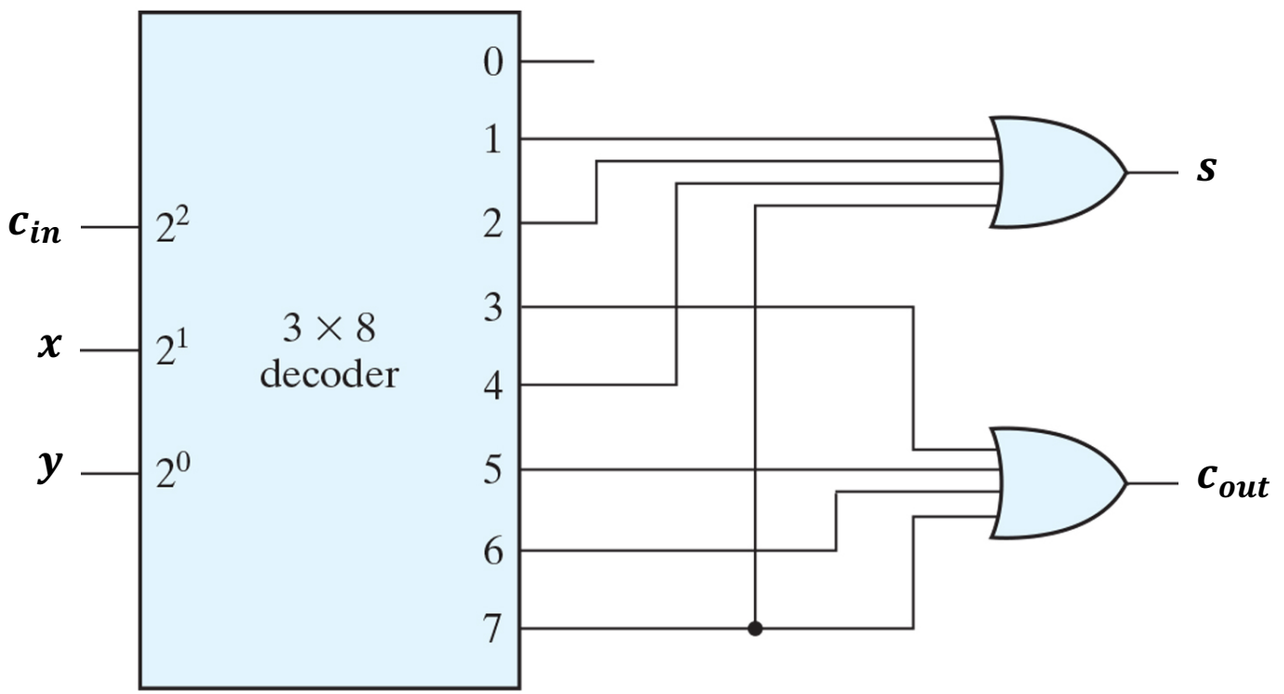
c_{in}	x	y	c_{out}	s
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Binary Full Adder

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Binary Full Adder

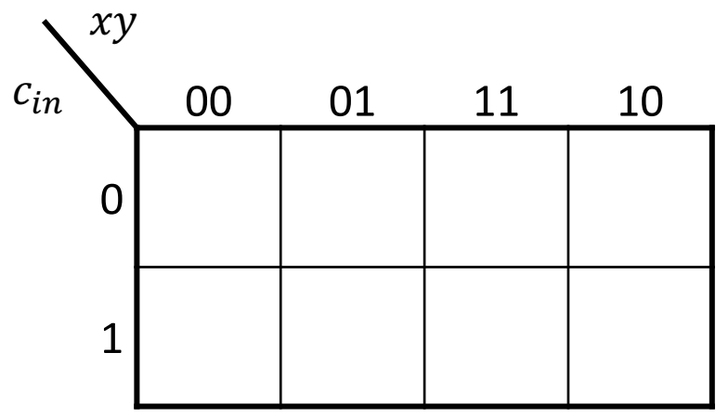
c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



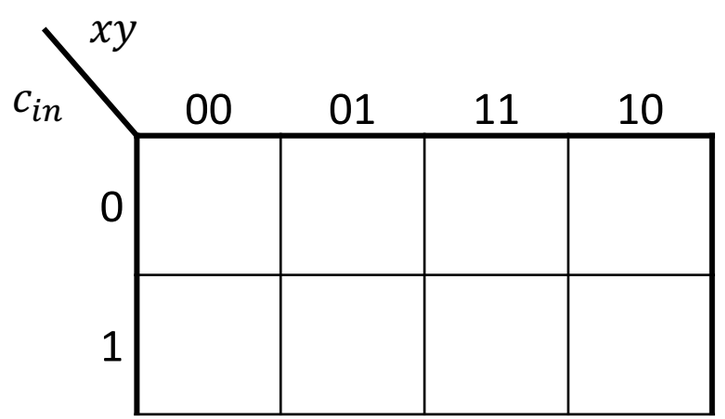
Binary Full Adder

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

c_{out}



s



Binary Full Adder

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

c_{out}

c_{in} \ xy	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$c_{out} = xy + c_{in}x + c_{in}y$$

s

c_{in} \ xy	00	01	11	10
0	0	1	0	1
1	1	0	1	0

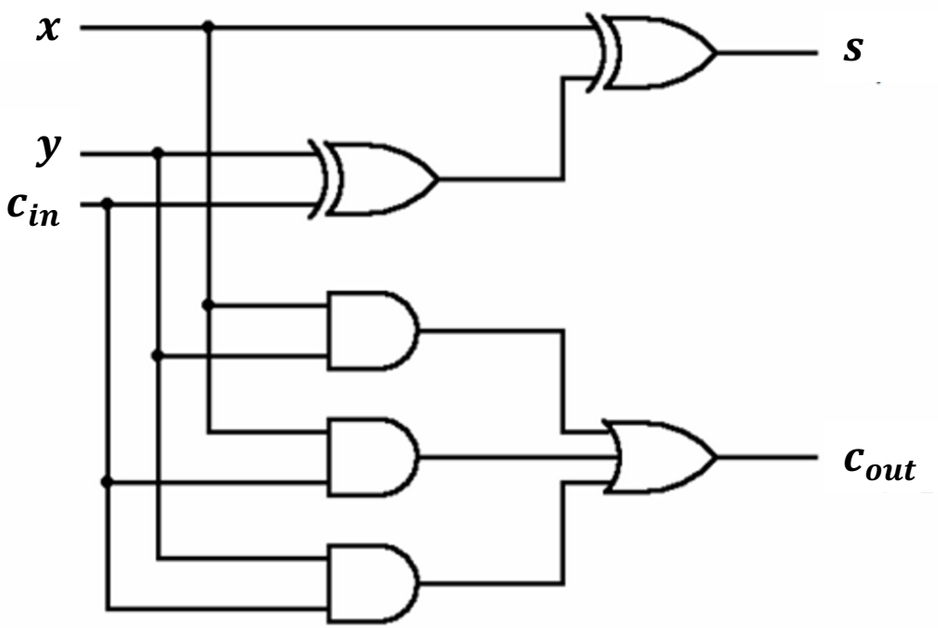
$$s = c_{in} \oplus x \oplus y$$

Binary Full Adder

$$c_{out} = xy + c_{in}x + c_{in}y$$

$$s = c_{in} \oplus x \oplus y$$

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

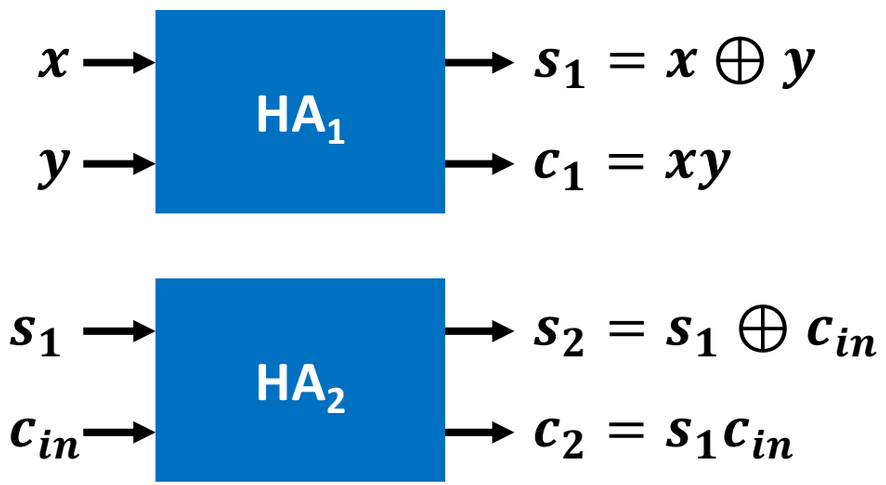


Binary Full Adder

$$c_{out} = xy + c_{in}x + c_{in}y$$

$$s = c_{in} \oplus x \oplus y$$

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

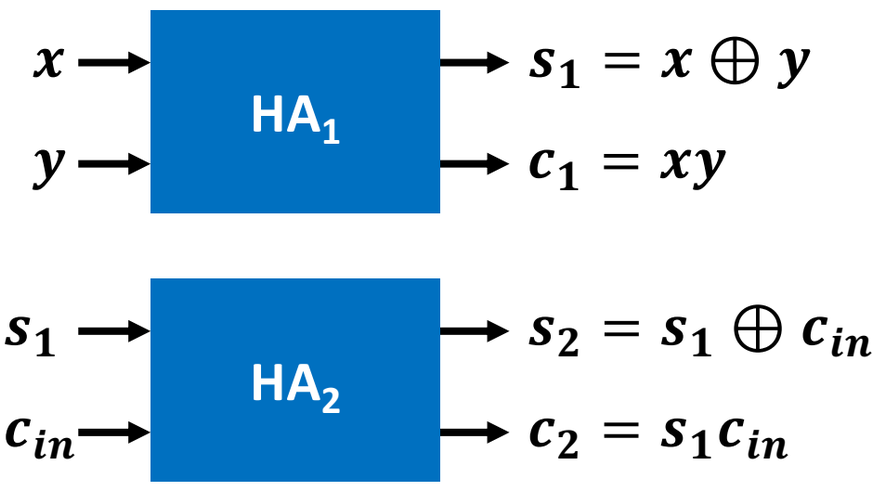


Binary Full Adder

$$c_{out} = xy + c_{in}x + c_{in}y$$

$$s = c_{in} \oplus x \oplus y$$

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$s_2 = x \oplus y \oplus c_{in} = s$$

