
بسم الله الرحمن الرحيم

ارائه پروژه VHDL

ضرب کننده ۸ بیتی به کمک Pipeline

رضا ادیبی‌سده

۹۶۱۸۴۵۱۰۲

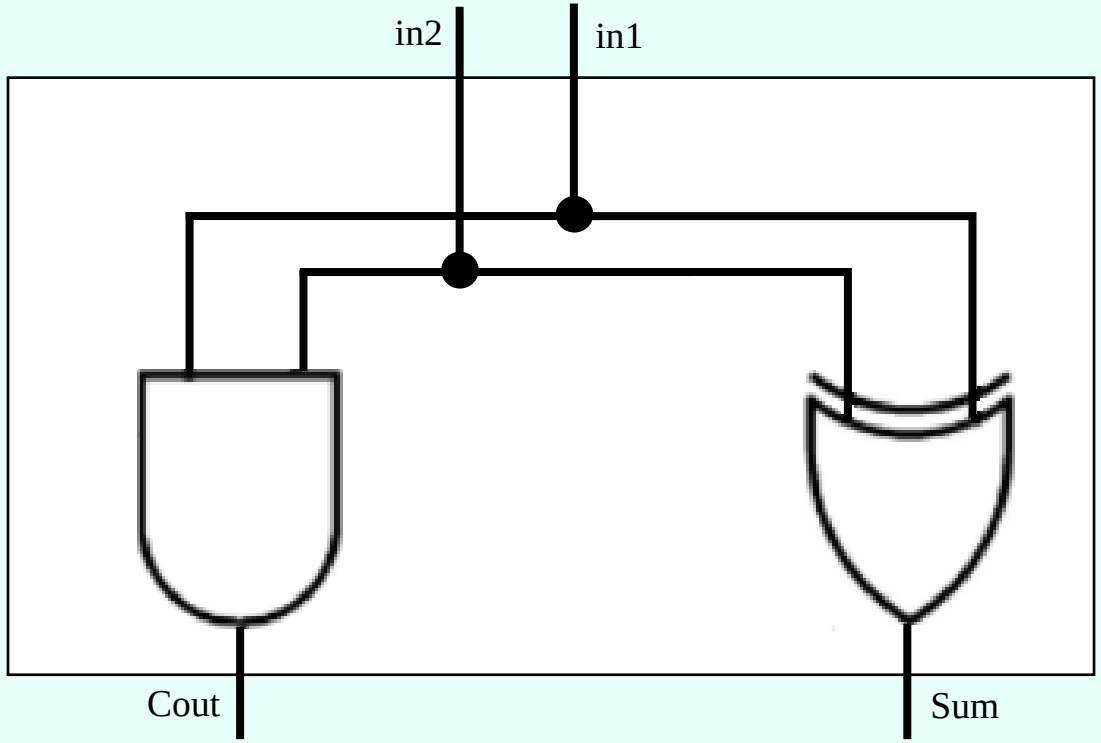
مدرس: استاد حدادیان

Half Adder

کامپوننت Half Adder دو ورودی in1 و in2 را گرفته و خروجی‌های sum و cout را تولید می‌کند که cout نتیجـه عمـل and روی ورودی‌های in1 و in2 بوده و sum نیز نتیجه عمل xor روی ورودی‌های in1 و in2 می‌باشد.

Half Adder

Schema



Half Adder

VHDL Code

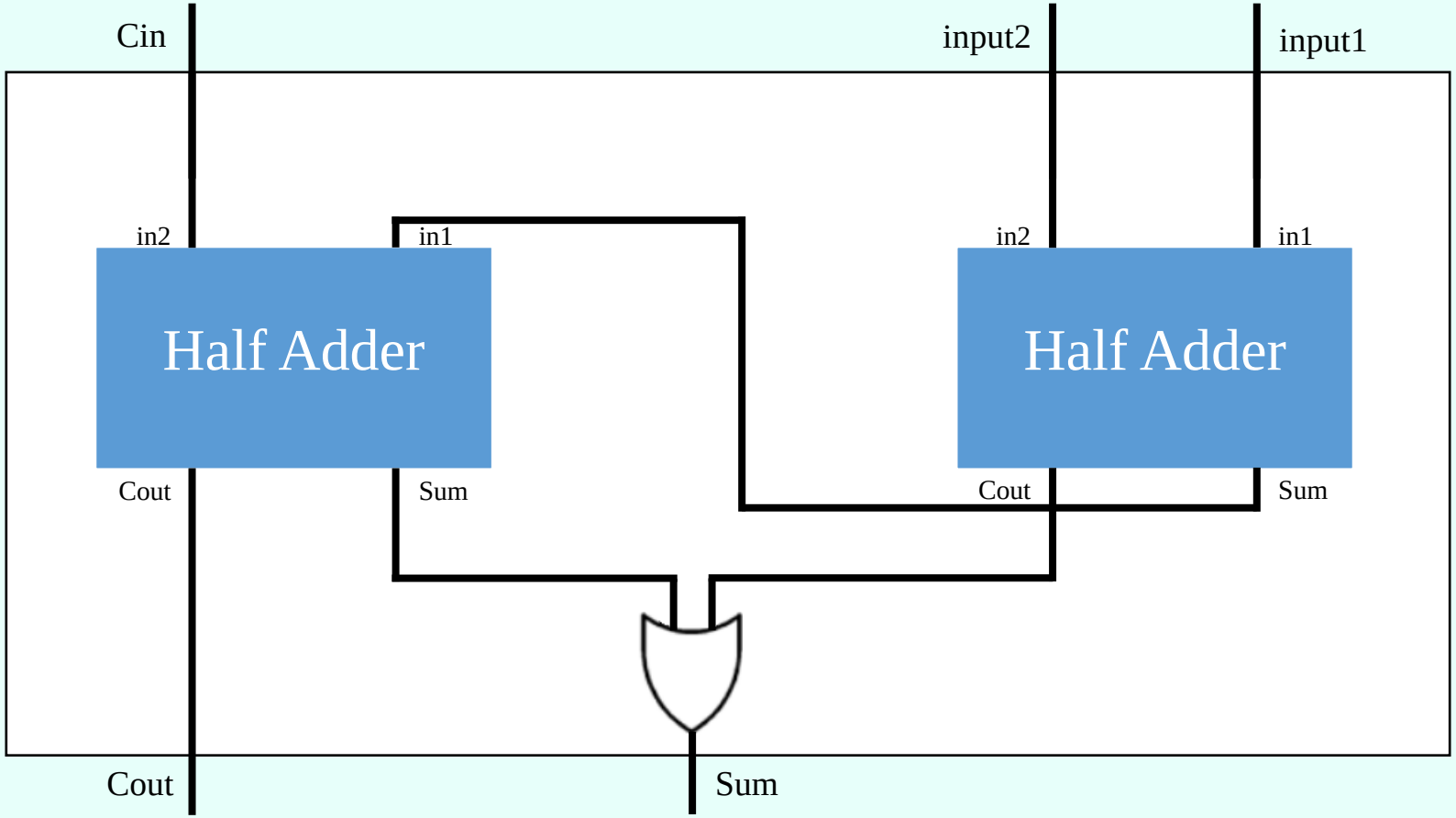
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Half_Adder is
	Port(
		in1, in2: in  STD_LOGIC;
		sum, c: out  STD_LOGIC
	);
end Half_Adder;

