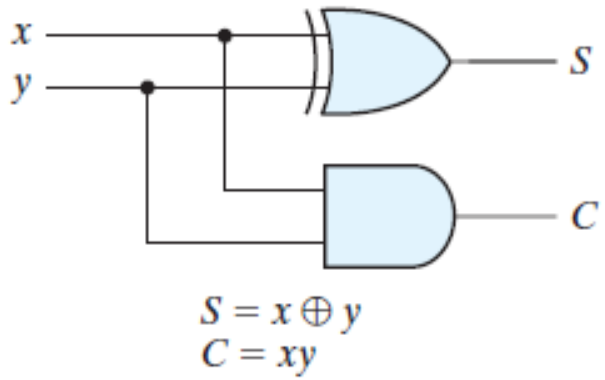
طراحی کامپیوتری سیستم‌های دیجیتال

جلسه پنجم



جمع کننده

نیم جمع کننده Half adder



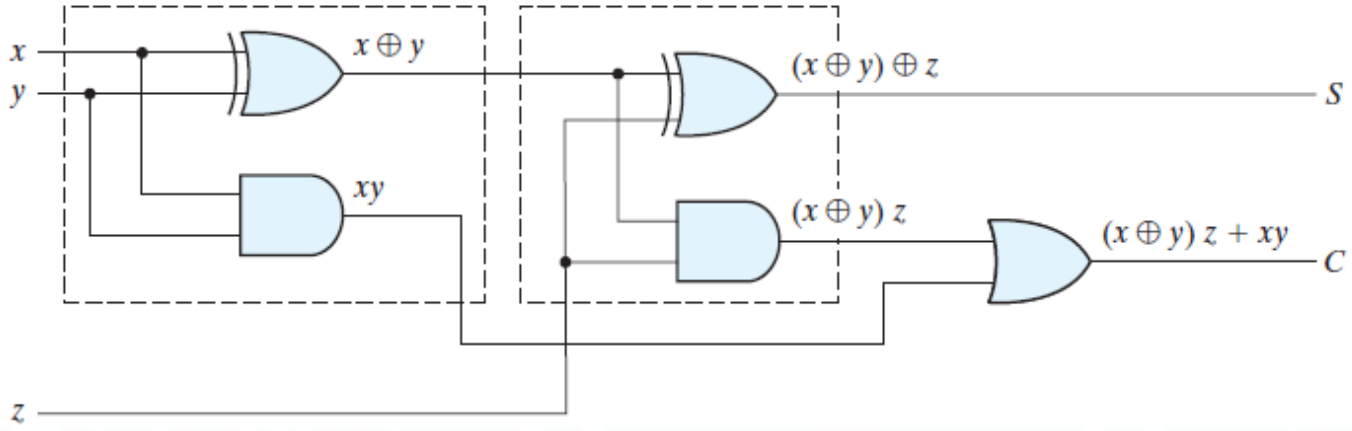
Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

```
module half_adder (output S, C, input x, y);
// Instantiate primitive gates
  xor (S, x, y);
  and (C, x, y);
endmodule
```

جمع کننده

تمام جمع کننده Full adder

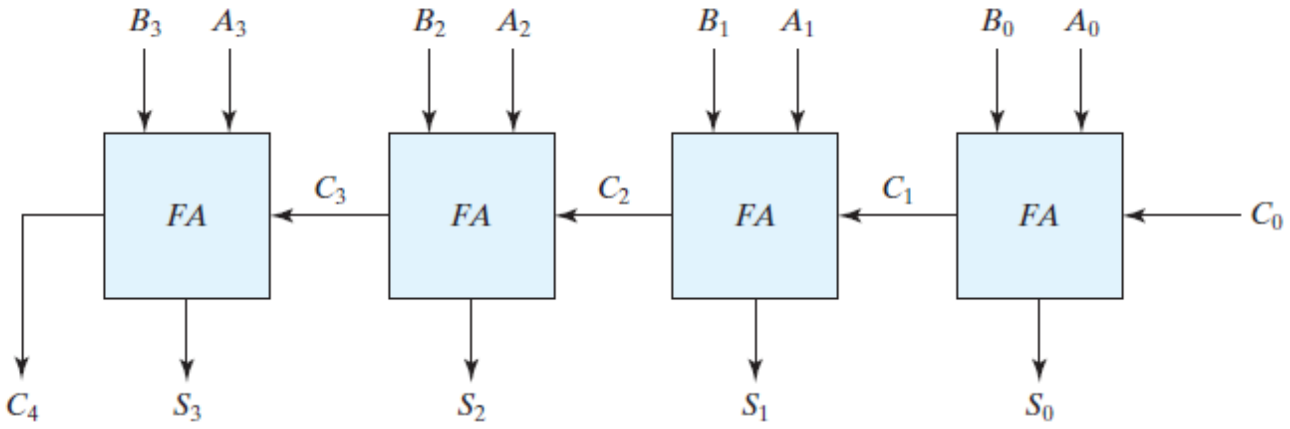



```
module full_adder (output S, C, input x, y, z);
  wire S1, C1, C2;

// Instantiate half adders
  half_adder HA1 (S1, C1, x, y);
  half_adder HA2 (S, C2, S1, z);
  or G1 (C, C2, C1);
endmodule
```