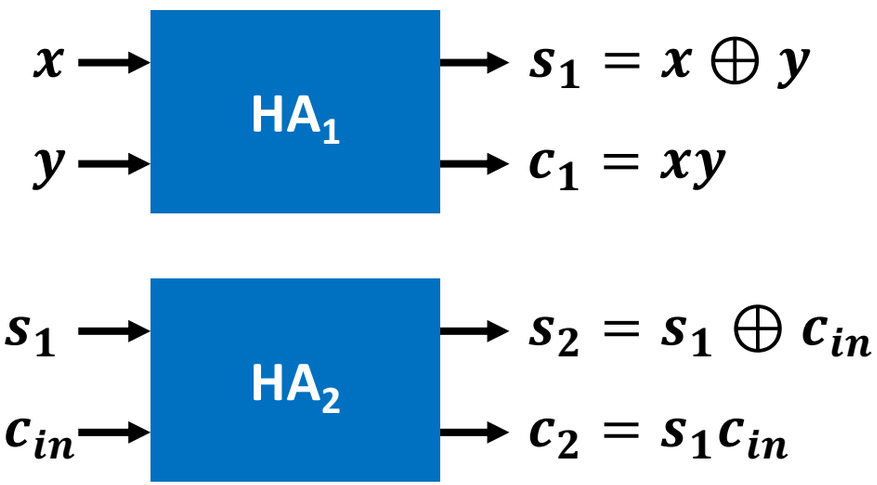
$$c_2 = (x \oplus y)c_{in}$$

Binary Full Adder

$$c_{out} = xy + c_{in}x + c_{in}y$$

$$s = c_{in} \oplus x \oplus y$$

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



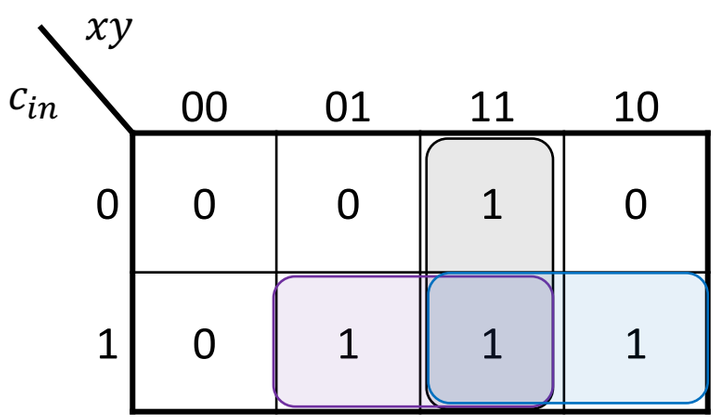
$$s_2 = x \oplus y \oplus c_{in} = s$$

$$c_2 = (x \oplus y)c_{in}$$

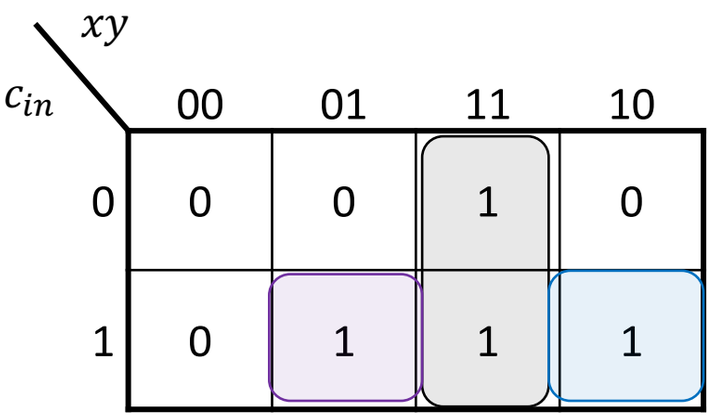
$$c_{out} = c_1 + c_2$$

Binary Full Adder

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$c_{out} = xy + c_{in}x + c_{in}y$$
$$= xy + c_{in}(x + y)$$



$$c_{out} = xy + c_{in}\bar{x}y + c_{in}x\bar{y}$$
$$= xy + c_{in}(x \oplus y)$$

$$c_1 = xy$$
$$c_2 = (x \oplus y)c_{in}$$

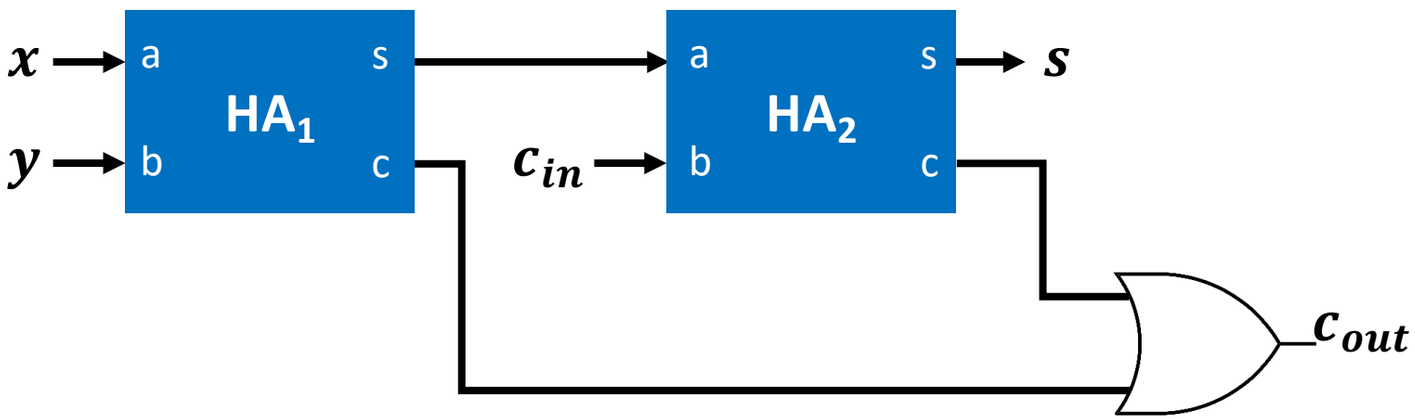
$$c_{out} = c_1 + c_2$$

Binary Full Adder

$$c_{out} = xy + c_{in}(x \oplus y) = c_1 + c_2$$

$$s = c_{in} \oplus s_1$$

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

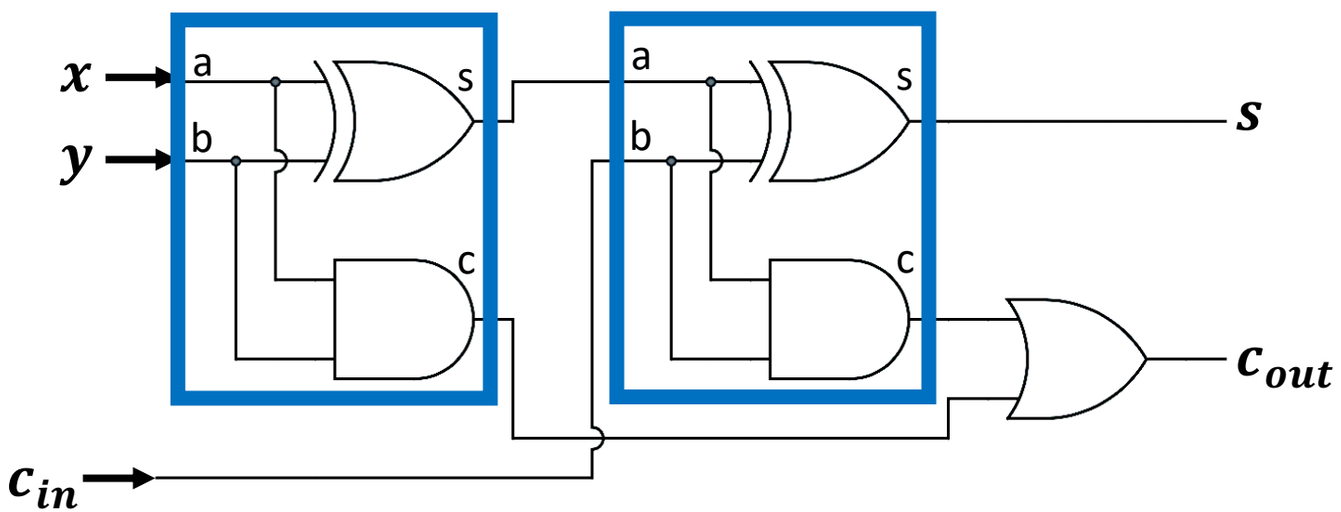


Binary Full Adder

$$c_{out} = xy + c_{in}(x \oplus y) = c_1 + c_2$$

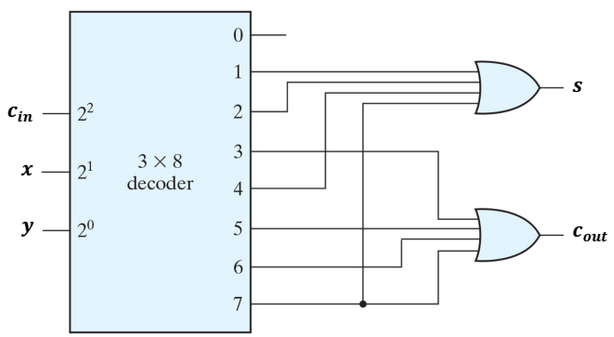
$$s = c_{in} \oplus s_1$$

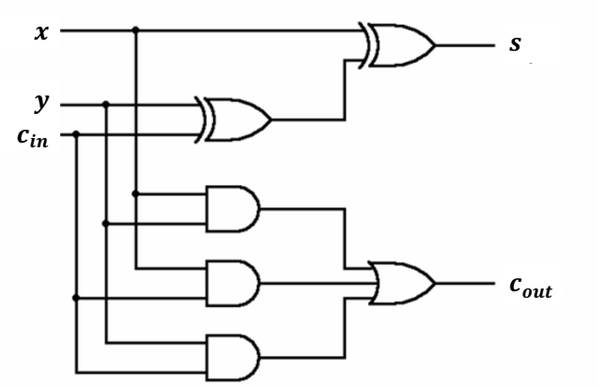
c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