architecture Behavioral of Half_Adder is
begin
	sum <= in1 xor in2;
	c <= in1 and in2;
end Behavioral;
```

Full Adder

کامپوننت Full Adder سه ورودی با نام‌های input1، input2 و cin و همچنین دو خروجی با نام‌های sum و cout دارد. در این کامپوننت از دو کامپوننت Half Adder استفاده شده است که ورودی‌های HA1 همان input1 و input2 متعلق به Full Adder می‌باشند و ورودی‌های HA2 پورت cin متعلق به Full Adder و sum خروجی HA1 می‌باشند.
خروجی sum متعلق به Full Adder برابر با حاصل عمل or روی cout متعلق به HA1 و sout متعلق به HA2 بوده و خروجی cout متعلق به Full Adder برابر با مقدار cout متعلق به HA2 می‌باشد.

Full Adder

Schema



Full Adder

VHDL Code

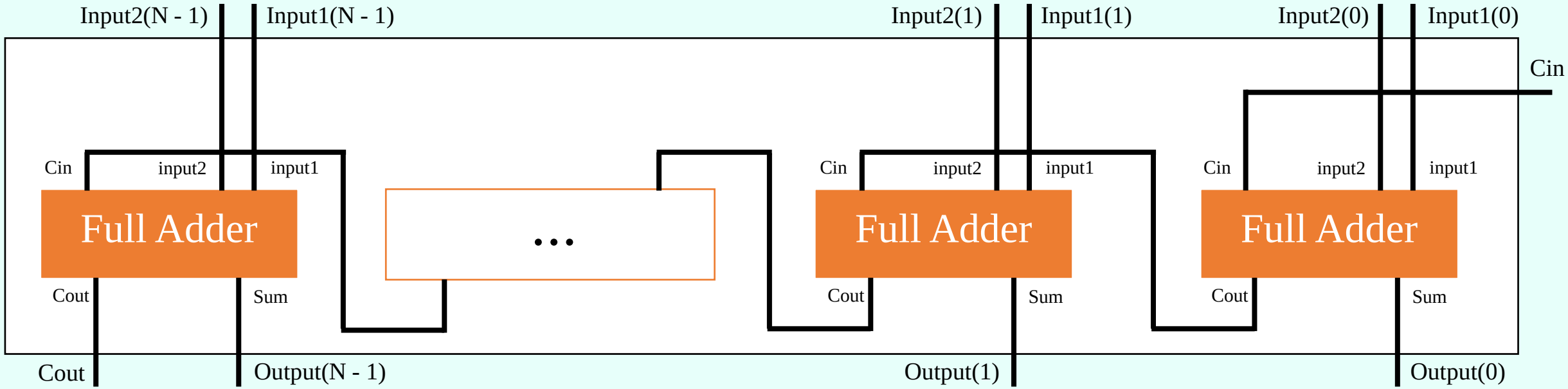
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Full_Adder is
	Port(
		input1, input2, cin: in  STD_LOGIC;
		sum, cout: out  STD_LOGIC
	);
end Full_Adder;
architecture Behavioral of Full_Adder is
	Component Half_Adder is
		Port(
			in1, in2: in  STD_LOGIC;
			sum, c: out  STD_LOGIC
		);
	end Component;
	signal s1, s2, s3: STD_LOGIC;
begin
	HA1: Half_Adder
		port map(
			in1 => input1,
			in2 => input2,
			sum => s1,
			c => s2
		);
	HA2: Half_Adder
		port map(
			in1 => s1,
			in2 => cin,
			sum => sum,
			c => s3
		);
	cout <= s2 or s3;
end Behavioral;
```

