طراحی خودکار مدارهای دیجیتال
جلسه ۳– آشنایی با Verilog (بخش ۲)

روح الله دیانت

دانشگاه قم- دانشکده فنی- گروه مهندسی کامپیوتر

خلاصه مطالب قبلی

- آشنایی با دستورات ساده Verilog: پیاده‌سازی گیت‌های and، or، not، nor، nand، xor ، bufif1…
- پیاده‌سازی مدارهای ساده ترکیبی: مدار شامل and و or، mux21 و mux41
- پیاده‌سازی سلسله‌مراتبی و امکان استفاده از ماژول‌های طراحی شده قبلی در پیاده‌سازی ماژول‌های دیگر.
- آشنایی با مفاهیم شبیه‌سازی، سنتز و برنامه‌ریزی FPGA . انجام شبیه‌سازی با Modelsim، سنتز و برنامه‌ریزی با ISE

فهرست مطالب این جلسه

- ماژول‌های اولیه (Primitive) و غیر اولیه (Nonprimitive).
- هم‌زمانی (Concurrency) در Verilog.
- تعریف آرایه در Verilog .
- طراحی جمع‌کننده در Verilog.

ماژول‌های اولیه (Primitive) و غیر اولیه (Nonprimitive).

- در برنامه پیاده‌سازی mux41، ماژول mux21 احضار شده است.

```
module mux21 (O,I0,I1,S);
input I0,I1,S;
output O;
wire w1,w2,w3;
not n1(w1,S);
and a1(w2,w1,I0);
and a2(w3,I1,S);
or o1(O,w2,w3);
endmodule
```

- دستور and نیز در واقع یک دستور احضار ماژول است. با این تفاوت که ماژول and در خود Verilog تعریف شده است اما mux21 را برنامه‌نویس، نوشته است.

```
module mux41 (O,I0,I1,I2,I3,S0,S1);
input I0,I1,I2,I3,S0,S1;
output O;
mux21 m0 (W1,I0,I1,S0);
mux21 m1 (W2,I2,I3,S0);
mux21 m2 (O,W1,W2,S1);
endmodule
```

- به ماژول‌های تعریف شده در Verilog (مانند and)، اصطلاحا ماژول اولیه گفته می‌شود.
- به ماژول‌های نوشته شده توسط برنامه‌نویس، اصطلاحا یک ماژول غیر اولیه اطلاق می‌گردد.

همزمانی (Concurrency) در Verilog

- **نکته ۱:** یک مدار ترکیبی مانند mux21 یا mux41 را در نظر بگیرید. در یک سخت‌افزار واقعی، اگر تغییری حتی در یک نقطه از مدار رخ دهد، می تواند در نقاط دیگر مدار تأثیر بگذارد. به بیان دیگر، اگر در نقطه‌ای از مدار، تغییر مقدار رخ دهد، مقادیر همه نقاط مدار می‌بایست مورد ارزیابی مجدد قرار گیرند. به این مفهوم، اصطلاحا همزمانی در سخت‌افزار گفته می‌شود.
- در یک زبان توصیف سخت‌افزار این قابلیت می‌بایست وجود داشته باشد.
- برخلاف یک برنامه نرم‌افزاری که در آن، دستورات به صورت ترتیبی اجرا می‌شوند، در یک زبان توصیف سخت‌افزاری مانند Verilog، در هنگام شبیه‌سازی، با هر تغییر مقدار، همه دستورات ماژول، مورد ارزیابی مجدد قرار می‌گیرند.

هم‌زمانی (Concurrency) در Verilog (ادامه)

❖ نکته ۲: ابزار سنتز در مواجهه با یک دستور احضار ماژول، آن قطعه را به مدار اضافه می‌کند. همچنین، هنگام رسیدن به دستورات تعریف `input`، `output` و `wire` ، پورت‌های ورودی و خروجی و سیم‌ها را به مدار می‌افزاید.

❖نتیجه: در `Verilog`، تقدّم و تأخر نوشتن دستورات اهمیتی ندارد. برای سنتز، هر دستور یک سخت‌افزار را مدل می‌کند و به مدار اضافه می‌کند.

