



دوره مقدماتی آشنایی با FPGA و VHDL

کد همروند

محمدرضا عزیزی

امیرعلی ابراهیمی

بهار ۱۳۹۷

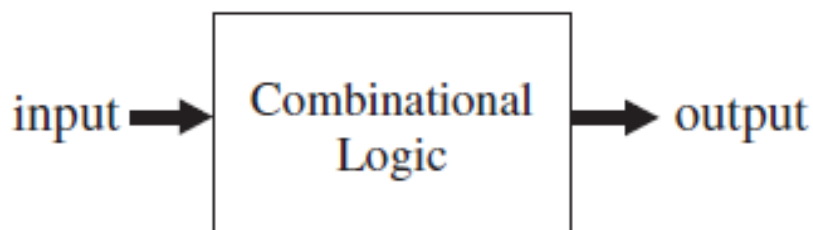
همروند در مقابل ترتیبی

- این فصل را با مرور تفاوت های اساسی مد/ر های ترکیبی (combinational) و ترتیبی (sequential) شروع می کنیم.
- در ادامه، تفاوت های کدهای همروند (Concurrent) و ترتیبی را بررسی می کنیم.
- با ساختار کدهای همروند در زبان VHDL آشنا می شویم.
- با استفاده از ساختار کد همروند، مثال هایی آموزشی و کاربردی را در میان و انتهای فصل بررسی می کنیم.

مدار ترکیبی در مقابل مدار ترتیبی (۱)

➤ مدار ترکیبی:

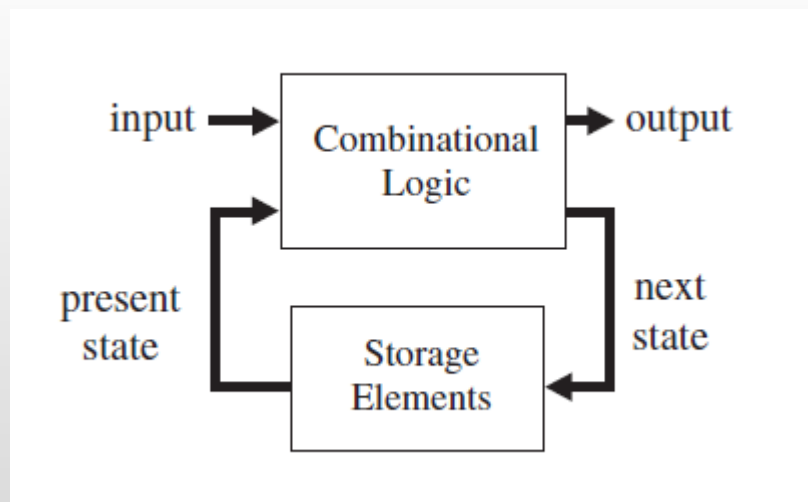
- خروجی مدار فقط به ورودی‌های مدار وابسته است.
- به صورت کلی، سیستم به واحدهای حافظه نیازی ندارد.
- با گیت‌های منطقی قابل پیاده‌سازی است.



مدار ترکیبی در مقابل مدار ترتیبی (۲)

➤ مدار ترتیبی:

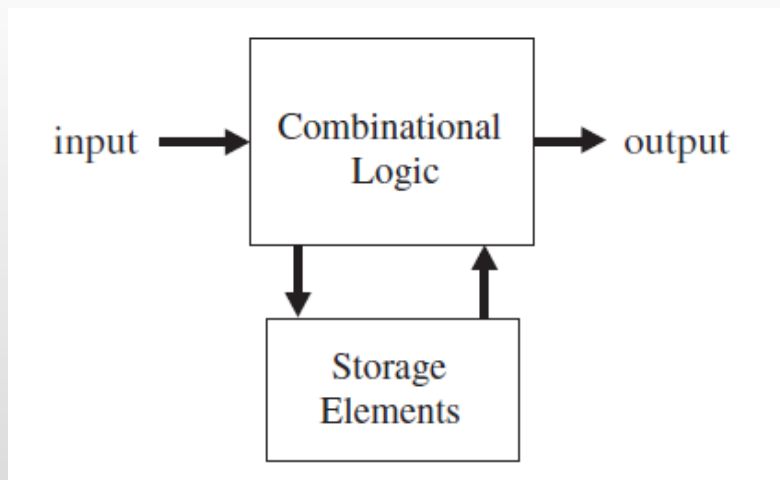
- خروجی مدار علاوه بر ورودی‌های هم اکنون، به ورودی‌های قبلی هم وابسته است.
- سیستم به واحدهای حافظه نیاز دارد.
- این واحدهای حافظه توسط حلقه *feedback* به یک مدار ترکیبی متصل می‌شوند.
 - پس ورودی‌های قبلی، در خروجی فعلی مدار تاثیر می‌گذارند.