N Bit Full Adder

کامپوننت N Bit Full Adder دارای تعداد 2N بیت ورودی برای اعداد Nبیتی input1 و input2 و همچنین یـک ورودی بـرای cin می‌باشد.
این کامپوننت N خروجی برای حاصل جمع اعداد input1 و input2 به نام output و همچنین یک خروجی برای cout دارد.
کامپوننت N Bit Full Adder از تعداد N عدد Full Adder تشکیل شده است که ورودی‌های input1 و input2 هرکدام از آن‌ها به یک بیت از ورودی‌های input1 و input2 متعلق به N Bit Full Adder متصل می‌باشد. ورودی cin اولین Full Adder همـان cin ورودی N Bit Full Adder و ورودی cin سایر Full Adderها خروجی cout متعلق به Full Adder قبلی می‌باشد.
هر بیت از خروجی N بیتی N Bit Full Adder به ترتیب به خروجی sum یکی از Full Adderهـا متصـل اسـت و خروجـی cout متعلق به N Bit Full Adder نیز به خروجی cout متعلق به آخرین Full Adder متصل می‌باشد.

N Bit Full Adder

Schema



N Bit Full Adder

VHDL Code

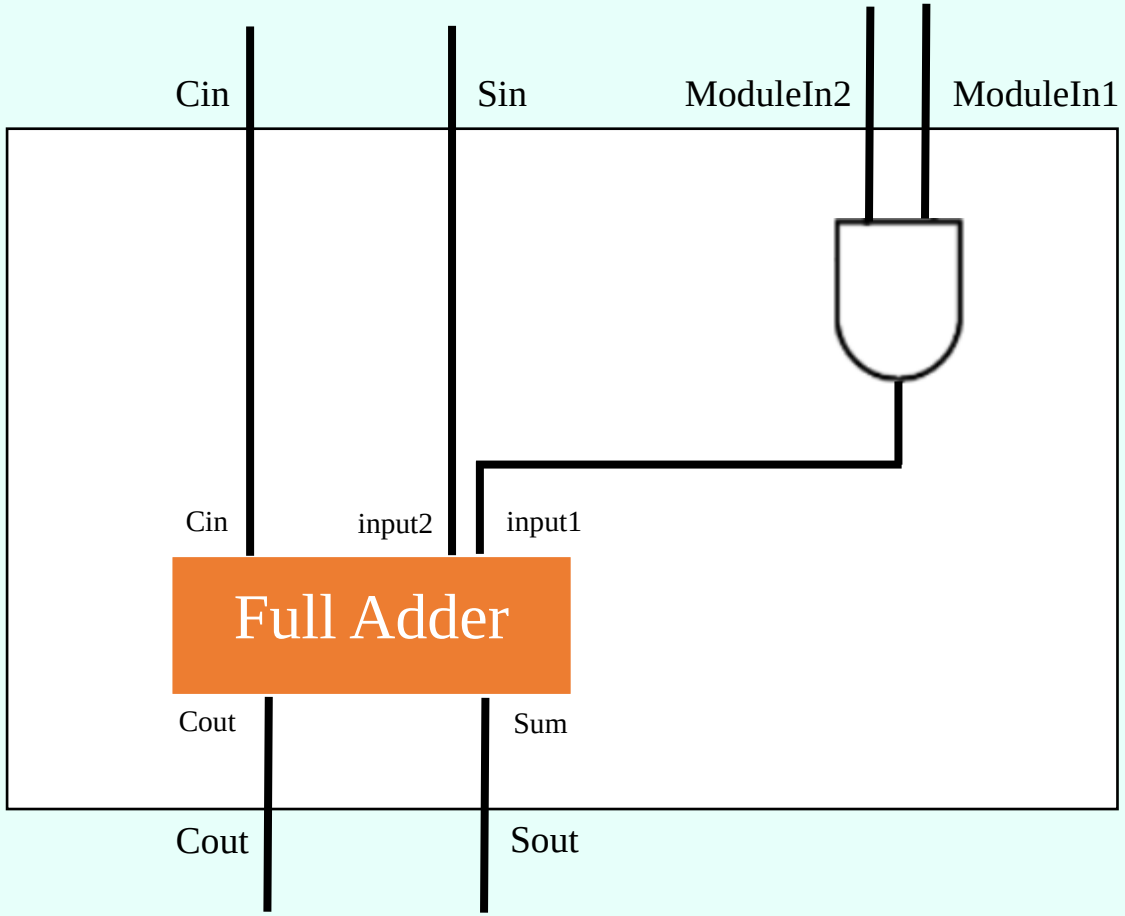
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FullAdder_Nbit is
	Generic(
		N: integer := 8
	);
	Port(
		input1, input2: in STD_LOGIC_VECTOR(N - 1 downto
0);
		cin: in STD_LOGIC;
		output: out STD_LOGIC_VECTOR(N - 1 downto 0);
		cout: out STD_LOGIC
	);
end FullAdder_Nbit;