تعریف آرایه در Verilog

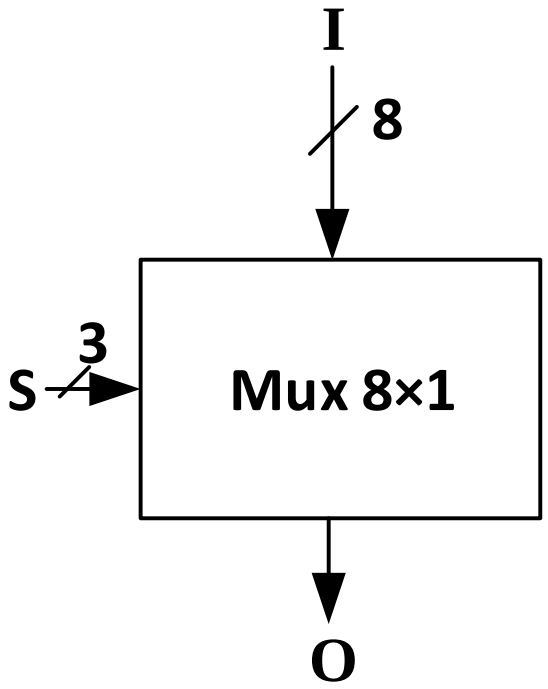
❖**مثال ۱:** فرض کنید هدف، طراحی یک mux 8*1 باشد. این ماژول، دارای ۸ ورودی اصلی و ۳ ورودی انتخاب است. خط اول این ماژول به صورت زیر می‌تواند باشد.

```
module mux81(O,I0,I1,I2,I3,I4,I5,I6,I7,S0,S1,S2);
```

❖ اگر ابعاد mux بیشتر شود، تعداد خطوط بیشتر می‌شوند. نوشتن برنامه به این صورت طولانی چندان جالب نیست.

❖ ارائه مفهوم آرایه در Verilog این مشکل را مرتفع می‌سازد.

تعریف آرایه در Verilog



- برای درک بهتر نحوه استفاده از آرایه، ماژول را به صورت روبرو رسم نموده‌ایم. در اینجا I و S دو خط آرایه‌ای به ترتیب با اندازه ۸ و ۳ می‌باشند.
- در برنامه صفحه بعد با استفاده از تعریف آرایه، پیاده‌سازی ماژول mux 8*1 انجام شده است.
- برای سادگی بیشتر برنامه نویسی، پیاده سازی mux 8*1 را با کمک mux 4*1 و mux 2*1 انجام داد.

تعریف آرایه در Verilog

```
module mux21 (O,I0,I1,S);
input I0,I1,S;
output O;
wire w1,w2,w3;
not n1(w1,S);
and a1(w2,w1,I0);
and a2(w3,I1,S);
or o1(O,w2,w3);
endmodule
module mux41 (O,I0,I1,I2,I3,S0,S1);
input I0,I1,I2,I3,S0,S1;
output O;
mux21 m0 (W1,I0,I1,S0);
mux21 m1 (W2,I2,I3,S0);
mux21 m2 (O,W1,W2,S1);
endmodule
```

```
module mux81(O,I,S);
input [7:0] I;
input [2:0] S;
output O;
wire W1,W2;
mux41 m0(W1,I[0],I[1],I[2],I[3],S[0],S[1]);
mux41 m1(W2,I[4],I[5],I[6],I[7],S[0],S[1]);
mux21 m2(O,W1,W2,S[2]);
endmodule
```

- دستور `input [7:0] I` پورت ورودی I را به صورت یک آرایه ۸ بیتی تعریف می‌کند.
- به طور مشابه، دستور `input [2:0]`، پورت ورودی S را به صورت یک آرایه ۳ عنصری تعریف می‌کند.

تعریف آرایه در Verilog (ادامه)

- امکان تعریف به صورتی مثل `input [0:7] I` نیز وجود دارد. در این وضعیت، سمت راست‌ترین بیت، بیت شماره ۷ خط `I` است.
- به طور کلی تعریف آرایه با هر محدوده دلخواه `a:b` به صورت افزایشی (مانند مثال بالا) یا کاهشی (مانند مثال صفحه قبل) وجود دارد.
- در اغلب موارد، این دو روش با هم معادل هستند. البته در جلسات بعد، مثال‌هایی را مشاهده خواهیم کرد که در آنها بین این دو نحوه تعریف آرایه تفاوت وجود دارد.
- با استفاده از اندیس، می‌توان به هر عنصر آرایه دسترسی داشت. مثلا `I[2]` .