مدار ترکیبی در مقابل مدار ترتیبی (۳)

➤ بررسی یک اشکال رایج:

- هر مداری که واحد حافظه (فلیپ فلاپها) داشته باشد، لزوماً مدار ترتیبی نیست.
- به عنوان مثال، پیاده‌سازی RAM (Random Access Memory) می‌تواند به صورت شکل زیر باشد.
- در این شکل، واحدهای حافظه به صورت مستقیم با مدار ترکیبی در ارتباط هستند و نه به حالت *feedback*.



- پس ورودی‌های قبلی، در خروجی فعلی مدار تاثیر **نمی‌گذارند**. ← مدار ترکیبی

کد همروند در مقابل کد ترتیبی (۱)

➤ کد VHDL به صورت پیش فرض، همروند (concurrent) یا موازی است.

➤ تنها کدهایی که درون PROCESS، FUNCTION و PROCEDURE نوشته شوند، ترتیبی (sequential) هستند.

■ کدهای داخل این بلوک‌ها ترتیبی هستند اما هر یک از این بلوک‌ها، به صورت کامل، به بلوک‌ها و دستورات دیگر مدار، همروند هستند.

➤ مثال: فرض کنید stat1، stat2 و stat3 سه دستور همروند باشند. هر یک از حالت‌های زیر، مدار فیزیکی یکسانی تولید می‌کند:

stat1		stat3		stat1	
stat2	≡	stat2	≡	stat3	≡ etc.
stat3		stat1		stat2	

کد همروند در مقابل کد ترتیبی (۲)

- با استفاده از کد همروند، تنها مدارهای ترکیبی قابل پیاده سازی هستند.
- برای پیاده سازی مدارهای ترتیبی، باید از کد ترتیبی (فصل ۶) استفاده شود.
 - البته با استفاده از کد ترتیبی، می توان مدارهای ترکیبی را نیز پیاده سازی کرد.
 - در فصل ۶ توضیح داده خواهد شد.
- در این فصل، کد همروند مورد بررسی قرار خواهد گرفت.
 - کدهایی که خارج از PROCESS، FUNCTION و PROCEDURE نوشته می شوند.
 - در کد همروند از دستورات زیر می توان استفاده کرد:
 - عملگرها (فصل ۴)
 - WHEN (WHEN/ELSE or WITH/SELECT/WHEN)
 - GENERATE

استفاده از عملگرها (۱)

➤ با استفاده از عملگرها می توان هر مدار ترکیبی ای را پیاده سازی کرد.

➤ مثال: مالتی پلکسر ۴ به ۱ (با استفاده از عملگرها)

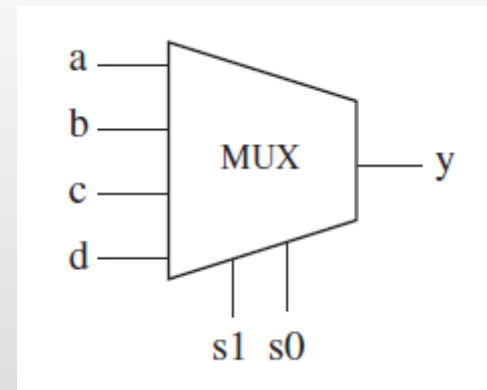
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

-----

ENTITY mux IS
    PORT ( a, b, c, d, s0, s1: IN STD_LOGIC;
          y: OUT STD_LOGIC);
END mux;