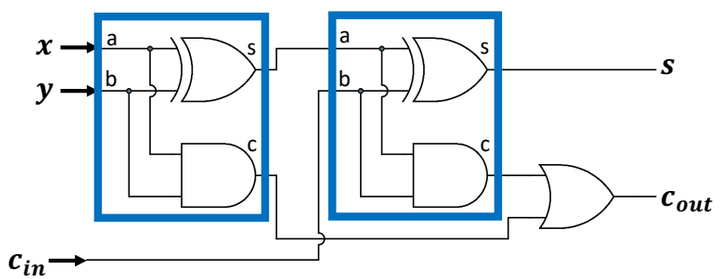


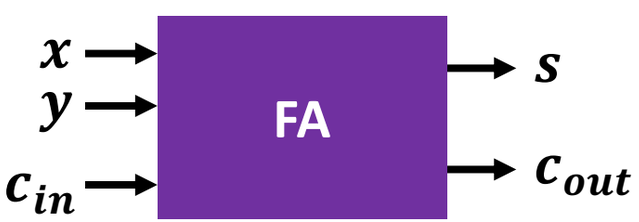
Binary Full Adder

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1







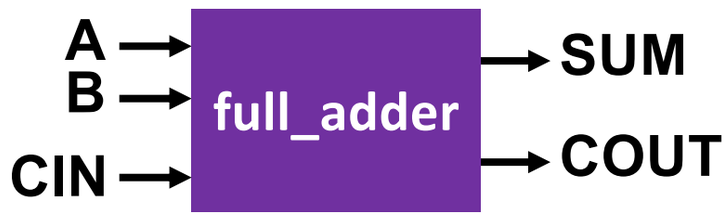


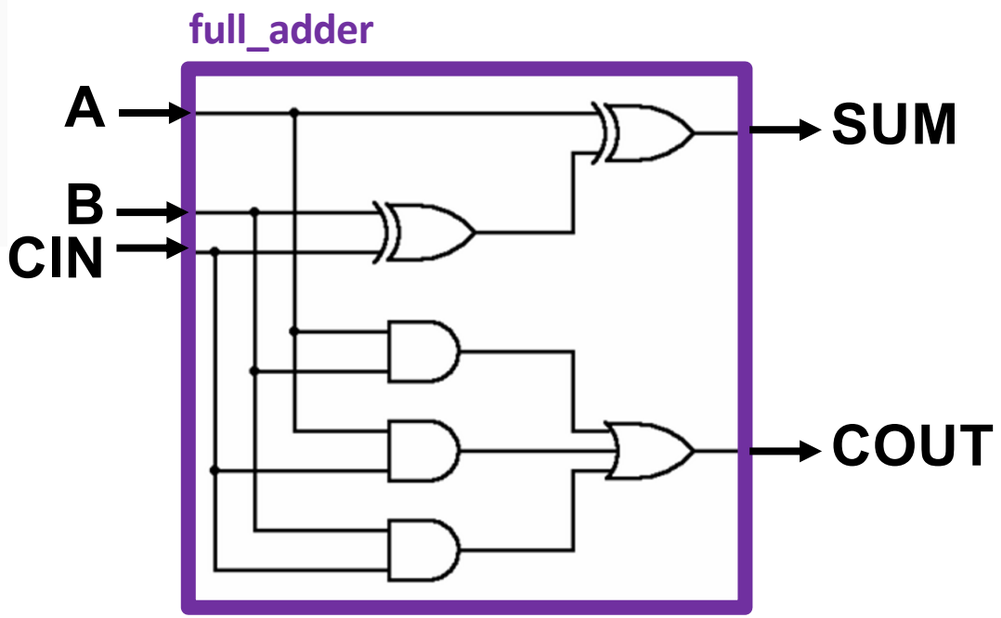
Full Adder VHDL

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder;
8
9
10  architecture full_adder of full_adder is
11  begin
12     SUM <= A xor B xor CIN;
13     COUT <= (A and B) or (B and CIN) or (CIN and A);
14  end full_adder;
15
```

Full Adder VHDL

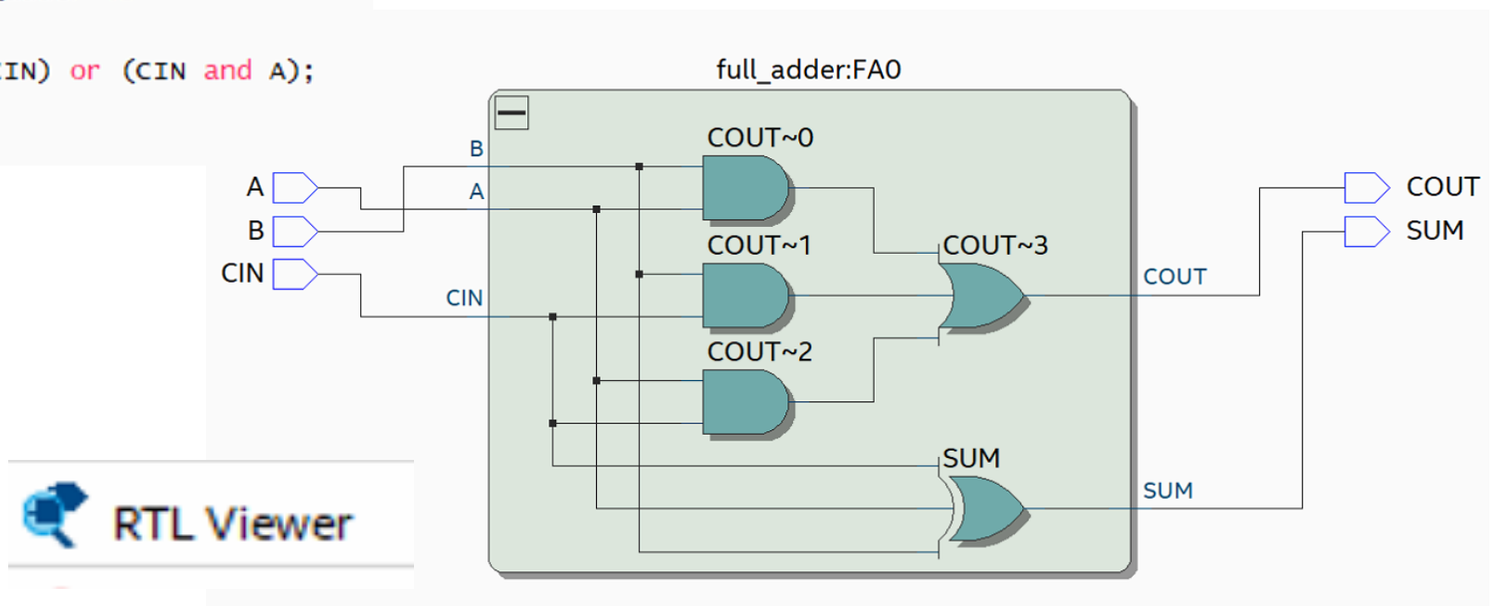
```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder;
8
9
10  architecture full_adder of full_adder is
11  begin
12     SUM <= A xor B xor CIN;
13     COUT <= (A and B) or (B and CIN) or (CIN and A);
14  end full_adder;
15
```





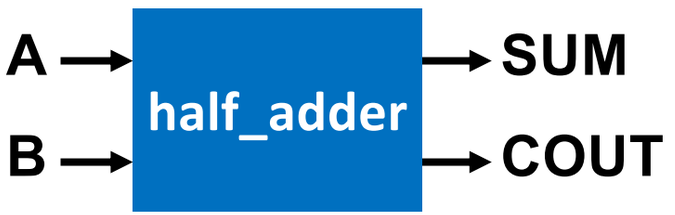
Full Adder VHDL

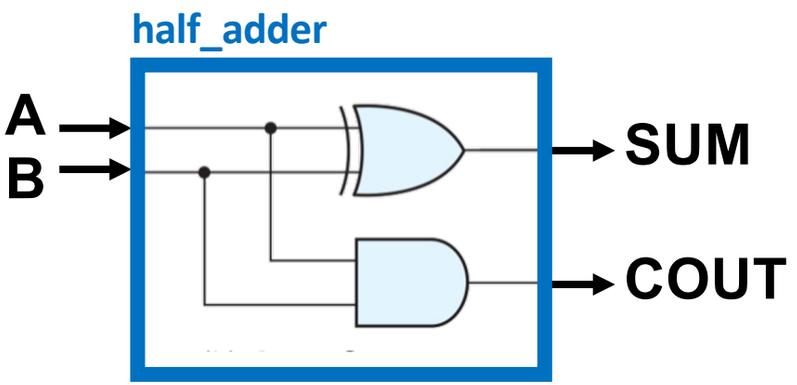
```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder;
8
9
10  architecture full_adder of full_adder is
11  begin
12     SUM <= A xor B xor CIN;
13     COUT <= (A and B) or (B and CIN) or (CIN and A);
14  end full_adder;
15
```



Half Adder VHDL

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4
5   entity half_adder is
6      port (A, B: in std_logic;
7            SUM, COUT: out std_logic);
8   end half_adder;
9
10
11  architecture ha_xor of half_adder is
12  begin
13      SUM <= A xor B;
14      COUT <= A and B;
15  end ha_xor;
16
```



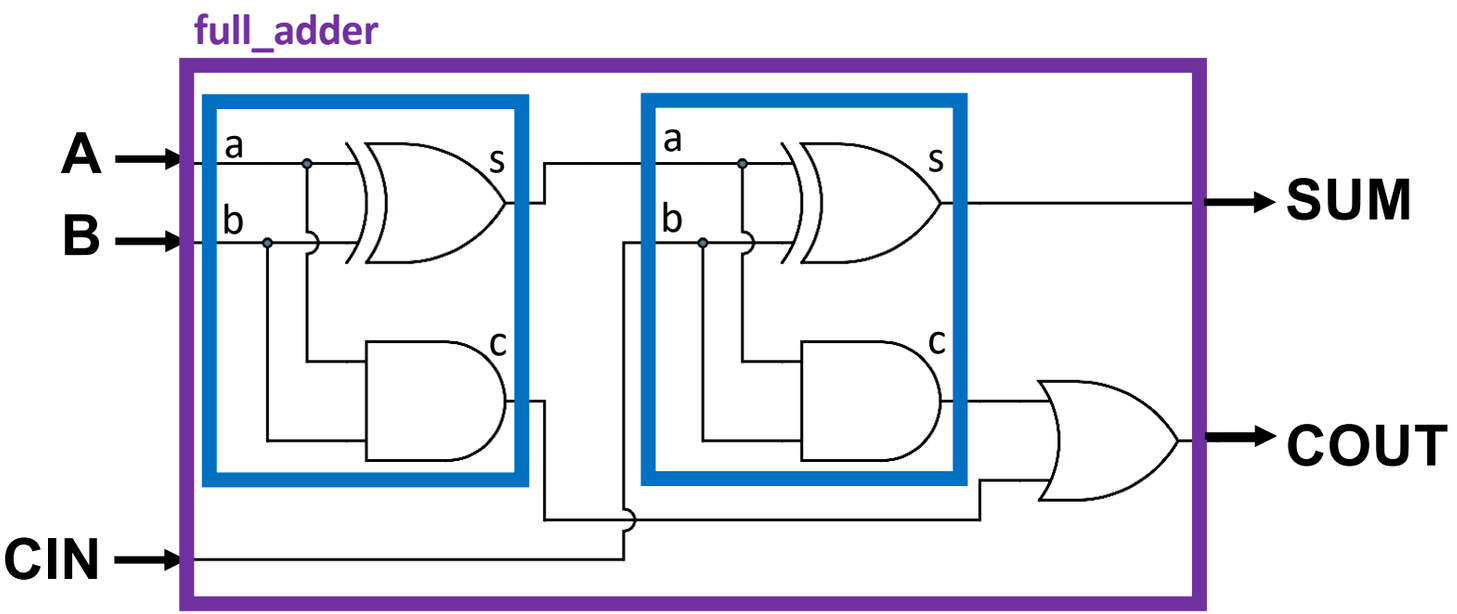


Full Adder VHDL: Structural

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder_struct is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder_struct;
8
9   architecture structure of full_adder_struct is
10     component half_adder is
11        port (A, B: in std_logic;
12              COUT, SUM: out std_logic);
13     end component;
14     signal SUM1, COUT1, COUT2: std_logic;
15  begin
16     HA0: half_adder port map ( A=>A,
17                                B=>B,
18                                COUT=>COUT1,
19                                SUM=>SUM1 );
20     HA1: half_adder port map ( A=>SUM1,
21                                B=>CIN,
22                                COUT=>COUT2,
23                                SUM=>SUM );
24     COUT <= COUT1 or COUT2;
25  end structure;
26
```

Full Adder VHDL: Structural

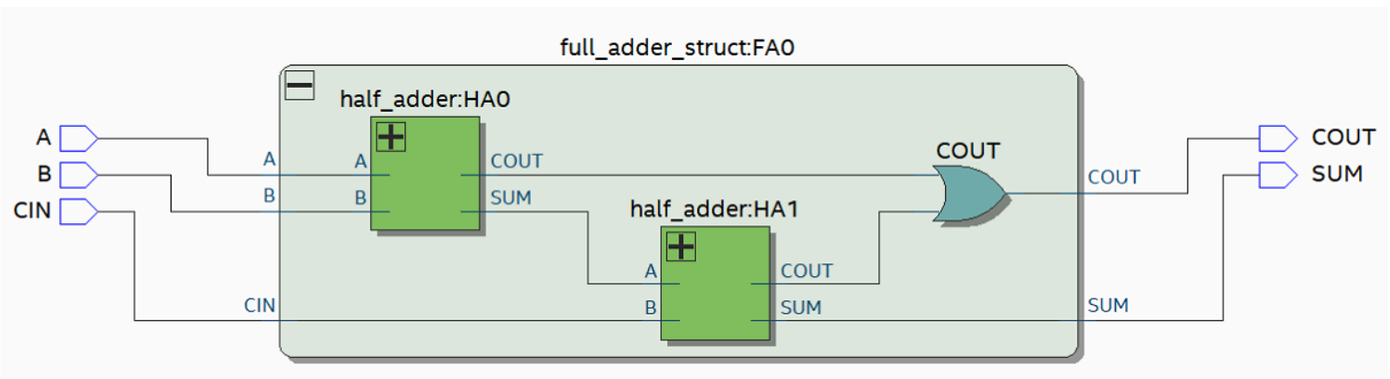
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity full_adder_struct is
    port (A, B, CIN: in std_logic;
          SUM, COUT: out std_logic);
end full_adder_struct;

architecture structure of full_adder_struct is
    component half_adder is
        port (A, B: in std_logic;
              COUT, SUM: out std_logic);
    end component;
    signal SUM1, COUT1, COUT2: std_logic;
begin
    HA0: half_adder port map ( A=>A,
                               B=>B,
                               COUT=>COUT1,
                               SUM=>SUM1 );
    HA1: half_adder port map ( A=>SUM1,
                               B=>CIN,
                               COUT=>COUT2,
                               SUM=>SUM );
    COUT <= COUT1 or COUT2;
end structure;
```