تعریف آرایه در Verilog (ادامه)

❖**مثال ۲:** در این مثال، برنامه‌ای مشابه مثال ۱ نوشته شده است؛ با این تفاوت که در اینجا، علاوه بر mux 8*1، در پیاده‌سازی mux 4*1 نیز از مفهوم آرایه استفاده شده است.

تعریف آرایه در Verilog (ادامه)

- در این برنامه، در پیاده‌سازی mux 4*1 نیز از آرایه استفاده شده است.
- در Verilog امکان مشخص کردن محدوده‌ای از آرایه وجود دارد. I[3:0] در دستور احضار ماژول mux41، بیت های 0 تا 3 از خط I را مشخص می‌کند.

```
module mux81(O,I,S);
input [7:0] I;
input [2:0] S;
output O;
wire W1,W2;
mux41 m0(W1,I[3:0],S[1:0]);
mux41 m1(W2,I[7:4],S[1:0]);
mux21 m2(O,W1,W2,S[2]);
endmodule
```

```
module mux41 (O,I,S);
input [3:0] I;
input [1:0] S;
output O;
mux21 m0 (W1,I[0],I[1],S[0]);
mux21 m1 (W2,I[2],I[3],S[0]);
mux21 m2 (O,W1,W2,S[1]);
endmodule
```

```
module mux21 (O,I0,I1,S);
input I0,I1,S;
output O;
wire w1,w2,w3;
not n1(w1,S);
and a1(w2,w1,I0);
and a2(w3,I1,S);
or o1(O,w2,w3);
Endmodule
```

تعریف آرایه در Verilog (ادامه)

❖ **نکته:** زمانی که یک آرایه به صورت افزایشی (کاهشی) تعریف شده باشد دسترسی محدوده‌ای به آن هم باید افزایشی (کاهشی) باشد. مثلا در ماژول `mux81` که `I` به صورت کاهشی تعریف شده است، دسترسی محدوده‌ای هم باید به صورت کاهشی انجام شود. مثلا نوشتن `I[0:3]` در اینجا درست نیست.

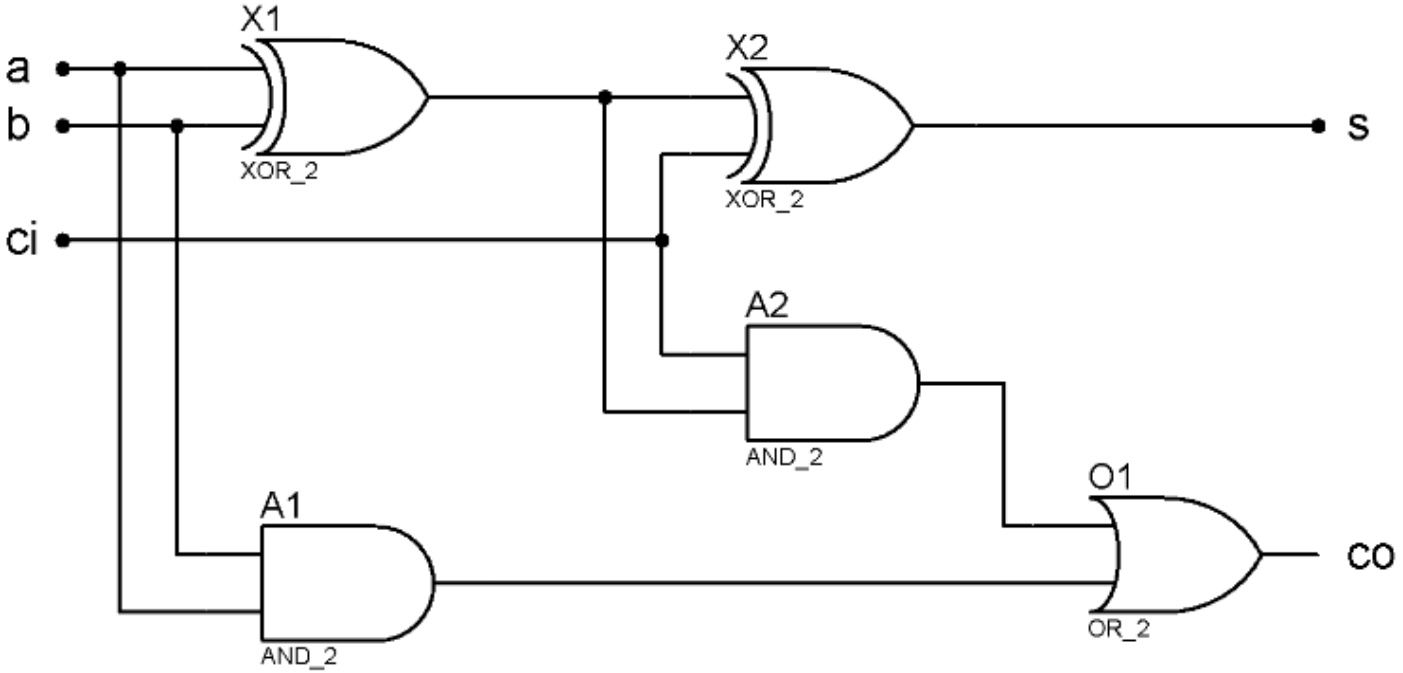
پیاده‌سازی جمع کننده موازی (PA)