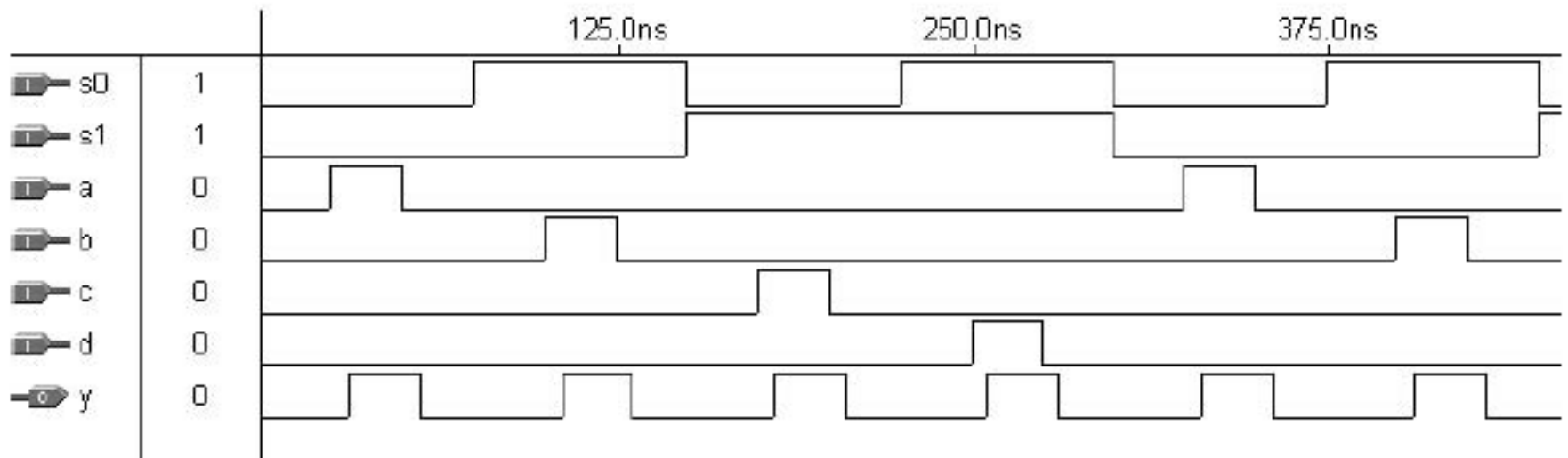
-----

ARCHITECTURE pure_logic OF mux IS
BEGIN
    y <= (a AND NOT s1 AND NOT s0) OR
         (b AND NOT s1 AND s0) OR
         (c AND s1 AND NOT s0) OR
         (d AND s1 AND s0);
END pure_logic;
```



استفاده از عملگرها (۲)

➤ مثال: مالتی پلکسر ۴ به ۱



(I) WHEN (SIMPLE AND SELECTED)

➤ WHEN (حالت ساده):

```
assignment WHEN condition ELSE  
assignment WHEN condition ELSE  
...;
```

➤ WITH / SELECT / WHEN (Selected WHEN):

```
WITH identifier SELECT  
assignment WHEN value,  
assignment WHEN value,  
...;
```

- در استفاده از **WITH / SELECT / WHEN**، تمامی حالت‌ها باید در نظر گرفته شود.
- با کلمه کلیدی **OTHERS** می‌توان برای مجموعه حالت‌های دیگر عملی را انجام داد.
- کلمه کلیدی **UNAFFECTED** برای انجام ندادن هیچ دستوری، استفاده می‌شود.

(۲) WHEN (SIMPLE AND SELECTED)

➤ مثال:

----- With WHEN/ELSE -----

```
outp <= "000" WHEN (inp='0' OR reset='1') ELSE  
        "001" WHEN ctl='1' ELSE  
        "010";
```

---- With WITH/SELECT/WHEN -----

```
WITH control SELECT  
    output <= "000" WHEN reset,  
              "111" WHEN set,  
              UNAFFECTED WHEN OTHERS;
```

(۳) WHEN (SIMPLE AND SELECTED)

➤ در استفاده از WHEN (در حالت ساده) می‌توان شرط‌های نوشته شده در جلوی کلمه کلیدی WHEN را با یکدیگر AND، OR، یا دیگر عملیات منطقی کرد.

```
output <= '0' when input = '1' or input = 'Z';
```

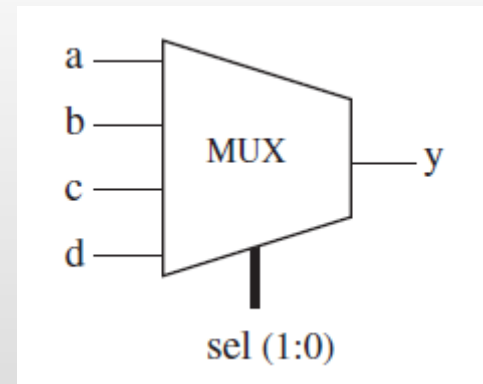
➤ WHEN (در حالت selected) می‌تواند سه فرم مختلف داشته باشد:

- WHEN value -- single value
- WHEN value1 to value2 -- range, for enumerated data types
-- only
- WHEN value1 | value2 | ... -- value1 or value2 or ...

(۴) WHEN (SIMPLE AND SELECTED)

➤ مثال: مالتی پلکسر ۴ به ۱ (با استفاده از WHEN)