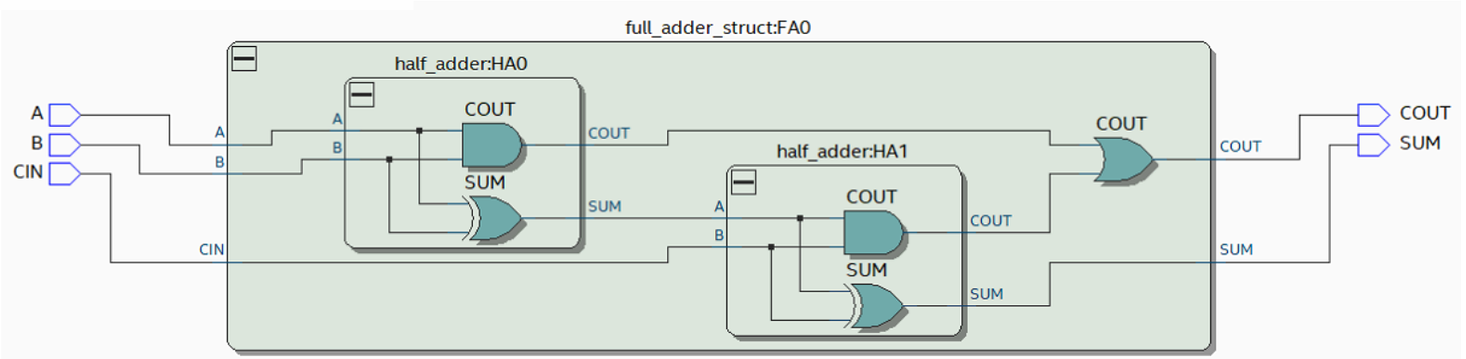


Full Adder VHDL: Structural

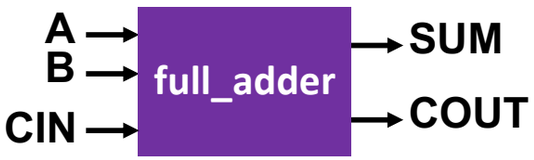
RTL Viewer

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder_struct is
5       port (A, B, CIN: in std_logic;
6             SUM, COUT: out std_logic);
7   end full_adder_struct;
8
9   architecture structure of full_adder_struct is
10      component half_adder is
11          port (A, B: in std_logic;
12                COUT, SUM: out std_logic);
13      end component;
14      signal SUM1, COUT1, COUT2: std_logic;
15  begin
16      HA0: half_adder port map ( A=>A,
17                                 B=>B,
18                                 COUT=>COUT1,
19                                 SUM=>SUM1 );
20      HA1: half_adder port map ( A=>SUM1,
21                                 B=>CIN,
22                                 COUT=>COUT2,
23                                 SUM=>SUM );
24      COUT <= COUT1 or COUT2;
25  end structure;
26
```



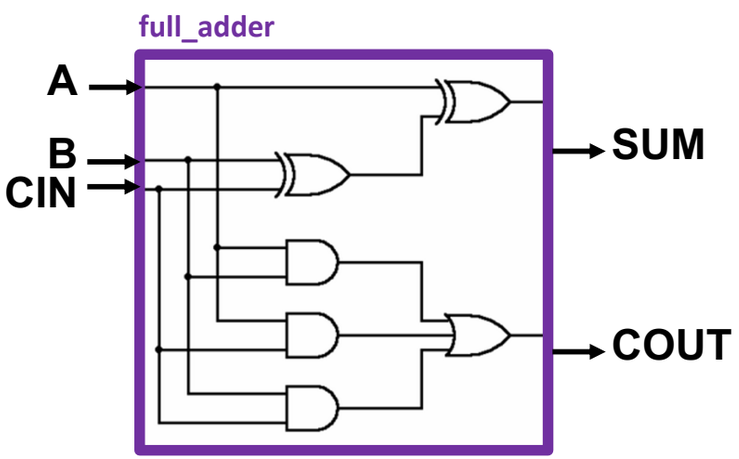


Full Adder VHDL



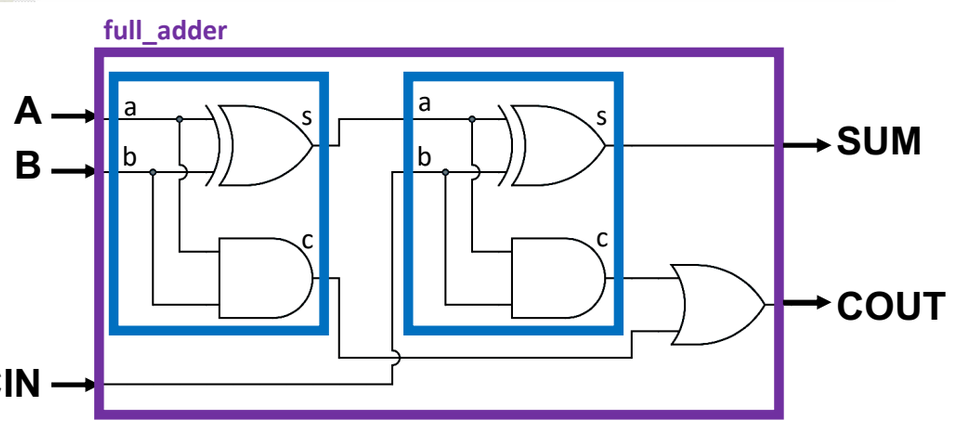
Data-Flow Model

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder;
8
9
10  architecture full_adder of full_adder is
11  begin
12     SUM <= A xor B xor CIN;
13     COUT <= (A and B) or (B and CIN) or (CIN and A);
14  end full_adder;
15
```

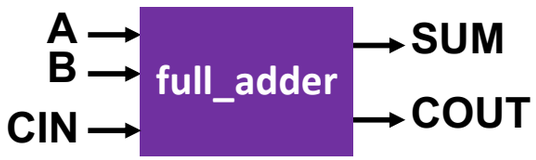


Structural Model

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder_struct is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder_struct;
8
9   architecture structure of full_adder_struct is
10     component half_adder is
11        port (A, B: in std_logic;
12              COUT, SUM: out std_logic);
13     end component;
14     signal SUM1, COUT1, COUT2: std_logic;
15  begin
16     HA0: half_adder port map (A=>A, B=>B, COUT=>COUT1, SUM=>SUM1);
17     HA1: half_adder port map (A=>SUM1, B=>CIN, COUT=>COUT2, SUM=>SUM);
18     COUT <= COUT1 or COUT2;
19  end structure;
20
```



Full Adder VHDL

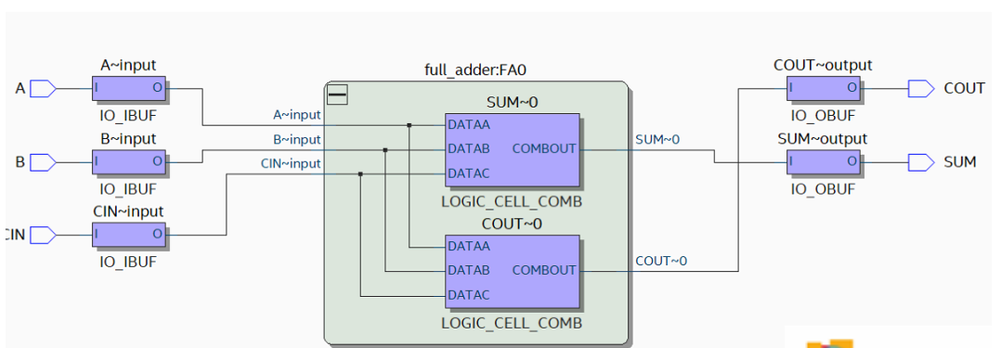


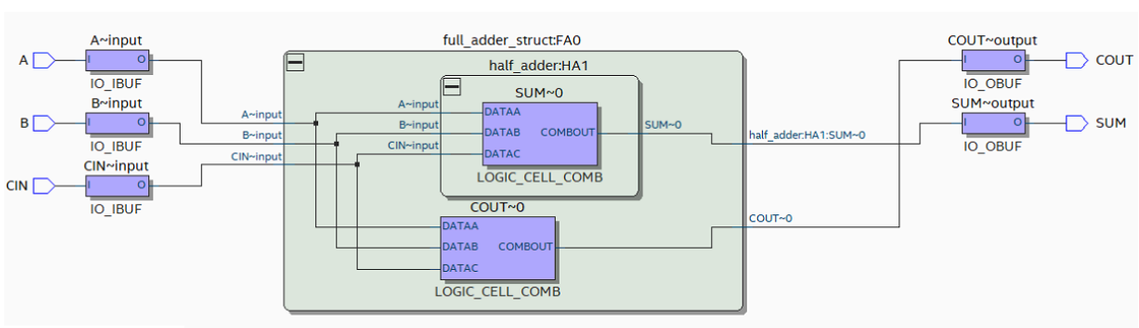
Data-Flow Model

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder is
5       port (A, B, CIN: in std_logic;
6             SUM, COUT: out std_logic);
7   end full_adder;
8
9
10  architecture full_adder of full_adder is
11  begin
12      SUM <= A xor B xor CIN;
13      COUT <= (A and B) or (B and CIN) or (CIN and A);
14  end full_adder;
15
```

Structural Model

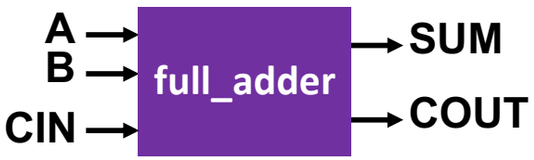
```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder_struct is
5       port (A, B, CIN: in std_logic;
6             SUM, COUT: out std_logic);
7   end full_adder_struct;
8
9   architecture structure of full_adder_struct is
10      component half_adder is
11          port (A, B: in std_logic;
12                COUT, SUM: out std_logic);
13      end component;
14      signal SUM1, COUT1, COUT2: std_logic;
15  begin
16      HA0: half_adder port map (A=>A, B=>B, COUT=>COUT1, SUM=>SUM1);
17      HA1: half_adder port map (A=>SUM1, B=>CIN, COUT=>COUT2, SUM=>SUM);
18      COUT <= COUT1 or COUT2;
19  end structure;
20
```