جمع کننده

جمع کننده چهار بیتی




```
module ripple_carry_4_bit_adder ( output [3: 0] Sum, output C4,
  input [3: 0] A, B, input C0);
  wire          C1, C2, C3;          // Intermediate carries
// Instantiate chain of full adders
full_adder      FA0 (Sum[0], C1, A[0], B[0], C0),
                FA1 (Sum[1], C2, A[1], B[1], C1),
                FA2 (Sum[2], C3, A[2], B[2], C2),
                FA3 (Sum[3], C4, A[3], B[3], C3);

endmodule
```

برنامه نویسی سلسله مراتبی

جمع کننده چهار بیتی

```
module half_adder (output S, C, input x, y);
// Instantiate primitive gates
  xor (S, x, y);
  and (C, x, y);
endmodule

module full_adder (output S, C, input x, y, z);
  wire S1, C1, C2;

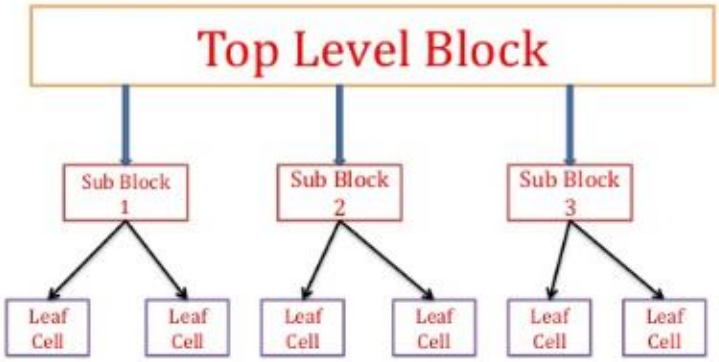
// Instantiate half adders
  half_adder HA1 (S1, C1, x, y);
  half_adder HA2 (S, C2, S1, z);
  or G1 (C, C2, C1);
endmodule

module ripple_carry_4_bit_adder ( output [3: 0] Sum, output C4,
  input [3: 0] A, B, input C0);
  wire          C1, C2, C3;          // Intermediate carries
// Instantiate chain of full adders
full_adder      FA0 (Sum[0], C1, A[0], B[0], C0),
                FA1 (Sum[1], C2, A[1], B[1], C1),
                FA2 (Sum[2], C3, A[2], B[2], C2),
                FA3 (Sum[3], C4, A[3], B[3], C3);
endmodule
```

برنامه نویسی سلسله مراتبی

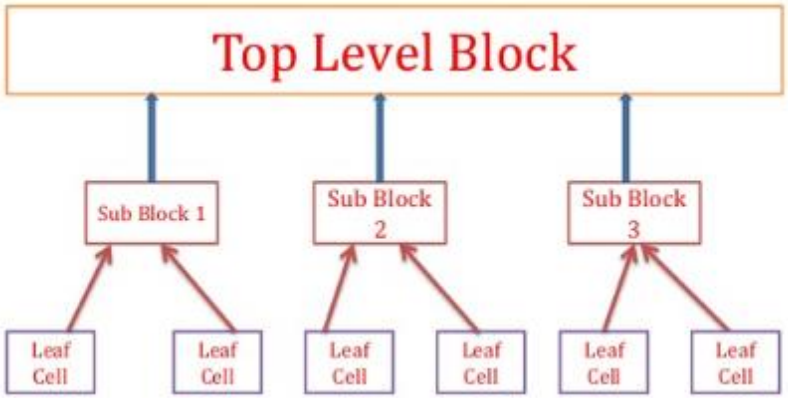
- طراحی بالا به پایین

در ابتدا بلوک های بزرگتر تعریف می شود و سپس زیر بلوک ها تا رسیدن به کوچکترین واحد ادامه پیدا می کند.