```
1 ----- Solution 1: with WHEN/ELSE -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY mux IS
6     PORT ( a, b, c, d: IN STD_LOGIC;
7           sel: IN STD_LOGIC_VECTOR (1 DOWNT0 0);
8           y: OUT STD_LOGIC);
9 END mux;
10 -----
11 ARCHITECTURE mux1 OF mux IS
12 BEGIN
13     y <= a WHEN sel="00" ELSE
14         b WHEN sel="01" ELSE
15         c WHEN sel="10" ELSE
16         d;
17 END mux1;
18 -----
```



(۵) WHEN (SIMPLE AND SELECTED)

1 --- Solution 2: with WITH/SELECT/WHEN -----

2 LIBRARY ieee;

3 USE ieee.std_logic_1164.all;

4 -----

5 ENTITY mux IS

6 PORT (a, b, c, d: IN STD_LOGIC;

7 sel: IN STD_LOGIC_VECTOR (1 DOWNT0 0);

8 y: OUT STD_LOGIC);

9 END mux;

10 -----

11 ARCHITECTURE mux2 OF mux IS

12 BEGIN

13 WITH sel SELECT

14 y <= a WHEN "00", -- notice "," instead of ";"

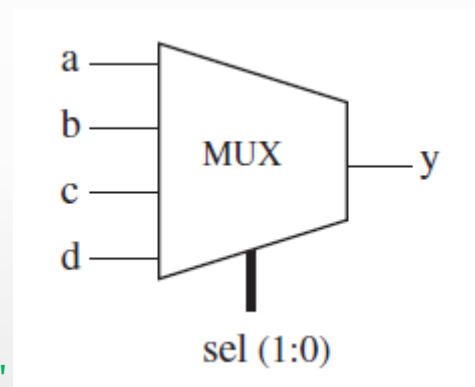
15 b WHEN "01",

16 c WHEN "10",

17 d WHEN OTHERS; -- cannot be "d WHEN "11" " Why?!

18 END mux2;

19 -----



(۶) WHEN (SIMPLE AND SELECTED)

```
1  -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  -----
5  ENTITY mux IS
6      PORT ( a, b, c, d: IN STD_LOGIC;
7              sel: IN INTEGER RANGE 0 TO 3;
8              y: OUT STD_LOGIC);
9  END mux;
10 ---- Solution 1: with WHEN/ELSE -----
11 ARCHITECTURE mux1 OF mux IS
12 BEGIN
13     y <= a WHEN sel=0 ELSE
14         b WHEN sel=1 ELSE
15         c WHEN sel=2 ELSE
16         d;
17 END mux1;
```

(V) WHEN (SIMPLE AND SELECTED)

```
18 -- Solution 2: with WITH/SELECT/WHEN -----
19 ARCHITECTURE mux2 OF mux IS
20 BEGIN
21     WITH sel SELECT
22         y <= a WHEN 0,
23             b WHEN 1,
24             c WHEN 2,
25             d WHEN 3; -- is this correct? Why?
26 END mux2;
27 -----
```

➤ در این حالت، در هنگام شبیه‌سازی می‌توان هر یک از architecture ها را برای شبیه‌سازی انتخاب کرد، اما برای سنتز کردن، باید یکی از architecture ها را کامنت کرد.

▪ همچنین می‌توان از بلوک CONFIGURATION استفاده کرد. ← بیشتر بدانید!

(A) WHEN (SIMPLE AND SELECTED)

➤ مثال: بافر سه حالت:

- این مدار، در صورتی که مقدار `ena` برابر ۰ باشد، مقدار ورودی را بر روی خروجی قرار می‌دهد و در غیر این صورت، مقدار `“ZZZZZZZZ”` را بر روی خروجی قرار می‌دهد.