Technology Map Viewer

Full Adder VHDL

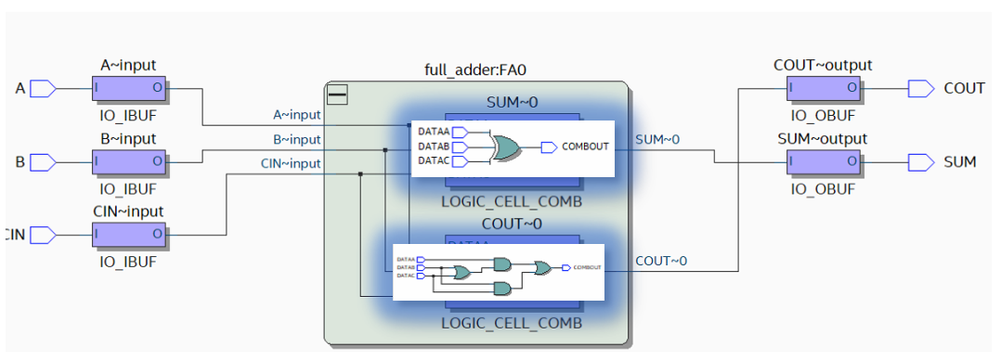


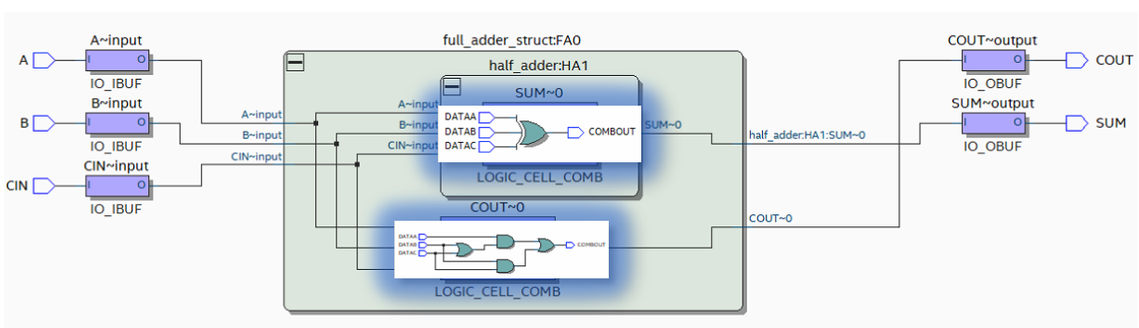
Data-Flow Model

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder;
8
9
10  architecture full_adder of full_adder is
11  begin
12     SUM <= A xor B xor CIN;
13     COUT <= (A and B) or (B and CIN) or (CIN and A);
14  end full_adder;
15
```

Structural Model

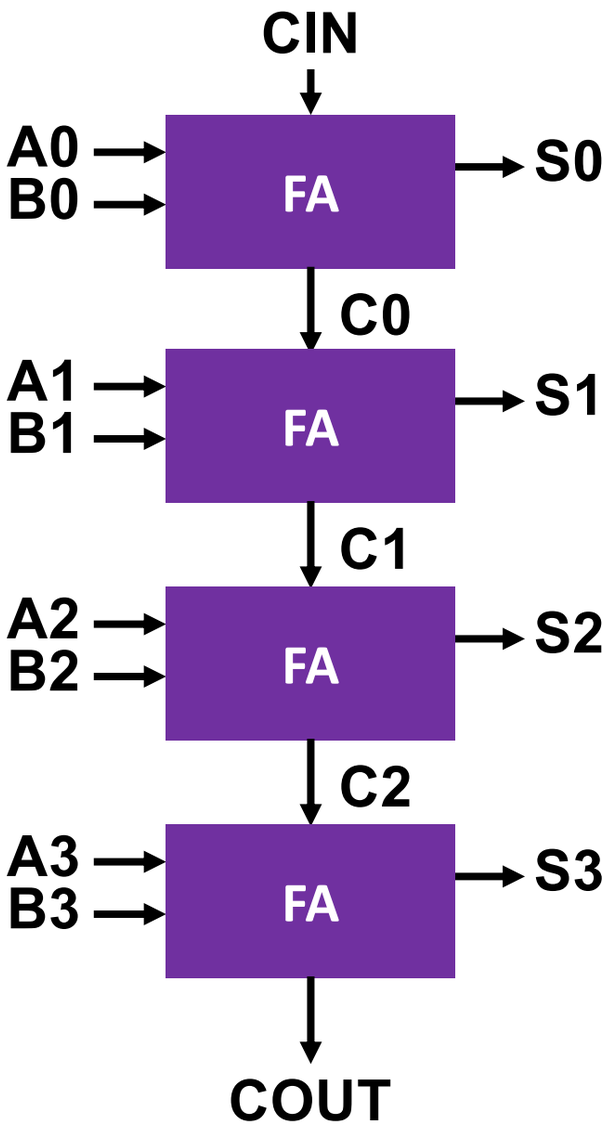
```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity full_adder_struct is
5      port (A, B, CIN: in std_logic;
6            SUM, COUT: out std_logic);
7   end full_adder_struct;
8
9   architecture structure of full_adder_struct is
10     component half_adder is
11        port (A, B: in std_logic;
12              COUT, SUM: out std_logic);
13     end component;
14     signal SUM1, COUT1, COUT2: std_logic;
15  begin
16     HA0: half_adder port map (A=>A, B=>B, COUT=>COUT1, SUM=>SUM1);
17     HA1: half_adder port map (A=>SUM1, B=>CIN, COUT=>COUT2, SUM=>SUM);
18     COUT <= COUT1 or COUT2;
19  end structure;
20
```





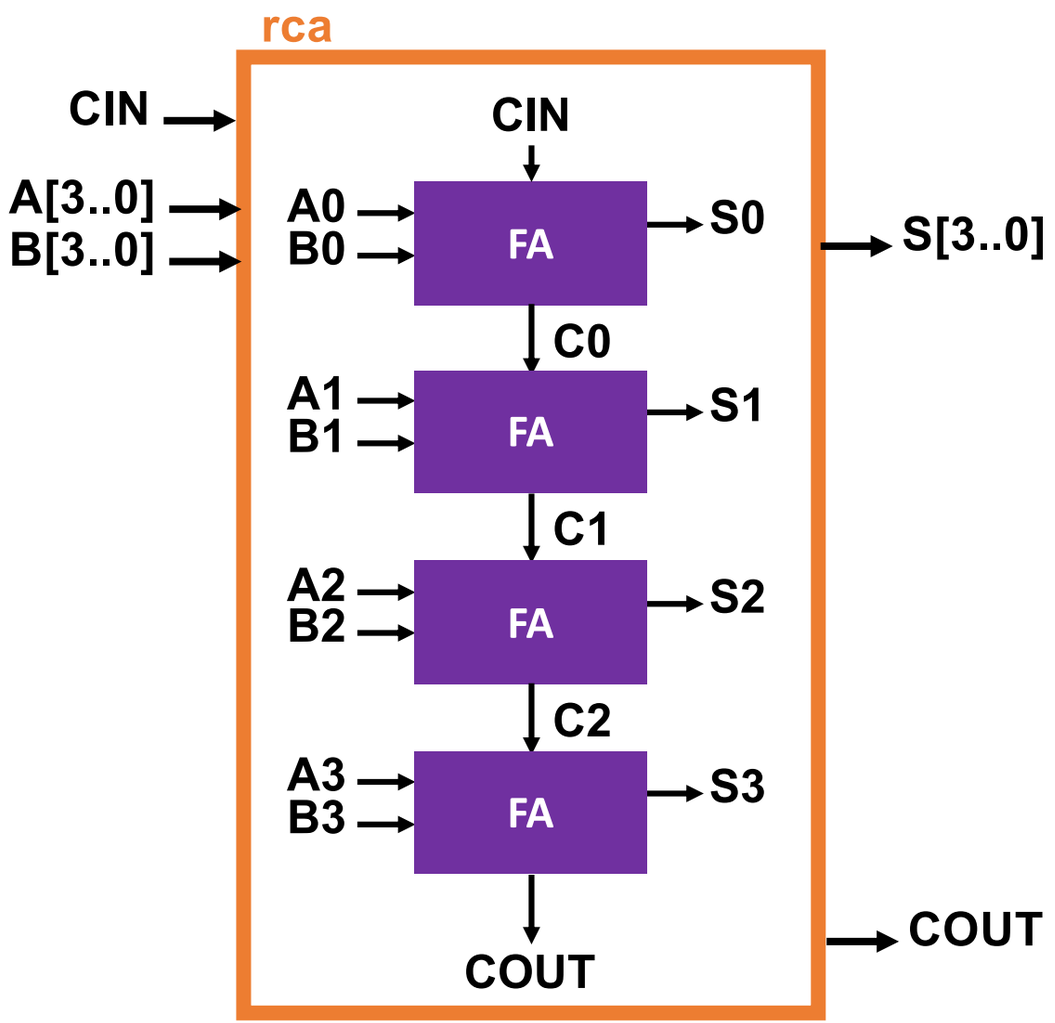
Ripple-Carry Adder

Cascade **n** full adders to add **n** bits, connecting each carry-out to the carry-in for the next stage.



Ripple-Carry Adder VHDL

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity rca is
5      port (A, B : in std_logic_vector(3 downto 0);
6            CIN : in std_logic;
7            S : out std_logic_vector(3 downto 0);
8            COUT : out std_logic);
9   end rca;
10
11  architecture rca_struct of rca is
12     component full_adder
13        port (A, B, CIN: in std_logic;
14              SUM, COUT: out std_logic);
15     end component;
16     signal C : std_logic_vector(2 downto 0);
17
18  begin
19     FA0: full_adder port map(A=>A(0), B=>B(0), CIN=>CIN,
20                              SUM=>S(0), COUT=>C(0));
21     FA1: full_adder port map(A=>A(1), B=>B(1), CIN=>C(0),
22                              SUM=>S(1), COUT=>C(1));
23     FA2: full_adder port map(A=>A(2), B=>B(2), CIN=>C(1),
24                              SUM=>S(2), COUT=>C(2));
25     FA3: full_adder port map(A=>A(3), B=>B(3), CIN=>C(2),
26                              SUM=>S(3), COUT=>COUT);
27  end rca_struct;
```