architecture Behavioral of FullAdder_Nbit is
	Component Full_Adder is
		Port(
			input1, input2, cin: in  STD_LOGIC;
			sum, cout: out  STD_LOGIC
		);
	end Component;
	signal s: STD_LOGIC_VECTOR(0 to N);
begin
	FA_gen: for i in 0 to N - 1 Generate
		FA: Full_Adder
			PORT MAP(
				input1 => input1(i),
				input2 => input2(i),
				cin => s(i),
				sum => output(i),
				cout => s(i + 1)
			);
	end Generate;
	s(0) <= cin;
	cout <= S(N);
end Behavioral;
```

C Module

کامپوننت C Module دارای ورودی‌های moduleIn1، moduleIn2، cin و sin می‌باشد.
در این کامپوننت از یک کامپوننت Full Adder استفاده شده است که ورودی input1 آن حاصل عمل and روی moduleIn1 و moduleIn2 است؛ ورودی input2 آن به ورودی sin و همچنین ورودی cin آن به ورودی cin متعلق به C Module متصل می‌باشد.
خروجی sout کامپوننت C Module همان خروجی sum متعلق به Full Adder و خروجی cout آن همان خروجی cout متعلق به Full Adder می‌باشد.

C Module

Schema



C Module

VHDL Code

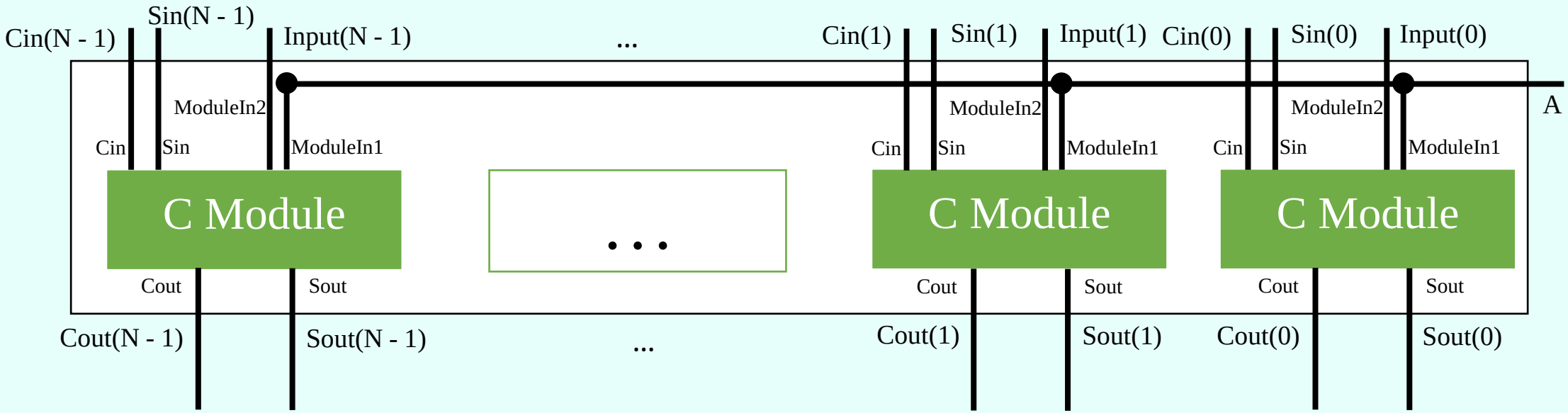
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity C_Module is
	port(
		ModuleIn1, ModuleIn2, Sin, Cin: in STD_LOGIC;
		Sout, Cout: out STD_LOGIC
	);
end C_Module;

architecture Behavioral of C_Module is
	Component Full_Adder is
		Port(
			input1, input2, cin: in  STD_LOGIC;
			sum, cout: out  STD_LOGIC
		);
	end Component;
	signal s1: STD_LOGIC := '0';
begin
	s1 <= ModuleIn1 and ModuleIn2;
	FA: Full_Adder
		port map(
			input1 => s1,
			input2 => Sin,
			cin => Cin,
			sum => Sout,
			cout => Cout
		);
end Behavioral;
```

N Bit Multiplier Row

کامپوننت N Bit Multiplier Row دارای N پایه‌ی ورودی input، N پایه‌ی ورودی sin، N پایه‌ی ورودی cin و یک بیت ورودی A می‌باشد.
این کامپوننت از N عدد کامپوننت C Module تشکیل شده است که ورودی input1 همه آن‌ها به ورودی A، ورودی input2 آن‌ها به یکی از بیت‌های ورودی input، ورودی sin آن‌ها به یکی از بیت‌های ورودی sin و ورودی cin آن‌ها به یکی از بیت‌های ورودی cin از کامپوننت N Bit Multiplier Row متصل است.
بیت‌های صفر تا N-1 خروجی sout متعلق به N Bit Multiplier Row به ترتیب به خروجی‌های sout کامپوننت‌های C Module و به طور مشابه بیت‌های صفر تا N-1 خروجی cout متعلق به N Bit Multiplier Row به ترتیب به خروجی‌های cout کامپوننت‌های C Module وصل شده است.