```
1 LIBRARY ieee;
2 USE ieee.std_logic_1164.all;
3 -----
4 ENTITY tri_state IS
5     PORT ( ena: IN STD_LOGIC;
6           input: IN STD_LOGIC_VECTOR (7 DOWNT0 0);
7           output: OUT STD_LOGIC_VECTOR (7 DOWNT0 0));
8 END tri_state;
9 -----
10 ARCHITECTURE tri_state OF tri_state IS
11 BEGIN
12     output <= input WHEN (ena='0') ELSE
13         (OTHERS => 'Z');
14 END tri_state;
15 -----
```

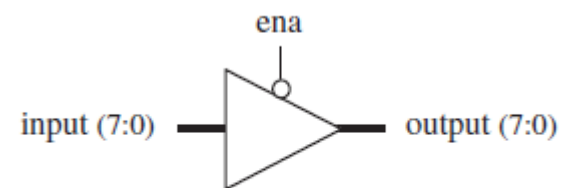


Figure 5.6
Tri-state buffer of example 5.3.

(۹) WHEN (SIMPLE AND SELECTED)

➤ مثال: بافر سه حالت:

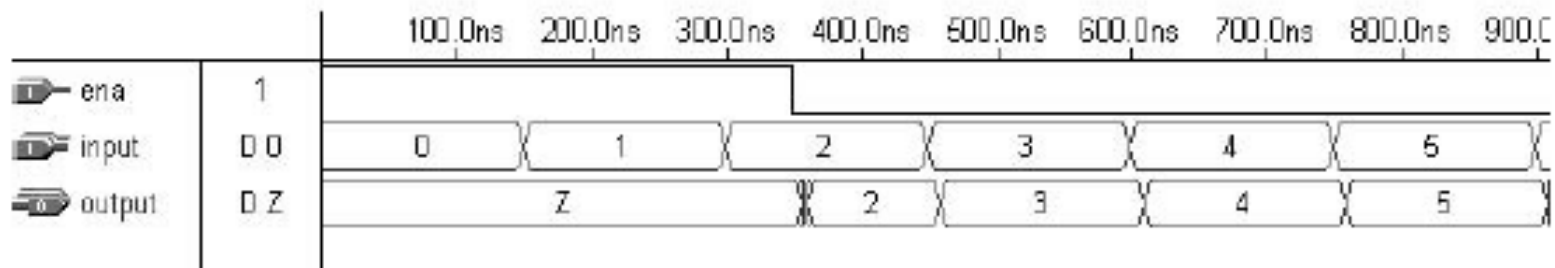


Figure 5.7
Simulation results of example 5.3.

(۱۰) WHEN (SIMPLE AND SELECTED)

➤ مثال: Encoder:

- در هر لحظه فقط یک ورودی می تواند مقدار ۱ داشته باشد که آدرس این ورودی، بر روی سیگنال خروجی قرار داده می شود.

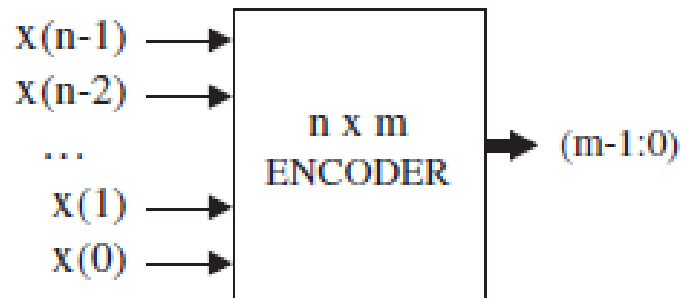


Figure 5.8
Encoder of example 5.4.

(۱۱) WHEN (SIMPLE AND SELECTED)

```
1 ---- Solution 1: with WHEN/ELSE -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY encoder IS
6     PORT ( x: IN STD_LOGIC_VECTOR (7 DOWNT0 0);
7           y: OUT STD_LOGIC_VECTOR (2 DOWNT0 0));
8 END encoder;
9 -----
10 ARCHITECTURE encoder1 OF encoder IS
11 BEGIN
12     y <= "000" WHEN x="00000001" ELSE
13         "001" WHEN x="00000010" ELSE
14         "010" WHEN x="00000100" ELSE
15         "011" WHEN x="00001000" ELSE
16         "100" WHEN x="00010000" ELSE
17         "101" WHEN x="00100000" ELSE
18         "110" WHEN x="01000000" ELSE
19         "111" WHEN x="10000000" ELSE
20         "ZZZ";
21 END encoder1;
22 -----
```

(۱۲) WHEN (SIMPLE AND SELECTED)