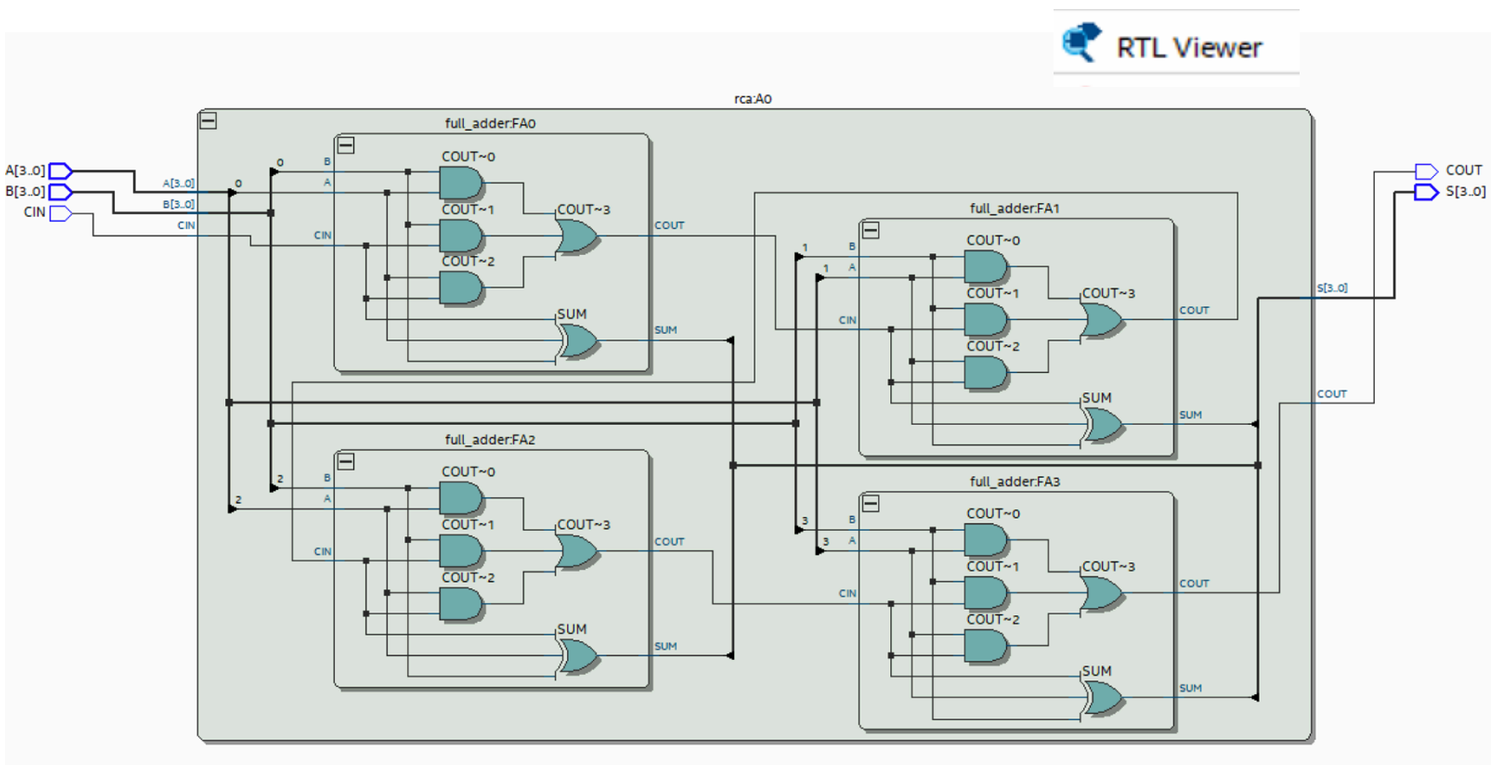


Ripple-Carry Adder VHDL

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity rca is
5      port (A, B : in std_logic_vector(3 downto 0);
6            CIN : in std_logic;
7            S : out std_logic_vector(3 downto 0);
8            COUT : out std_logic);
9   end rca;
10
11  architecture rca_struct of rca is
12     component full_adder
13        port (A, B, CIN: in std_logic;
14              SUM, COUT: out std_logic);
15     end component;
16     signal C : std_logic_vector(2 downto 0);
17
18  begin
19     FA0: full_adder port map(A=>A(0), B=>B(0), CIN=>CIN,
20                              SUM=>S(0), COUT=>C(0));
21     FA1: full_adder port map(A=>A(1), B=>B(1), CIN=>C(0),
22                              SUM=>S(1), COUT=>C(1));
23     FA2: full_adder port map(A=>A(2), B=>B(2), CIN=>C(1),
24                              SUM=>S(2), COUT=>C(2));
25     FA3: full_adder port map(A=>A(3), B=>B(3), CIN=>C(2),
26                              SUM=>S(3), COUT=>COUT);
27  end rca_struct;
```

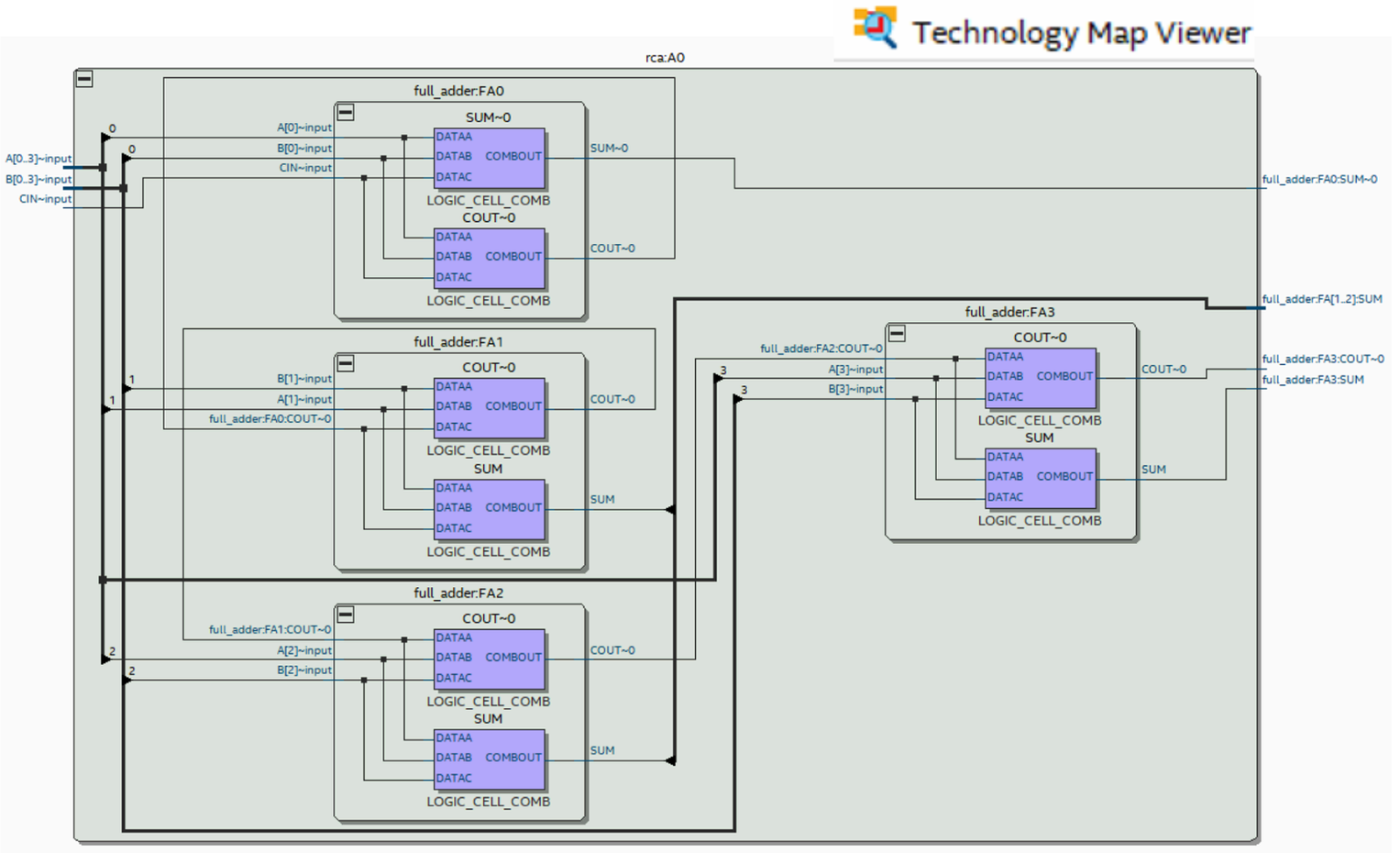
Ripple-Carry Adder VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity rca is
   port (A, B : in std_logic_vector(3 downto 0);
         CIN : in std_logic;
         S : out std_logic_vector(3 downto 0);
         COUT : out std_logic);
end rca;

architecture rca_struct of rca is
   component full_adder
      port (A, B, CIN: in std_logic;
            SUM, COUT: out std_logic);
   end component;
   signal C : std_logic_vector(2 downto 0);

begin
   FA0: full_adder port map(A=>A(0), B=>B(0), CIN=>CIN,
                            SUM=>S(0), COUT=>C(0));
   FA1: full_adder port map(A=>A(1), B=>B(1), CIN=>C(0),
                            SUM=>S(1), COUT=>C(1));
   FA2: full_adder port map(A=>A(2), B=>B(2), CIN=>C(1),
                            SUM=>S(2), COUT=>C(2));
   FA3: full_adder port map(A=>A(3), B=>B(3), CIN=>C(2),
                            SUM=>S(3), COUT=>COUT);
end rca_struct;
```



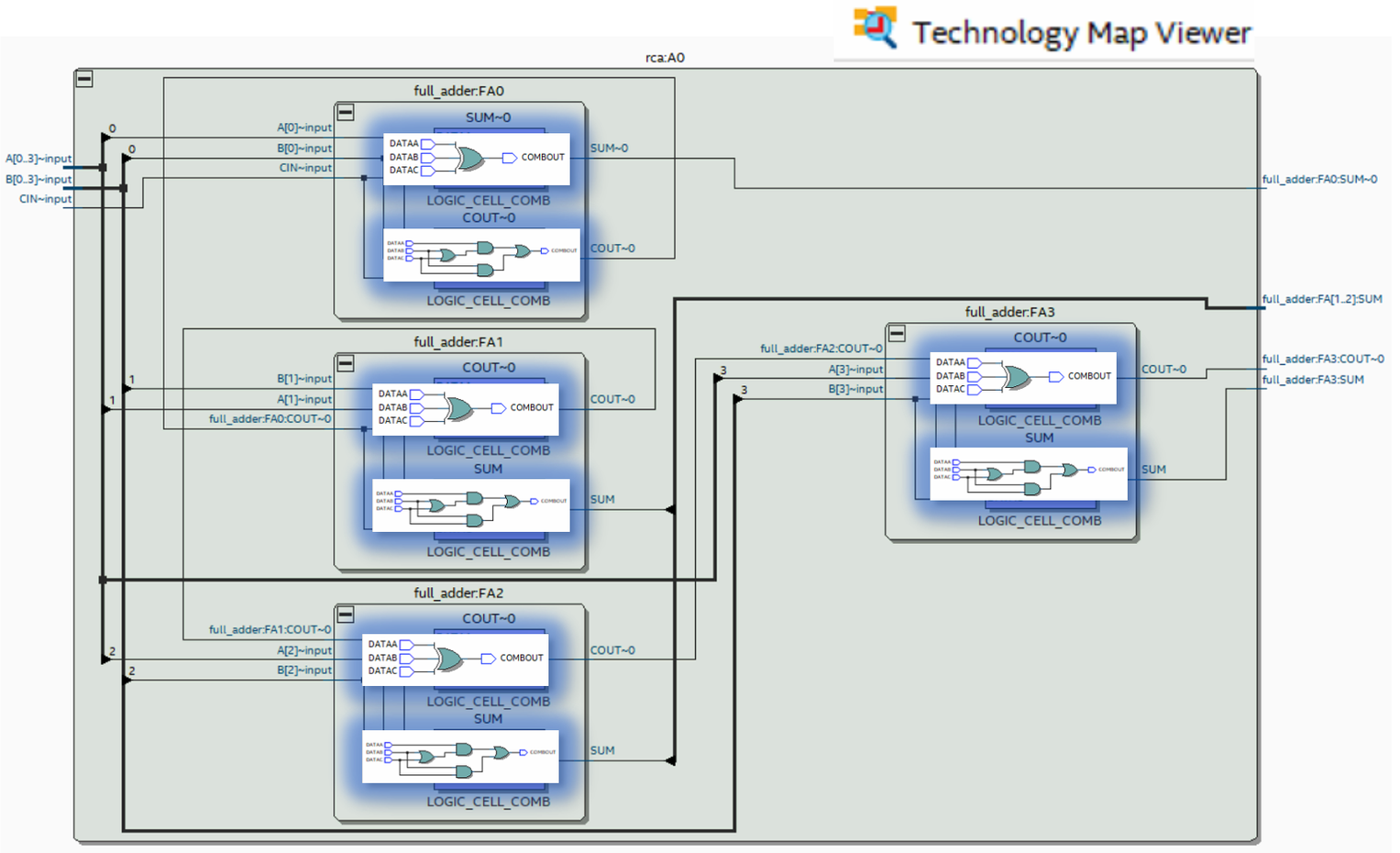
Ripple-Carry Adder VHDL

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity rca is
5      port (A, B : in std_logic_vector(3 downto 0);
6            CIN : in std_logic;
7            S : out std_logic_vector(3 downto 0);
8            COUT : out std_logic);
9   end rca;
10
11  architecture rca_struct of rca is
12     component full_adder
13        port (A, B, CIN: in std_logic;
14              SUM, COUT: out std_logic);
15     end component;
16     signal C : std_logic_vector(2 downto 0);
17
18  begin
19     FA0: full_adder port map(A=>A(0), B=>B(0), CIN=>CIN,
20                              SUM=>S(0), COUT=>C(0));
21     FA1: full_adder port map(A=>A(1), B=>B(1), CIN=>C(0),
22                              SUM=>S(1), COUT=>C(1));
23     FA2: full_adder port map(A=>A(2), B=>B(2), CIN=>C(1),
24                              SUM=>S(2), COUT=>C(2));
25     FA3: full_adder port map(A=>A(3), B=>B(3), CIN=>C(2),
26                              SUM=>S(3), COUT=>COUT);
27  end rca_struct;
```



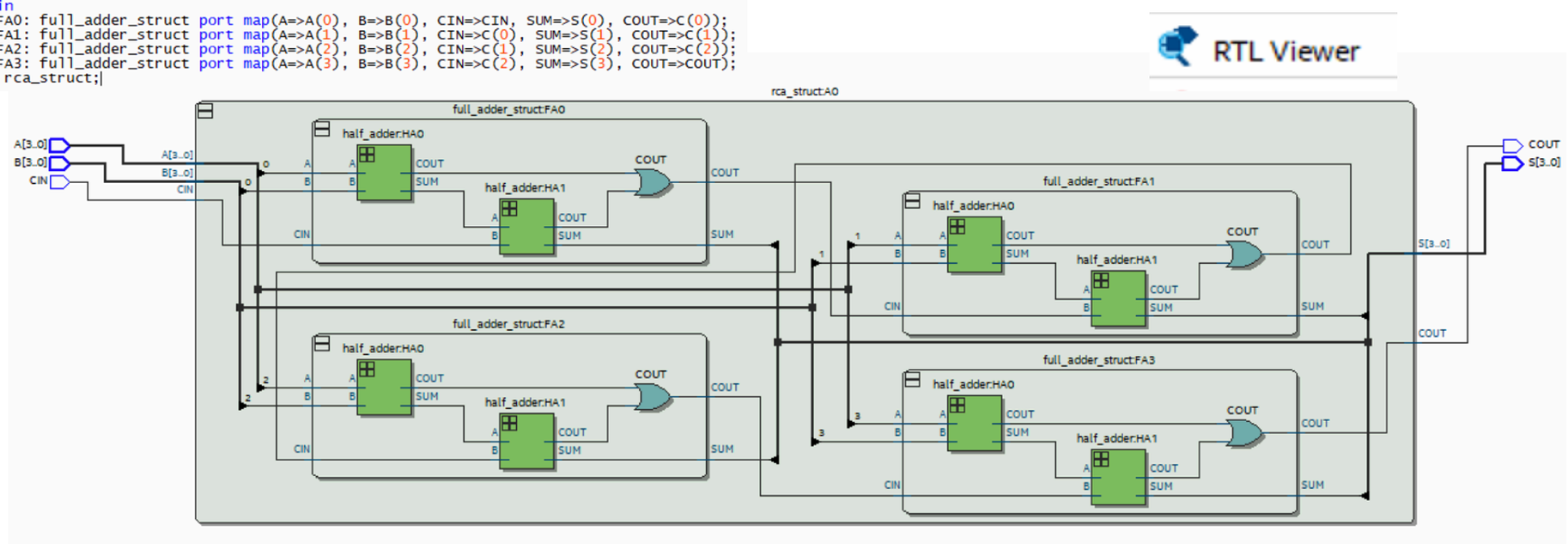
Ripple-Carry Adder VHDL

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3
4   entity rca is
5      port (A, B : in std_logic_vector(3 downto 0);
6            CIN : in std_logic;
7            S : out std_logic_vector(3 downto 0);
8            COUT : out std_logic);
9   end rca;
10
11  architecture rca_struct of rca is
12     component full_adder
13        port (A, B, CIN: in std_logic;
14              SUM, COUT: out std_logic);
15     end component;
16     signal C : std_logic_vector(2 downto 0);
17
18  begin
19     FA0: full_adder port map(A=>A(0), B=>B(0), CIN=>CIN,
20                              SUM=>S(0), COUT=>C(0));
21     FA1: full_adder port map(A=>A(1), B=>B(1), CIN=>C(0),
22                              SUM=>S(1), COUT=>C(1));
23     FA2: full_adder port map(A=>A(2), B=>B(2), CIN=>C(1),
24                              SUM=>S(2), COUT=>C(2));
25     FA3: full_adder port map(A=>A(3), B=>B(3), CIN=>C(2),
26                              SUM=>S(3), COUT=>COUT);
27  end rca_struct;
```