- طراحی پایین به بالا

تعریف از کوچکترین واحد ممکن صورت می گیرد و با توجه به آنها بلوک های بزرگتر ساخته می شود.



برنامه نویسی سلسله مراتبی

ماژول ها می توانند داخل ماژول های دیگر نمونه دهی شوند ولی یک ماژول نمی تواند داخل ماژول دیگری تعریف شود.

هر ماژول بایستی به طور مستقل تعریف شود، به بیان دیگر تعریف هیچ ماژولی نمی تواند بین کلمات begin/end ماژول دیگر قرار گیرد.

Behavioral Description

توصیف رفتاری Behavioral Description

این مدلسازی مدارات دیجیتال را در یک سطح کارکردی و الگوریتمی نمایش می دهد و بیشتر برای توصیف مدارات ترتیبی استفاده می شود.

در این مدلسازی از کلمه کلیدی always برای کنترل رویداد ها استفاده می شود.

خروجی هدف در این مدلسازی باید از نوع داده reg باشد.

* نوع داده reg تا زمانی که مقدار جدیدی به آن تخصیص داده نشود مقدار قبلی خود را حفظ می کند، برخلاف نوع داده wire که خروجی ها را به طور پیوسته بهنگام سازی میکند.

always @

فهرست حساسیت: یک عبارت کنترل رویداد با منطق ترکیبی واکنشی است.

هر زمان که تغییری در هر یک از متغییرهای فهرست شده بعد از نماد @ وجود داشته باشد بلوک مربوط به دستور always اجرا می شود.

مثال:

```
always      @(A or B or select)
always @ (in_0, in_1, in_2, in_3, select)
always @(posedge clock, negedge reset)
```

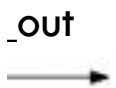
توصیف رفتاری

مالتی پلکسر دو-به-یک

```
// Behavioral description of two-to-one-line multiplexer

module mux_2x1_beh (m_out, A, B, select);
  output      m_out;
  input       A, B, select;
  reg         m_out;

  always      @(A or B or select)
    if (select == 1) m_out = A;
    else m_out = B;
endmodule
```



معادل دستور زیر با استفاده از assign

```
assign m_out = (select)? A : B;
```

استفاده از CASE

```
// Behavioral description of four-to-one line multiplexer

// Verilog 2001, 2005 port syntax

module mux_4x1_beh
( output reg m_out,
  input      in_0, in_1, in_2, in_3,
  input [1: 0] select
);
always @ (in_0, in_1, in_2, in_3, select)      // Verilog 2001, 2005 syntax
  case (select)
    2'b00:          m_out = in_0;
    2'b01:          m_out = in_1;
    2'b10:          m_out = in_2;
    2'b11:          m_out = in_3;
  endcase
endmodule
```

*درغیاب یک انطباق هیچ گزاره ای اجرا نمی شود.

استفاده از CASE

case با استفاده از default

به طراح اجازه میدهد تا تمام حالت های استفاده نشده را به یک حالت مطلوب ،بدون شرح جزئیات، نگاشت کند.

```
module mux (a,b,c,d,sel,y);
input a, b, c, d;
input [1:0] sel;
output y;

reg y;

always @ (a or b or c or d or sel)
case (sel)
  0 : y = a;
  1 : y = b;
  2 : y = c;
  3 : y = d;
  default : $display("Error in SEL");
endcase

endmodule
```

استفاده از CASE

case بدون استفاده از default

```
module mux_without_default (a,b,c,d,sel,y);
input a, b, c, d;
input [1:0] sel;
output y;

reg y;

always @ (a or b or c or d or sel)
case (sel)
  0 : y = a;
  1 : y = b;
  2 : y = c;
  3 : y = d;
  2'bxx,2'bx0,2'bx1,2'b0x,2'b1x,
  2'bzz,2'bz0,2'bz1,2'b0z,2'b1z : $display("Error in SEL");
endcase

endmodule
```

استفاده از CASE

CASEZ: که حالت Z را DON’T CARE در نظر میگیرد.