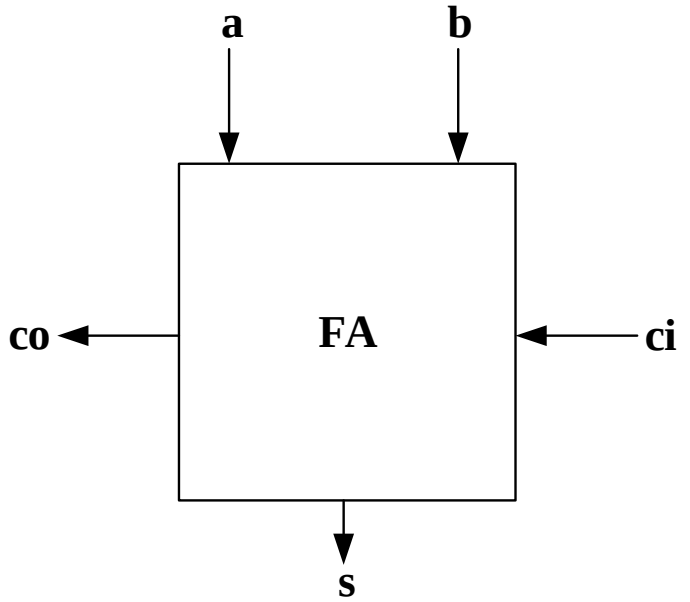
❖ **مثال ۳:** در این قسمت، یک جمع‌کننده موازی (Parallel adder) ۳۲ بیتی طراحی می‌شود.

در قالب این مثال، بسیاری از نکاتی که در این جلسه و جلسه قبل بیان شد، مرور خواهد شد. همچنین با مفهوم احضار آرایه‌ای از ماژول‌ها، آشنا می‌شویم.

پیاده‌سازی جمع کننده موازی (PA) (ادامه)

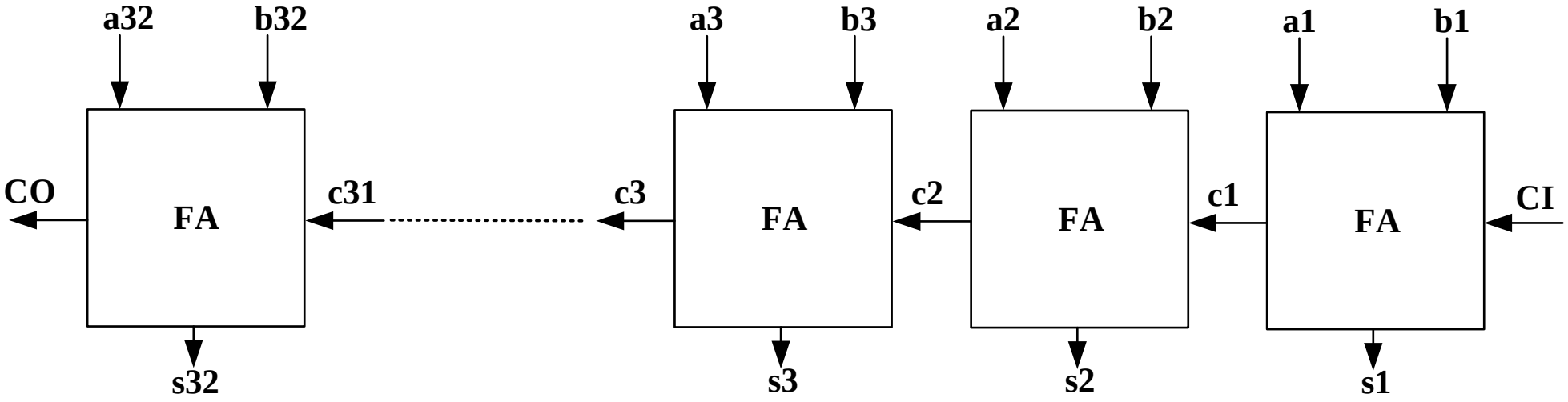
❖ **یادآوری از درس مدار منطقی:** یک جمع‌کننده موازی n بیتی از کنار هم قراردادن n تا تمام‌جمع‌کننده (Full adder) ساخته می‌شود. ساختار یک FAدر شکل زیر نشان داده شده است.