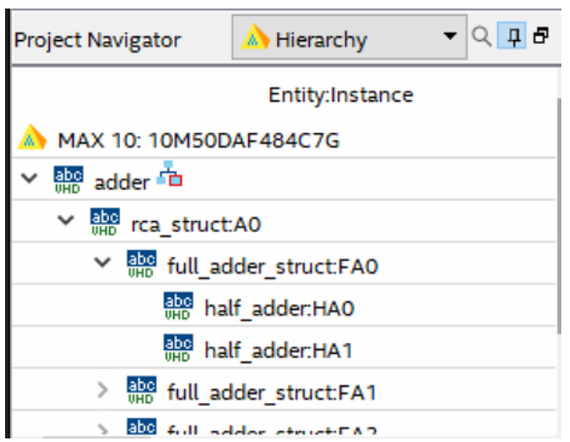
Ripple-Carry Adder VHDL

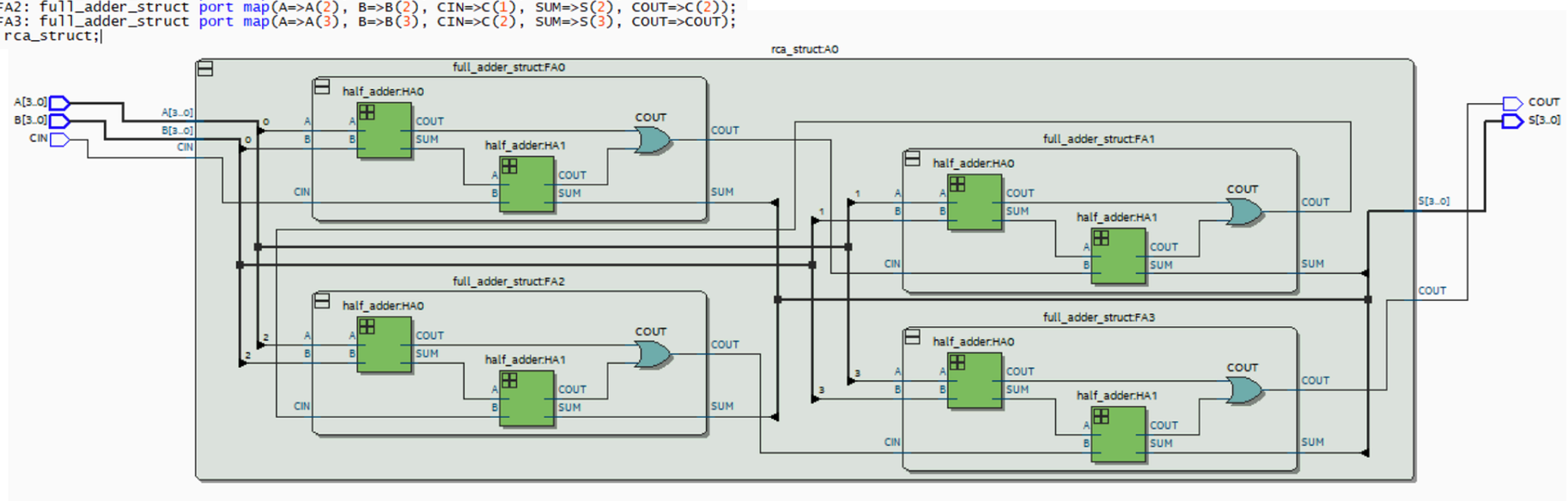
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity rca_struct is
   port (A, B : in std_logic_vector(3 downto 0);
         CIN : in std_logic;
         S : out std_logic_vector(3 downto 0);
         COUT : out std_logic);
end rca_struct;

architecture rca_struct of rca_struct is
   component full_adder_struct
      port (A, B, CIN: in std_logic;
            SUM, COUT: out std_logic);
   end component;
   signal C : std_logic_vector(2 downto 0);

begin
   FA0: full_adder_struct port map(A=>A(0), B=>B(0), CIN=>CIN, SUM=>S(0), COUT=>C(0));
   FA1: full_adder_struct port map(A=>A(1), B=>B(1), CIN=>C(0), SUM=>S(1), COUT=>C(1));
   FA2: full_adder_struct port map(A=>A(2), B=>B(2), CIN=>C(1), SUM=>S(2), COUT=>C(2));
   FA3: full_adder_struct port map(A=>A(3), B=>B(3), CIN=>C(2), SUM=>S(3), COUT=>COUT);
end rca_struct;
```



Ripple-Carry Adder VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity rca_struct is
    port (A, B : in std_logic_vector(3 downto 0);
          CIN : in std_logic;
          S : out std_logic_vector(3 downto 0);
          COUT : out std_logic);
end rca_struct;

architecture rca_struct of rca_struct is
    component full_adder_struct
        port (A, B, CIN: in std_logic;
              SUM, COUT: out std_logic);
    end component;
    signal C : std_logic_vector(2 downto 0);

begin
    FA0: full_adder_struct port map(A=>A(0), B=>B(0), CIN=>CIN, SUM=>S(0), COUT=>C(0));
    FA1: full_adder_struct port map(A=>A(1), B=>B(1), CIN=>C(0), SUM=>S(1), COUT=>C(1));
    FA2: full_adder_struct port map(A=>A(2), B=>B(2), CIN=>C(1), SUM=>S(2), COUT=>C(2));
    FA3: full_adder_struct port map(A=>A(3), B=>B(3), CIN=>C(2), SUM=>S(3), COUT=>COUT);
end rca_struct;
```

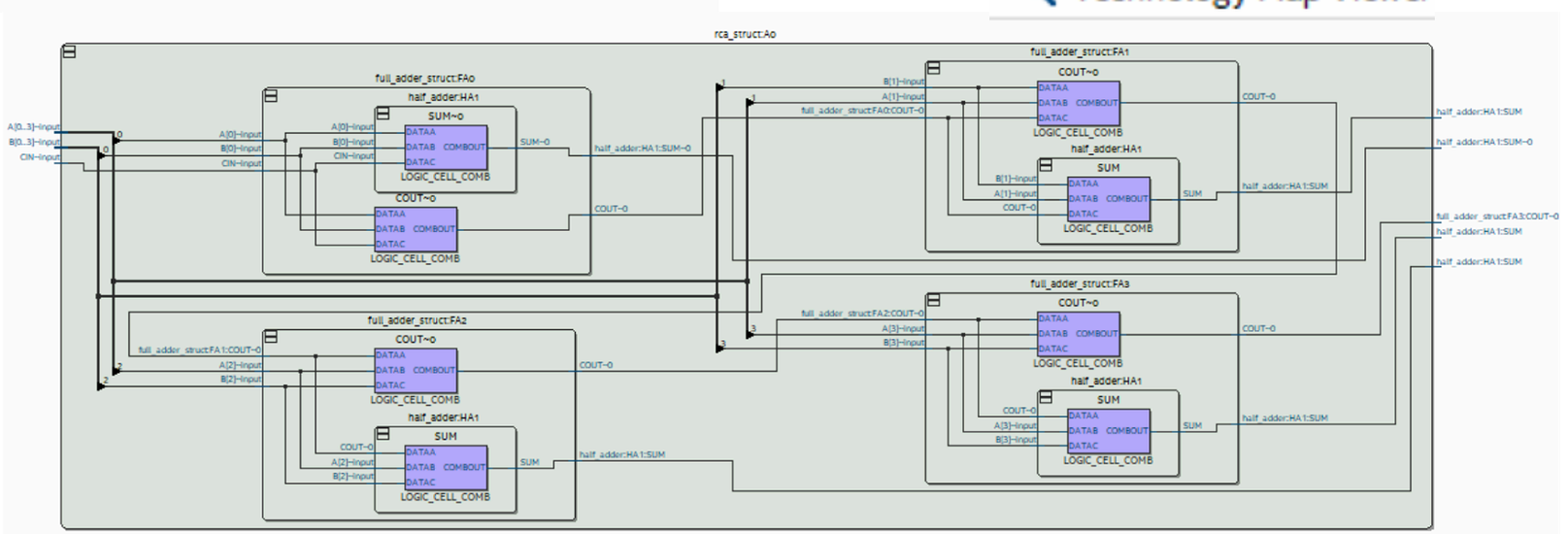




Ripple-Carry Adder VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity rca_struct is
   port (A, B : in std_logic_vector(3 downto 0);
         CIN : in std_logic;
         S : out std_logic_vector(3 downto 0);
         COUT : out std_logic);
end rca_struct;

architecture rca_struct of rca_struct is
   component full_adder_struct
      port (A, B, CIN: in std_logic;
            SUM, COUT: out std_logic);
   end component;
   signal C : std_logic_vector(2 downto 0);