N Bit Multiplier Row

Schema



N Bit Multiplier Row

VHDL Code

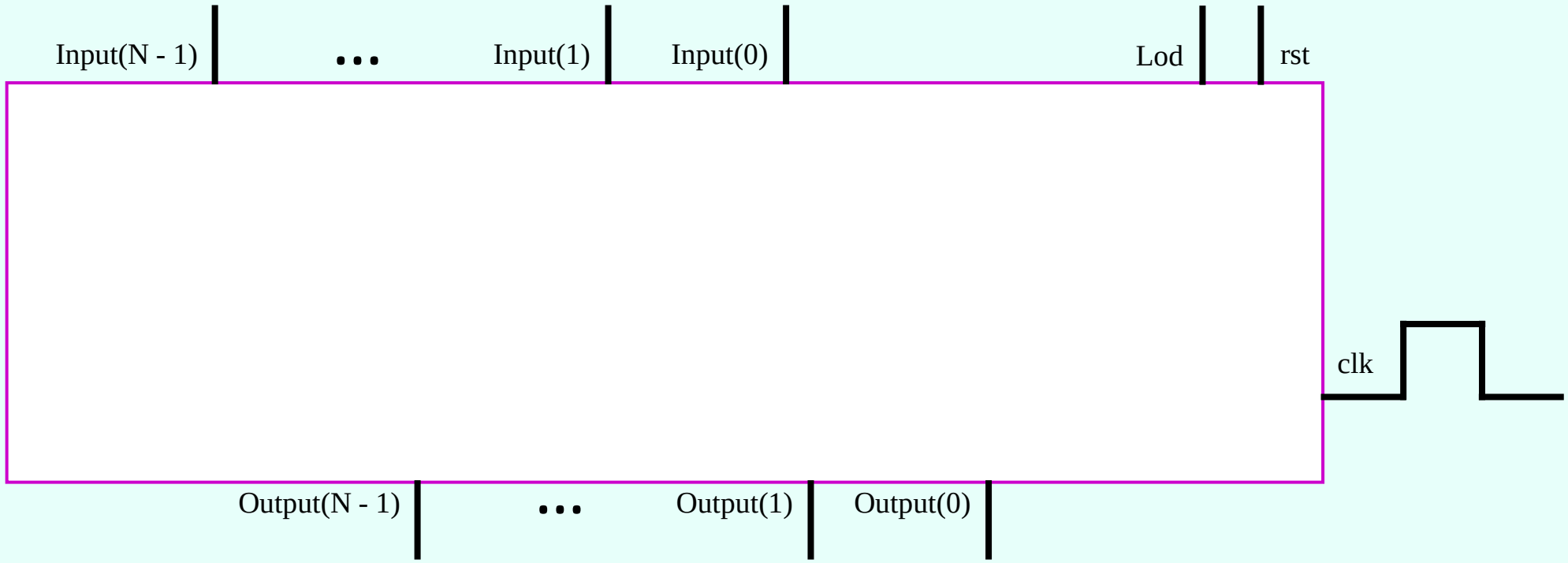
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Multiplier_Row is
	Generic(
		N: integer := 8
	);
	Port(
		A: in STD_LOGIC;
		input, Cin, Sin: in STD_LOGIC_VECTOR(N - 1
downto 0);
		Cout, Sout: out STD_LOGIC_VECTOR(N - 1
downto 0)
	);
end Multiplier_Row;
architecture Behavioral of Multiplier_Row is
	Component C_Module is
		port(
			ModuleIn1, ModuleIn2, Sin, Cin: in
STD_LOGIC;
				Sout, Cout: out STD_LOGIC
			);
	end Component;
begin
	CM_gen: for i in 0 to N - 1 Generate
		CM: C_Module
			port map(
				ModuleIn1 => A,
				ModuleIn2 => input(i),
				Cin => Cin(i),
				Sin => sin(i),
				Cout => Cout(i),
				Sout => Sout(i)
			);
	end Generate;
end Behavioral;
```

N Bit Rigester

کامپوننت N Bit Rigester دارای N بیت ورودی input برای ورود داده و پایه‌های clk برای دریافت کلاک ورودی، rst برای پاک کردن خروجی و Lod برای ارسال داده ورودی به خروجی می‌باشد.
در این کامپوننت در صورتی که در لبه‌ی بالا رونده‌ی کلاک پایه‌ی rst برابر با ’1‘ باشد در خروجی output مقدار تمام صفر مشاهده می‌شود؛ در صورتی که در لبه‌ی بالا رونده‌ی کلاک پایه‌ی Lod برابر با ’1‘ باشد مقدار خروجی output برابر با مقدار ورودی input خواهد بود و اگر پایه‌ی Lod برابر با ’0‘ باشد مقدار خروجی output بدون تغییر می‌ماند تا وقتی که در لبه‌ی بالارونده‌ی کلاک مقدار یکی از پایه‌های rst یا Lod برابر با ’1‘ شود.

N Bit Rigester

Schema



N Bit Rigester

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Rigester is
	Generic(
		N: integer := 16
	);
	port(
		input: in std_logic_vector(N - 1 downto 0);
		rst, Lod: in std_logic;
		clk: in std_logic;
		output: out std_logic_vector(N - 1 downto 0)
	);
end entity;

architecture Behavioral of Rigester is
begin
	process(clk)
	begin
		if (rst = '1') then
			output <= (others => '0');
		elsif Rising_edge(clk) then
			if (Lod = '1') then
				output <= input;
			end if;
		end if;
	end process;
end architecture;
```