```
1 ---- Solution 2: with WITH/SELECT/WHEN -----
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -----
5 ENTITY encoder IS
6     PORT ( x: IN STD_LOGIC_VECTOR (7 DOWNT0 0);
7           y: OUT STD_LOGIC_VECTOR (2 DOWNT0 0));
8 END encoder;
9 -----
10 ARCHITECTURE encoder2 OF encoder IS
11 BEGIN
12     WITH x SELECT
13         y <= "000" WHEN "00000001",
14             "001" WHEN "00000010",
15             "010" WHEN "00000100",
16             "011" WHEN "00001000",
17             "100" WHEN "00010000",
18             "101" WHEN "00100000",
19             "110" WHEN "01000000",
20             "111" WHEN "10000000",
21             "ZZZ" WHEN OTHERS;
22 END encoder2;
23 -----
```

(۱۳) WHEN (SIMPLE AND SELECTED)



		100.0ns		200.0ns		300.0ns		400.0ns		
 x	D0	0	1	2	4	8	16	32	64	128
 y	DZ	Z	0	1	2	3	4	5	6	7

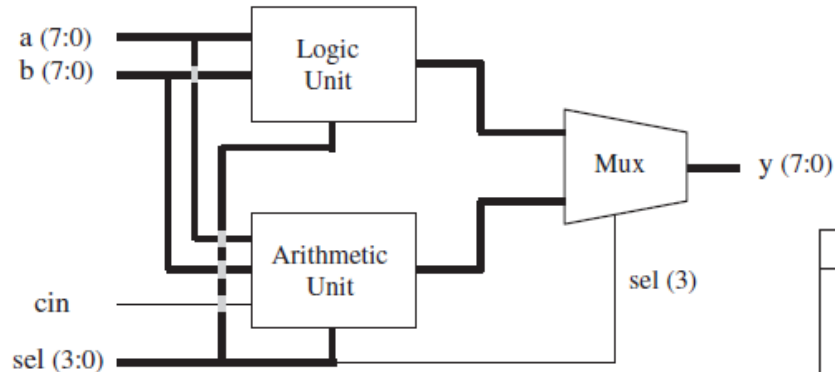
Figure 5.9

Simulation results of example 5.4.

- نکته: تعداد خطوط دستور WHEN در این دو روش زیاد است.
- با استفاده از دستور GENERATE (فصل ۵) یا LOOP (فصل ۶) می‌توان کد کوتاه‌تری برای پیاده‌سازی این سخت‌افزار نوشت.

(۱۴) WHEN (SIMPLE AND SELECTED)

➤ مثال: ALU (Arithmetic Logic Unit)



sel	Operation	Function	Unit
0000	$y \leq a$	Transfer a	Arithmetic
0001	$y \leq a+1$	Increment a	
0010	$y \leq a-1$	Decrement a	
0011	$y \leq b$	Transfer b	
0100	$y \leq b+1$	Increment b	
0101	$y \leq b-1$	Decrement b	
0110	$y \leq a+b$	Add a and b	
0111	$y \leq a+b+cin$	Add a and b with carry	
1000	$y \leq \text{NOT } a$	Complement a	Logic
1001	$y \leq \text{NOT } b$	Complement b	
1010	$y \leq a \text{ AND } b$	AND	
1011	$y \leq a \text{ OR } b$	OR	
1100	$y \leq a \text{ NAND } b$	NAND	
1101	$y \leq a \text{ NOR } b$	NOR	
1110	$y \leq a \text{ XOR } b$	XOR	
1111	$y \leq a \text{ XNOR } b$	XNOR	

(۱۵) WHEN (SIMPLE AND SELECTED)