begin
   FA0: full_adder_struct port map(A=>A(0), B=>B(0), CIN=>CIN, SUM=>S(0), COUT=>C(0));
   FA1: full_adder_struct port map(A=>A(1), B=>B(1), CIN=>C(0), SUM=>S(1), COUT=>C(1));
   FA2: full_adder_struct port map(A=>A(2), B=>B(2), CIN=>C(1), SUM=>S(2), COUT=>C(2));
   FA3: full_adder_struct port map(A=>A(3), B=>B(3), CIN=>C(2), SUM=>S(3), COUT=>COUT);
end rca_struct;
```

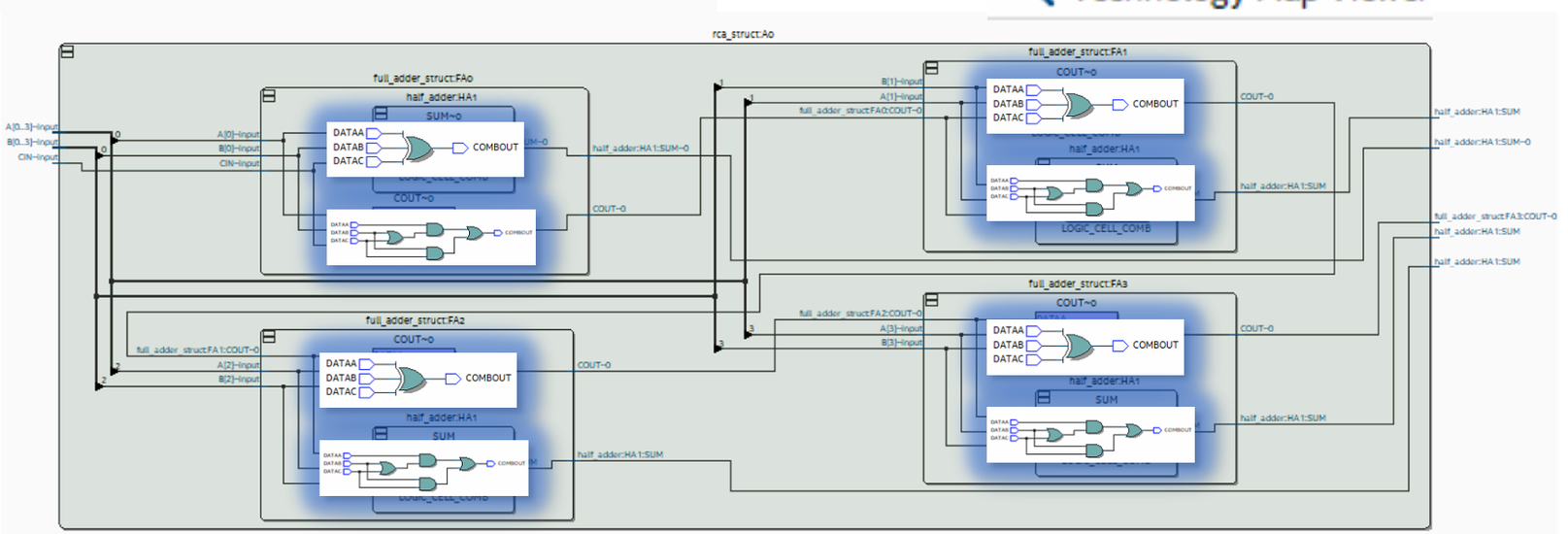
Technology Map Viewer



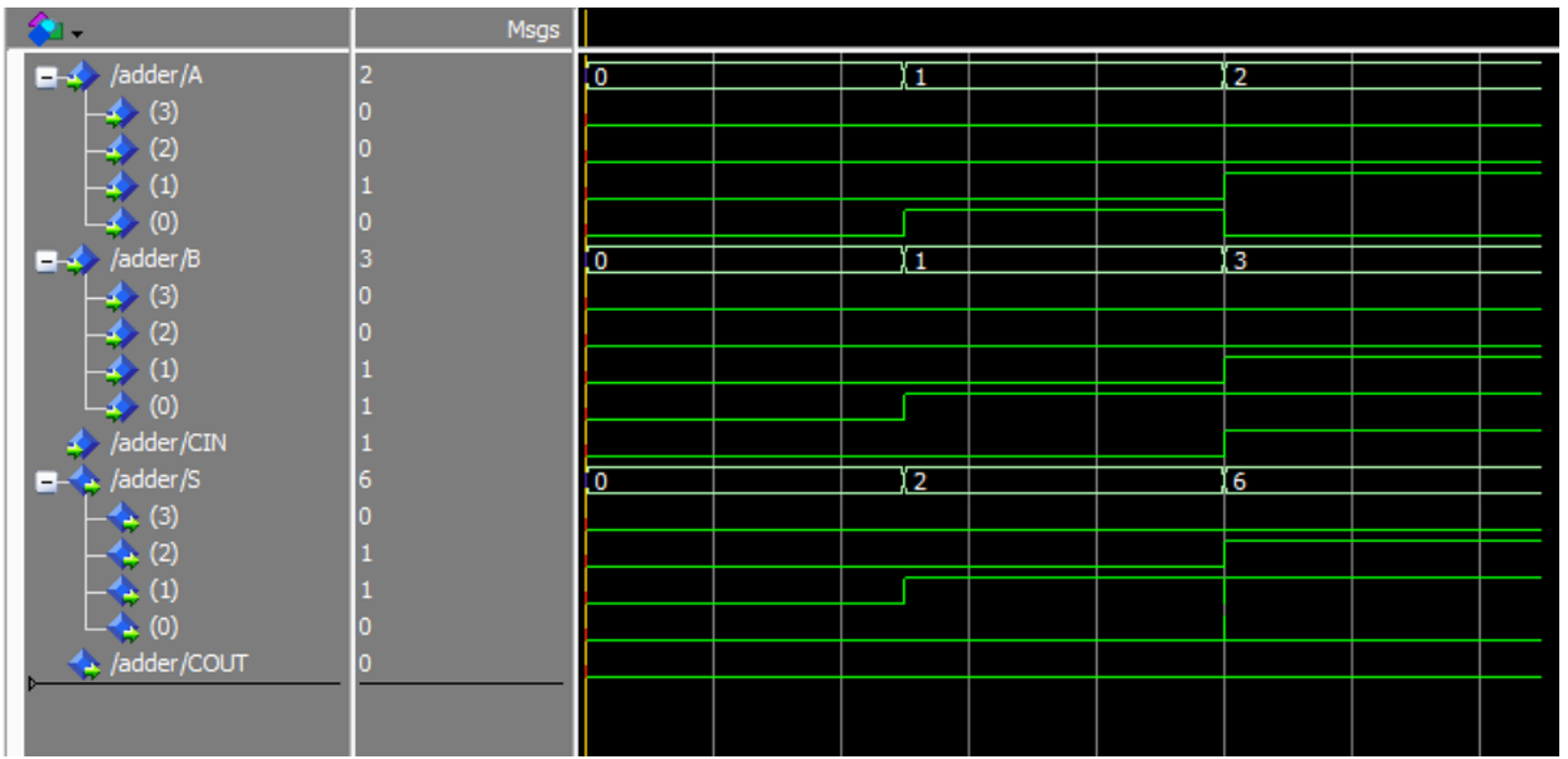
Ripple-Carry Adder VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity rca_struct is
   port (A, B : in std_logic_vector(3 downto 0);
         CIN : in std_logic;
         S : out std_logic_vector(3 downto 0);
         COUT : out std_logic);
end rca_struct;

architecture rca_struct of rca_struct is
   component full_adder_struct
      port (A, B, CIN: in std_logic;
            SUM, COUT: out std_logic);
   end component;
   signal C : std_logic_vector(2 downto 0);

begin
   FA0: full_adder_struct port map(A=>A(0), B=>B(0), CIN=>CIN, SUM=>S(0), COUT=>C(0));
   FA1: full_adder_struct port map(A=>A(1), B=>B(1), CIN=>C(0), SUM=>S(1), COUT=>C(1));
   FA2: full_adder_struct port map(A=>A(2), B=>B(2), CIN=>C(1), SUM=>S(2), COUT=>C(2));
   FA3: full_adder_struct port map(A=>A(3), B=>B(3), CIN=>C(2), SUM=>S(3), COUT=>COUT);
end rca_struct;
```

Technology Map Viewer

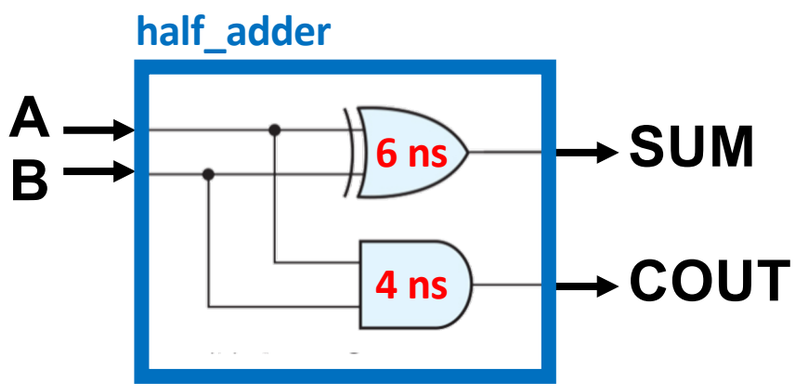


Ripple-Carry Adder Simulation

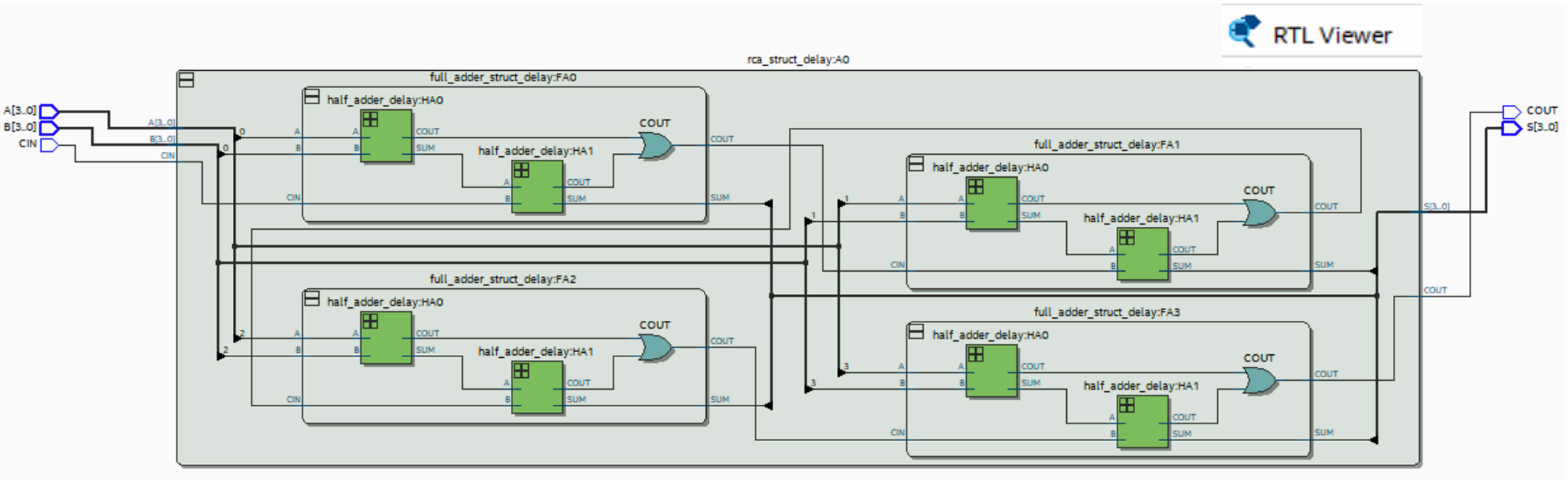


Half Adder VHDL with Gate Delays

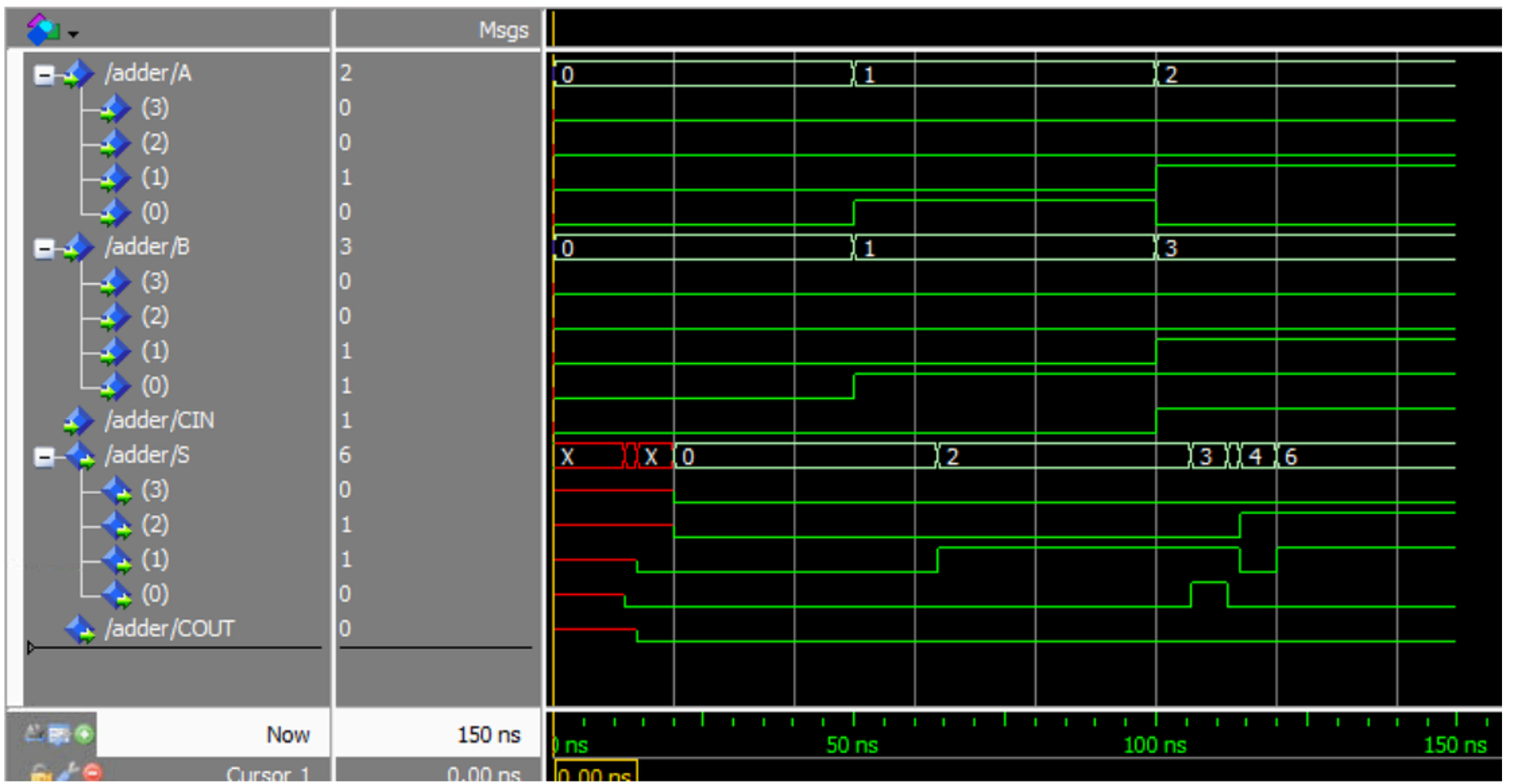
```
 1   library ieee;
 2   use ieee.std_logic_1164.all;
 3
 4
 5   entity half_adder_delay is
 6      port (A, B: in std_logic;
 7            SUM, COUT: out std_logic);
 8   end half_adder_delay;
 9
10
11   architecture ha_xor_delay of half_adder_delay is
12   begin
13       SUM <= A xor B after 6 ns;
14       COUT <= A and B after 4 ns;
15   end ha_xor_delay;
```



Ripple Carry Adder VHDL with Gate Delays



RCA Simulation with Gate Delays



RCA Simulation with Gate Delays