Multiplier

مدار ضرب کننده دارای دو ورودی N بیتی input1 و input2 و یک پایه‌ی clk برای کلاک ورودی می‌باشد.
این مدار شامل N کامپوننت N Bit Multiplier Row که مقدار N آن‌ها با مقدار N مدار برابر است به همراه N+1 کامپوننت N Bit Rigester که مقدار N آن دو برابر مقدار N مدار می‌باشد است، به هر زوج از این دو کامپوننت یک سطح(level) می‌گوییم.
در هر سطح برای اتصال خروجی‌های sout و cout متعلق به Multiplier Row به ورودی‌های Rigester از سیگنال‌های sout(برای اتصال بیت‌های 0 تا N-1 پورت sout متعلق به Multiplier Row به بیت‌های 0 تا N-1 پورت input از Rigester) و cout(برای اتصال بیت‌های 0 تا N-1 پورت cout متعلق به Multiplier Row به بیت‌های N تا 2N-1 پورت input از Rigester) مربوط به آن سطح استفاده می‌کنیم.
همچنین بین دو سطح متوالی برای اتصال خروجی Rigester سطح بالاتر به ورودی‌های sin و cin متعلق به Multiplier Row سطح پایین‌تر از سیگنال‌های sin(برای اتصال بیت‌های 1 تا N-1 از خروجی output متعلق به Rigester در کنار یک بیت ’0‘(به عنوان پر ارزش‌ترین بیت) به ورودی sin متعلق به Multiplier Row) و cin(برای اتصال بیت‌های N تا 2N-1 از خروجی output متعلق به Rigester به ورودی sin متعلق به Multiplier Row) سطح پایین‌تر استفاده می‌کنیم.
* Sinها و cinهای سطح 0 همگی مقداری برابر با ’0‘ دارند.
در هر سطح ورودی‌های input متعلق به Multiplier Row به input2 از مدار ضرب کننده متصل می‌باشند و ورودی A هر سطح نیز به بیت هم‌شماره با آن سطح از input1 وصل شده است.

در سطح N-1ام سیگنال‌های sin و cin خروجی از Rigester(سیگنال‌های سطح Nام) به ورودی‌های input1(بیت‌های 1 تا N-1 از سیگنال sin به همراه یک بیت ’0‘ در ارزشمندترین جایگاه) و input2(بیت‌های 0 تا N-1 از سیگنال cin) از یک کامپوننت N Bit Full Adder متصل می‌باشند.

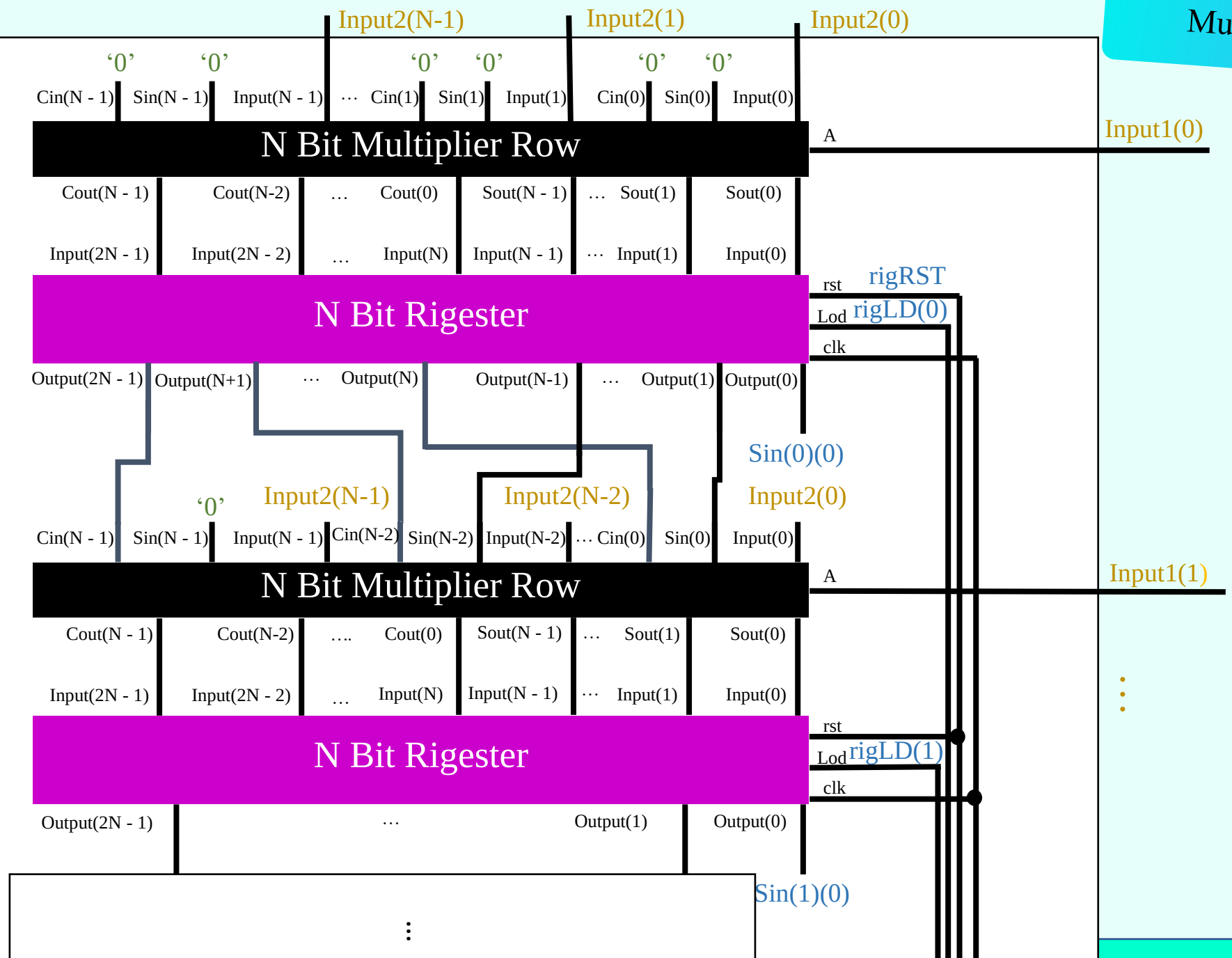
* ورودی cin از N Bit Full Adder برابر با ’0‘ می‌باشد.