CASEX: که حالت X و Z را DON’T CARE در نظر میگیرد.

```
always @ (opcode or a or b or c)
casez(opcode)
  4'b1zzx : begin // Don't care about lower 2:1 bit, bit 0 match with x
              out = a;
              $display("@%0dns 4'b1zzx is selected, opcode %b",$time,opcode);
            end
  4'b01?? : begin
              out = b; // bit 1:0 is don't care
              $display("@%0dns 4'b01?? is selected, opcode %b",$time,opcode);
            end
  4'b001? : begin  // bit 0 is don't care
              out = c;
              $display("@%0dns 4'b001? is selected, opcode %b",$time,opcode);
            end
  default : begin
              $display("@%0dns default is selected, opcode %b",$time,opcode);
            end
endcase
```

استفاده از CASE

مقایسه خروجی های CASE

```
always @ (sel)
case (sel)
  1'b0 : $display("Normal : Logic 0 on sel");
  1'b1 : $display("Normal : Logic 1 on sel");
  1'bx : $display("Normal : Logic x on sel");
  1'bz : $display("Normal : Logic z on sel");
endcase

always @ (sel)
casex (sel)
  1'b0 : $display("CASEX  : Logic 0 on sel");
  1'b1 : $display("CASEX  : Logic 1 on sel");
  1'bx : $display("CASEX  : Logic x on sel");
  1'bz : $display("CASEX  : Logic z on sel");
endcase

always @ (sel)
casez (sel)
  1'b0 : $display("CASEZ  : Logic 0 on sel");
  1'b1 : $display("CASEZ  : Logic 1 on sel");
  1'bx : $display("CASEZ  : Logic x on sel");
  1'bz : $display("CASEZ  : Logic z on sel");
endcase
```

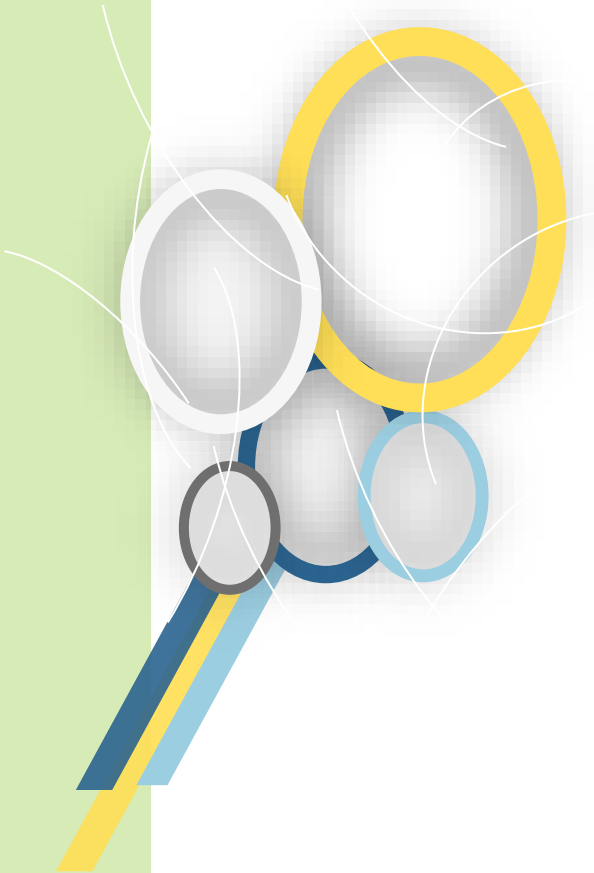
خروجی

```
    Driving 0
Normal : Logic 0 on sel
CASEX  : Logic 0 on sel
CASEZ  : Logic 0 on sel

    Driving 1
Normal : Logic 1 on sel
CASEX  : Logic 1 on sel
CASEZ  : Logic 1 on sel

    Driving x
Normal : Logic x on sel
CASEX  : Logic 0 on sel
CASEZ  : Logic x on sel

    Driving z
Normal : Logic z on sel
CASEX  : Logic 0 on sel
CASEZ  : Logic 0 on sel
```



خسته نباشید