پیاده‌سازی جمع کننده موازی (PA) (ادامه)

- با کنار هم قراردادن ۳۲ عدد FA به صورت زیر، PA ساخته می‌شود.



- با توجه به مطالب بیان شده در جلسه قبل، برای طراحی PA در Verilog، ابتدا

پیاده‌سازی جمع کننده موازی (PA) (ادامه)

- با توجه به مطالب بیان شده در جلسه قبل، برای طراحی PA در Verilog، ابتدا باید ماژول FA را طراحی نمود و سپس، با توجه به شکل صفحه قبل و با ۳۲ بار احضار ماژول FA، PA را ساخت.

- واضح است ۳۲ بار نوشتن دستور احضار ماژول، چندان مناسب نیست. برای حل این نقص، در Verilog امکان تعریف آرایه‌ای از ماژول‌ها، ارائه شده است.

- در برنامه صفحه بعد از همین تکنیک استفاده شده است.

پیاده‌سازی جمع کننده موازی (PA) (ادامه)

```
module FA (s,co,a,b,ci);
input a,b,ci;
output s,co;
wire w1,w2,w3;
xor x1(w1,a,b);
and a1(w2,a,b);
and a2(w3,w1,ci);
xor x2(s,w1,ci);
or o1(co,w2,w3);
endmodule
```

```
module PA (S,CO,A,B,CI);
input [32:1] A,B;
input CI;
output CO;
output  [32:1] S;
wire [31:1] C;

FA F[32:1] (S[32:1],{CO , C[31:1]},A[32:1],B[32:1],{C[31:1],CI});

endmodule
```

پیاده‌سازی جمع کننده موازی (PA) (ادامه)

- در دستور احضار ماژول FA، عبارت FA F[32:1]، موجب ۳۲ بار احضار ماژول FA می‌شود.
- در ادامه این دستور، باید برای هر یک از آرگومان‌های دستور، ۳۲ پارامتر مشخص کرد. مثلا برای s ، یک آرایه ۳۲ عنصری مشخص شده است.
- آرگومان دوم، نقلی خروجی است که برای FA آخر co بوده و برای بقیه از c[31:1] است. برای کنار هم قراردادن این موارد، در Verilog از { } استفاده می‌شود. {co, c[31:1]} موجب می‌شود یک موجودیت ۳۲ بیتی ایجاد شود که نام سمت چپ‌ترین عنصر آن co است و بقیه عناصر (از سمت چپ) c[31]، c[31]...c[1] می‌باشند.

پیاده‌سازی جمع کننده موازی (PA) (ادامه)

```
module FA (s,co,a,b,ci);
input a,b,ci;
output s,co;
wire w1,w2,w3;
xor x1(w1,a,b);
and a1(w2,a,b);
and a2(w3,w1,ci);
xor x2(s,w1,ci);
or o1(co,w2,w3);
endmodule
```

```
module PA (S,CO,A,B,CI);
input [32:1] A,B;
input CI;
output CO;
output  [32:1] S;
wire [31:1] C;
FA F[32:1] (S[32:1],{CO , C[31:1]},A[32:1],B[32:1],{C[31:1],CI});
endmodule
```

```
module PA_test;
reg [32:1] A,B;
reg CI;
wire CO;
wire  [32:1] S;
PA P0(S,CO,A,B,CI);
initial begin
A=32'd234; B=32'd1000;
CI=0;
#100;
A=32'd2000;
end
endmodule
```

- در اینجا، ماژول تست PA به برنامه اضافه شده است. خطوط A و B به صورت مبنای ۱۰ مقداردهی شده اند. A=32'd234; یعنی A یک مقدار ۳۲ بیتی است و عدد ۲۳۴ ، عددی در مبنای ۱۰ (Decimal) می‌باشد.

پایان