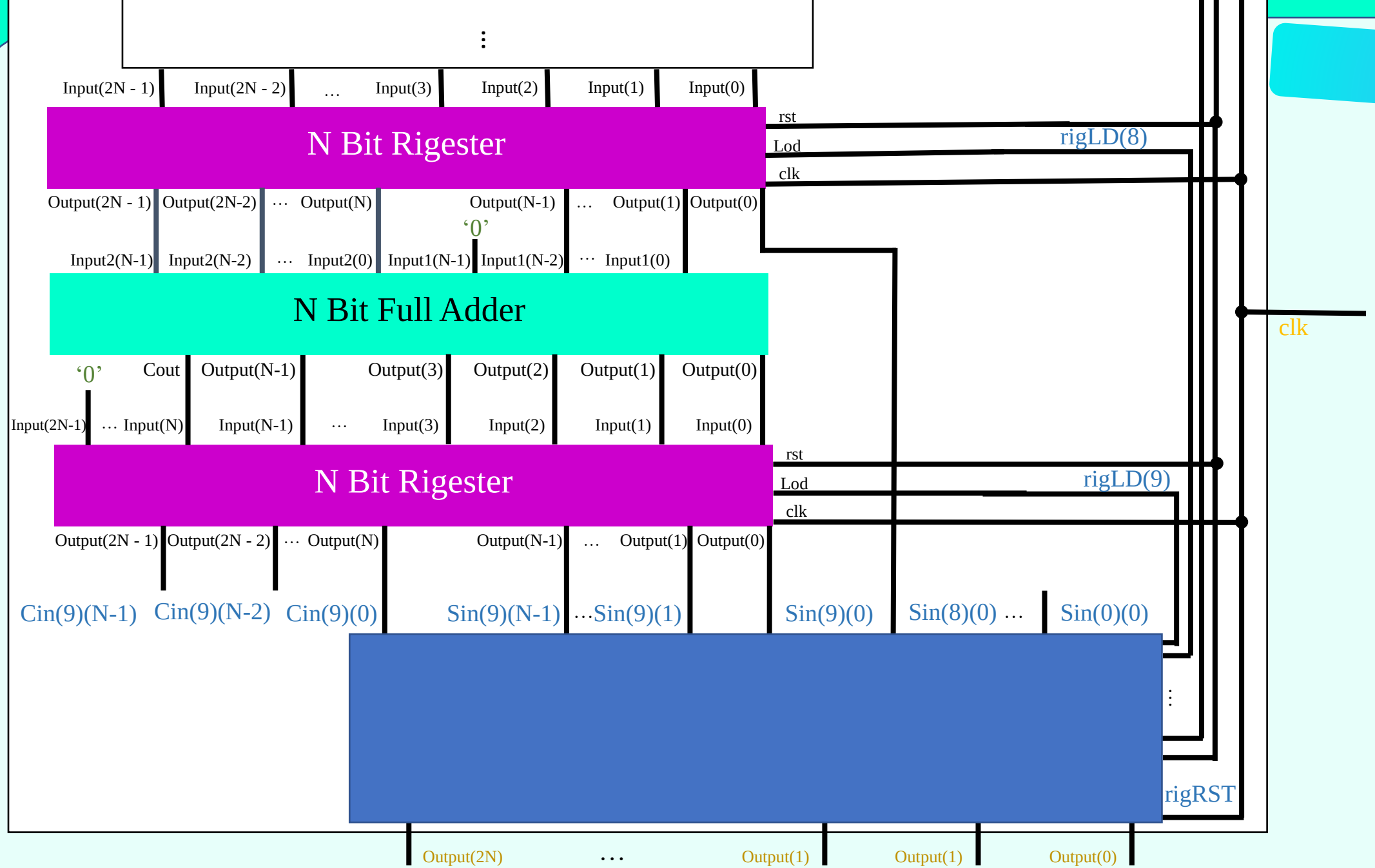
خروجی‌های output از N Bit Full Adder به وسیله سیگنال sin سطح Nام به پایه‌های 0 تا N-1 و خروجی cout از N Bit Full Adder به وسیله بیت صفر سیگنال cin به بیت N از ورودی input متعلق به Rigester این سطح متصل می‌باشد.

در نهایت در بخش process که با هر کلاک ورودی فراخوانی می‌شود اگر clkCounter برابر با صفر بود پایه rst همه Rigesterها که به سیگنال rigRST متصل هستند را برابر با ’0‘ کرده و پایه Lod از Rigester سطح صفر که به بیت صفر از سیگنال rigLD متصل است را برابر با ’1‘ می‌کنیم. اگر مقدار clkCounter بین 1 تا N-1 بود پایه بیت شماره clkCounter از سیگنال rigLD را ’1‘ و بیت قبلی از این سیگنال را ’0‘ می‌کنیم. اگر مقدار clkCounter برابر با N+1 بود فقط بیت Nام از سیگنال rigLD را ’0‘ می‌کنیم. در نهایت اگر مقدار clkCounter برابر با N+2 بود مقادیر بیت صفر از سیگنال‌های sin سطوح 1 تا N را به بیت‌های 0 تا N-1 از خروجی output مدار و بیت‌های 0 تا N-1 از سیگنال‌های sin سطح N+1 به بیت‌های N تا 2N-1 از خروجی output مدار وصل کرده و مقدار clkCounter را برابر با صفر قرار می‌دهیم.