```
1  -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE ieee.std_logic_unsigned.all;
5  -----
6  ENTITY ALU IS
7      PORT (a, b: IN STD_LOGIC_VECTOR (7 DOWNT0 0);
8            sel: IN STD_LOGIC_VECTOR (3 DOWNT0 0);
9            cin: IN STD_LOGIC;
10           y: OUT STD_LOGIC_VECTOR (7 DOWNT0 0));
11 END ALU;
12 -----
13 ARCHITECTURE dataflow OF ALU IS
14     SIGNAL arith, logic: STD_LOGIC_VECTOR (7 DOWNT0 0);
15 BEGIN
```


(۱۶) WHEN (SIMPLE AND SELECTED)

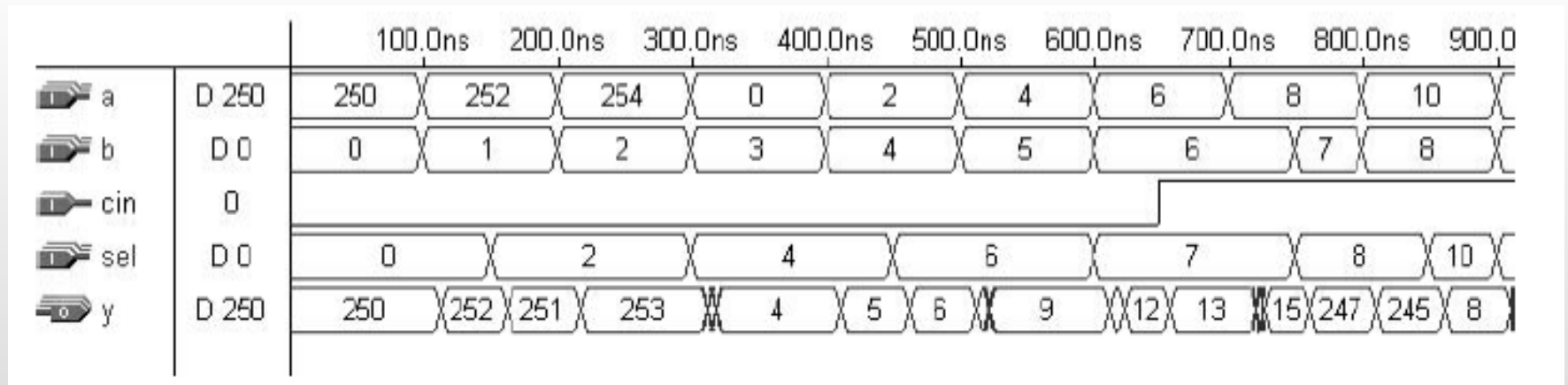
```
16  ----- Arithmetic unit: -----
17  WITH sel(2 DOWNT0 0) SELECT
18      arith <= a WHEN "000",
19          a+1 WHEN "001",
20          a-1 WHEN "010",
21          b WHEN "011",
22          b+1 WHEN "100",
23          b-1 WHEN "101",
24          a+b WHEN "110",
25          a+b+cin WHEN OTHERS;
26  ----- Logic unit: -----
27  WITH sel(2 DOWNT0 0) SELECT
28      logic <= NOT a WHEN "000",
29          NOT b WHEN "001",
30          a AND b WHEN "010",
31          a OR b WHEN "011",
32          a NAND b WHEN "100",
33          a NOR b WHEN "101",
34          a XOR b WHEN "110",
35          NOT (a XOR b) WHEN OTHERS;
```

(IV) WHEN (SIMPLE AND SELECTED)

```

36  ----- Mux: -----
37  WITH sel(3) SELECT
38      y <= arith WHEN '0',
39      logic WHEN OTHERS;
40  END dataflow;
41  -----

```



(۱) GENERATE

➤ تکرار یک بخش کد (مشابه با دستور ترتیبی LOOP (فصل ۶))

➤ باید حتما لیبل گذاری شود.

➤ FOR / GENERATE

```
label: FOR identifier IN range GENERATE  
    (concurrent assignments)  
END GENERATE;
```

➤ چند دستور GENERATE می‌توانند درون یک دیگر nest شوند.

▪ زیرا یک دستور GENERATE، دستور همروند است و درون دستور GENERATE نیز می‌بایست کد همروند نوشته شود.

(۲) GENERATE

➤ مثال:

```
SIGNAL x: BIT_VECTOR (7 DOWNT0 0);  
SIGNAL y: BIT_VECTOR (15 DOWNT0 0);  
SIGNAL z: BIT_VECTOR (7 DOWNT0 0);  
...  
G1: FOR i IN x'RANGE GENERATE  
    z(i) <= x(i) AND y(i+8);  
END GENERATE;
```

➤ نکته مهم: **بازه** دستور GENERATE در صورتی که **استاتیک نباشد**، **سنتزپذیر نخواهد بود**.

```
NotOK: FOR i IN 0 TO choice GENERATE  
    (concurrent statements)  
END GENERATE;
```

(۳) GENERATE

➤ نکته مهم: مراقب سیگنال های multiply-driven یا unresolved باشید!!

OK: FOR i IN 0 TO 7 GENERATE

output(i) <= '1' WHEN (a(i) AND b(i)) = '1' ELSE '0';

END GENERATE; ...

NotOK: FOR i IN 0 TO 7 GENERATE

accum <= "11111111" WHEN (a(i) AND b(i)) = '1' ELSE "00000000";

END GENERATE;

NotOK: For i IN 0 to 7 GENERATE

accum <= accum + 1 WHEN x(i) = '1';

END GENERATE;

(۴) GENERATE

➤ مثال: Vector Shifter:

▪ فرض کنید، مقدار ورودی برابر با “1111” باشد. مقدار خروجی با توجه به مقدار sel یکی از عبارات زیر می تواند باشد:

- row(0): 0 0 0 0 1 1 1 1
- row(1): 0 0 0 1 1 1 1 0
- row(2): 0 0 1 1 1 1 0 0
- row(3): 0 1 1 1 1 0 0 0
- row(4): 1 1 1 1 0 0 0 0

(۵) GENERATE