Multiplier

Schema




Input(2N - 1)
Input(2N - 2)
…
Input(3)
Input(2)
Input(1)
Input(0)
N Bit Rigester
rst
Lod
clk
rigLD(8)
Output(2N - 1)
Output(2N-2)
⋯ Output(N)
Output(N-1)
… Output(1)
Output(0)
‘0’
Input2(N-1)
Input2(N-2)
⋯ Input2(0)
Input1(N-1)
Input1(N-2)
⋯ Input1(0)
N Bit Full Adder
‘0’
Cout
Output(N-1)
Output(3)
Output(2)
Output(1)
Output(0)
Input(2N-1)
… Input(N)
Input(N-1)
⋯
Input(3)
Input(2)
Input(1)
Input(0)
N Bit Rigester
rst
Lod
clk
rigLD(9)
Output(2N - 1)
Output(2N - 2)
⋯ Output(N)
Output(N-1)
… Output(1)
Output(0)
Cin(9)(N-1)
Cin(9)(N-2)
Cin(9)(0)
Sin(9)(N-1)
…Sin(9)(1)
Sin(9)(0)
Sin(8)(0) …
Sin(0)(0)
clk
rigRST
Output(2N)
…
Output(1)
Output(1)
Output(0)
Schema

Multiplier

VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Multiplier is
	Generic(
		N: integer := 8
	);
	Port(
		clk: in STD_LOGIC;
		input1, input2: in STD_LOGIC_VECTOR(N - 1
downto 0);
		output: out STD_LOGIC_VECTOR(N * 2 downto 0)
:= (others => '0')
	);
end Multiplier;

architecture Behavioral of Multiplier is
	Type Row is Array (Integer range<>) of
STD_LOGIC_VECTOR(N - 1 downto 0);
	Component Multiplier_Row is
		Generic(
			N: integer := 8
		);
		Port(
			A: in STD_LOGIC;
			input, Cin, Sin: in STD_LOGIC_VECTOR(N -
1 downto 0);
			Cout, Sout: out STD_LOGIC_VECTOR(N - 1
downto 0)
		);
	end Component;
	Component FullAdder_Nbit is
		Generic(
			N: integer := 8
		);
		Port(
			input1, input2: in STD_LOGIC_VECTOR(N - 1
downto 0);
			cin: in STD_LOGIC;
			output: out STD_LOGIC_VECTOR(N - 1
downto 0);
			cout: out STD_LOGIC
		);
	end Component;
```

Multiplier

VHDL Code

```
	Component Rigester is
		Generic(
			N: integer := 16
		);
		port(
			input: in std_logic_vector(N - 1 downto 0);
			rst, Lod: in std_logic;
			clk: in std_logic;
			output: out std_logic_vector(N - 1 downto 0)
		);
	end Component;
	signal Cout, Sout: Row(0 to N);
	signal Cin, Sin: Row(0 to N + 1);
	signal rigLD: STD_LOGIC_VECTOR(0 to N) := (others
=> '0');
	signal rigRST: STD_LOGIC := '1';
	signal clkCounter: INTEGER RANGE 0 to N + 2 := 0;
begin
	MR_gen: for i in 0 to N - 1 Generate
		MR:  Multiplier_Row
			generic map(
				N => N
			)
			port map(
				A => input1(i),
				input => input2,
				Cin => Cin(i),
				Sin(N - 1 downto 0) => '0' & Sin(i)(N - 1
downto 1),
				Cout => Cout(i),
				Sout => Sout(i)
			);
	end Generate;
```

```
Rig_gen: for i in 0 to N Generate
        Rig:  Rigester
                generic map(
                        N => N * 2
                )
                port map(
                        input(N - 1 downto 0) => Sout(i),
                        input(N * 2 - 1 downto N) => Cout(i),
                        rst => rigRST,
                        Lod => rigLD(i),
                        clk => clk,
                        output(N - 1 downto 0) => Sin(i + 1),
                        output(N * 2 - 1 downto N) => Cin(i + 1)
                );
end Generate;
Cin(0) <= (others => '0');
Sin(0) <= (others => '0');

FA: FullAdder_Nbit
                generic map(
                        N => N
                )
                port map(
                        input1(N - 1 downto 0) => '0' & Sin(N)(N - 1
downto 1),
                        input2 => Cin(N),
                        cin => '0',
                        cout => Cout(N)(0),
                        output => Sout(N)
                );
```

Multiplier

VHDL Code

```
    process(clk)
    begin
        if rising_edge(clk)then
            if (clkCounter = 0) then
                rigRST <= '0';
                rigLD(0) <= '1';
                clkCounter <= clkCounter + 1;
            elsif (clkCounter = N + 1) then
                rigLD(N) <= '0';
                clkCounter <= clkCounter + 1;
            elsif (clkCounter = N + 2) then
                output(0 to N - 1) <= Sin(1 to N)(0);
                output((N * 2) - 1 downto N) <= Sin(N +
1);
                clkCounter <= 0;
            else
                rigLD(clkCounter - 1) <= '0';
                rigLD(clkCounter) <= '1';
                clkCounter <= clkCounter + 1;
                end if;
            end if;
        end process;
    end Behavioral;
```

پایان