```
1  -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  -----
5  ENTITY shifter IS
6      PORT ( inp: IN STD_LOGIC_VECTOR (3 DOWNT0 0);
7              sel: IN INTEGER RANGE 0 TO 4;
8              outp: OUT STD_LOGIC_VECTOR (7 DOWNT0 0));
9  END shifter;
10 -----
11 ARCHITECTURE shifter OF shifter IS
12     SUBTYPE vector IS STD_LOGIC_VECTOR (7 DOWNT0 0);
13     TYPE matrix IS ARRAY (4 DOWNT0 0) OF vector;
14     SIGNAL row: matrix;
15 BEGIN
16     row(0) <= "0000" & inp;
17     G1: FOR i IN 1 TO 4 GENERATE
18         row(i) <= row(i-1)(6 DOWNT0 0) & '0';
19     END GENERATE;
20     outp <= row(sel);
21 END shifter;
22 -----
```

(۶) GENERATE

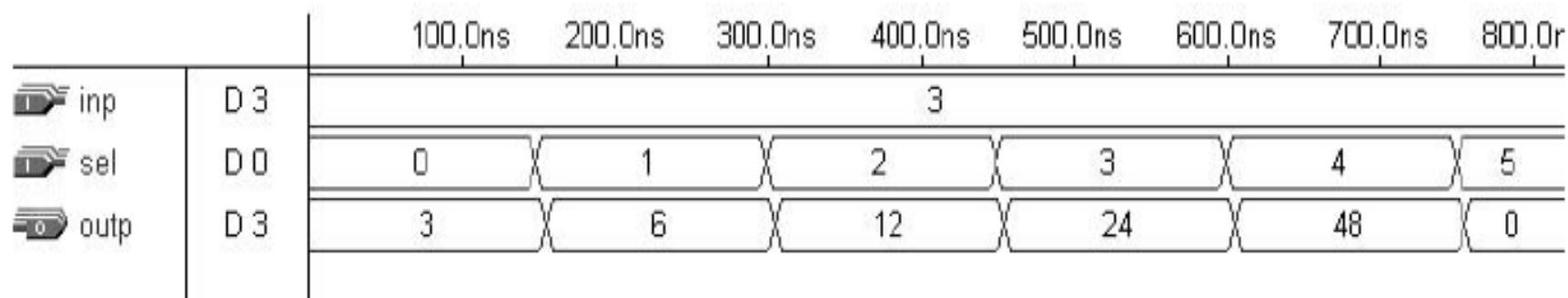


Figure 5.12
Simulation results of example 5.6.

(V) GENERATE

➤ دستور GENERATE شرطی:

```
label: IF condition GENERATE
    generate_statement_body
{ELSIF condition GENERATE
    generate_statement_body}
[ELSE GENERATE
    generate_statement_body]
END GENERATE [label];
```

➤ شرایط، نحوه کپی شدن دستورات را کنترل می کنند.

(Λ) GENERATE

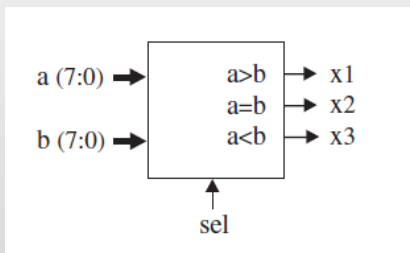
➤ مثال:

```
adder : FOR i IN width-1 DOWNTO 0 GENERATE
    signal carry_chain : unsigned(width-1 downto 1);
BEGIN
    adder_cell: IF i = width-1 GENERATE
        --if statements;
    ELSIF i = 0 GENERATE
        --elsif statements;
    ELSE GENERATE
        --else statements;
    END GENERATE adder_cell;
END GENERATE adder;
```

تمرین

تمرین ۵.۸: مقایسه کننده (comparator) ➤

- مداری طراحی کنید که با توجه به مقدار ورودی sel، دو ورودی ۸ بیتی a و b را با یکدیگر مقایسه کند.
- در صورتی که sel='1'، مقایسه با فرض علامتدار بودن مقدار دو ورودی a و b صورت گیرد.
- در صورتی که sel='0'، مقایسه با فرض بی علامت بودن مقدار دو ورودی a و b صورت گیرد.
- خروجی‌های مدار:
 - x1: در صورتی که نتیجه $a > b$ ، صحیح باشد، ۱ است و در غیر این صورت ۰.
 - x2: در صورتی که نتیجه $a = b$ ، صحیح باشد، ۱ است و در غیر این صورت ۰.
 - x3: در صورتی که نتیجه $a < b$ ، صحیح باشد، ۱ است و در غیر این صورت ۰.



پایان بخش کد همروند