طراحی خودکار مدارهای دیجیتال
جلسه ۲– آشنایی با Verilog (بخش ۱)

روح الله دیانت

دانشگاه قم- دانشکده فنی- گروه مهندسی کامپیوتر

خلاصه مطالب قبلی

- در جلسه قبل با اولین مثال از یک برنامه Verilog آشنا شدیم. در آن مثال، برنامه Verilog پیاده‌سازی کننده یک گیت and به صورت زیر نوشته شد.

```
module and_gate (c,a,b);
input a,b;
output c;
and a0(c,a,b);
endmodule
```

- شبیه‌سازی، سنتز و انتقال این برنامه به FPGA را انجام دادیم.

نکات دیگر در خصوص برنامه پیاده‌کننده گیت and

- علاوه بر دستور and، برای پیاده‌سازی سایر گیت‌ها، دستورات مشابه or، nand، nor، xor و xnor برای پیاده‌سازی کردن بقیه گیت‌ها وجود دارد. همچنین، دستور not برای ساخت یک معکوس‌کننده به صورت زیر مورد استفاده قرار می‌گیرد.

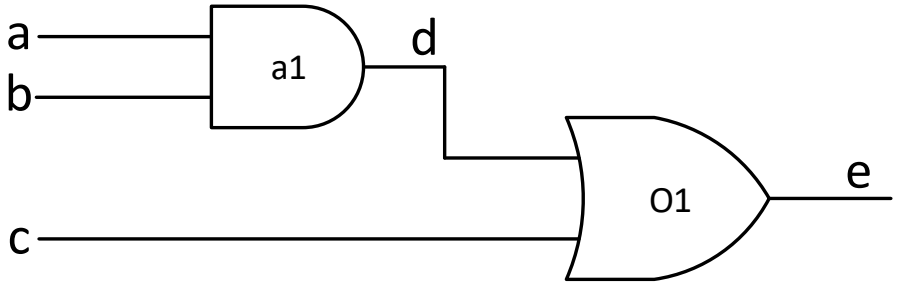
```
not n0 (b,a);
```

- **امکان تعریف گیت‌ها با بیش از ۲ ورودی نیز وجود دارد. مثلا دستور زیر یک and دارای سه ورودی را پیاده‌سازی می‌کند.**

```
and a0 (d,a,b,c);
```

برنامه شماره ۲ (آشنایی با نوع wire)

- برنامه ۲: برنامه زیر مدار شکل روبرو را پیاده‌سازی می‌کند.




```
module and_or_circ (e,a,b,c);
input a,b,c;
output e;
wire d;
and a1 (d,a,b);
or o1(e,c,d);
endmodule
```

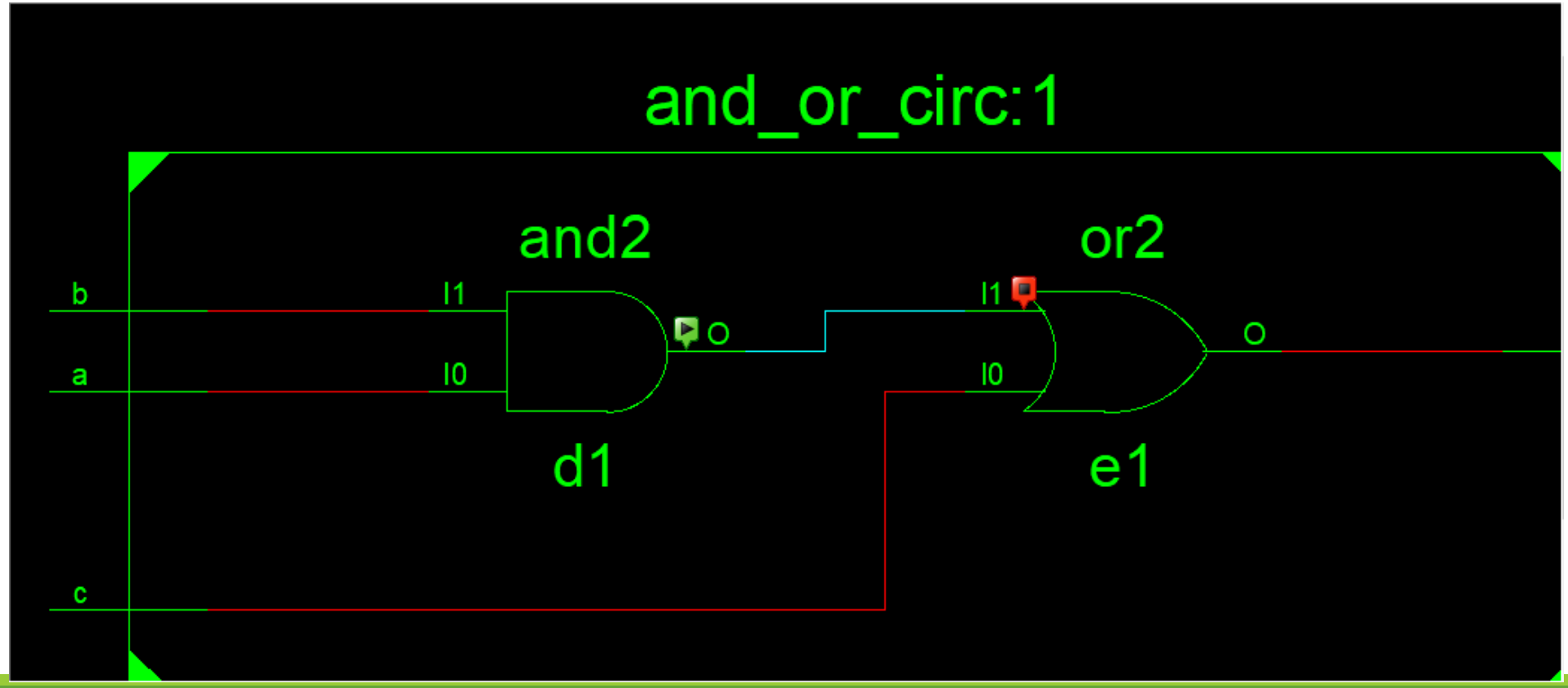
برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

- نکات:
  - d یک خط داخلی مدار می‌باشد و ورودی یا خروجی ماژول نیست. لذا نباید از نوع input و یا output تعریف گردد.
  - نوع wire (سیم) در verilog برای تعریف این نوع خطوط استفاده می‌شود.
  - خروجی and، یک ورودی or می‌باشد. با نظر گرفتن نام یکسان برای هر دو (d)، این اتصال و یکسان بودن را نشان می‌دهیم.

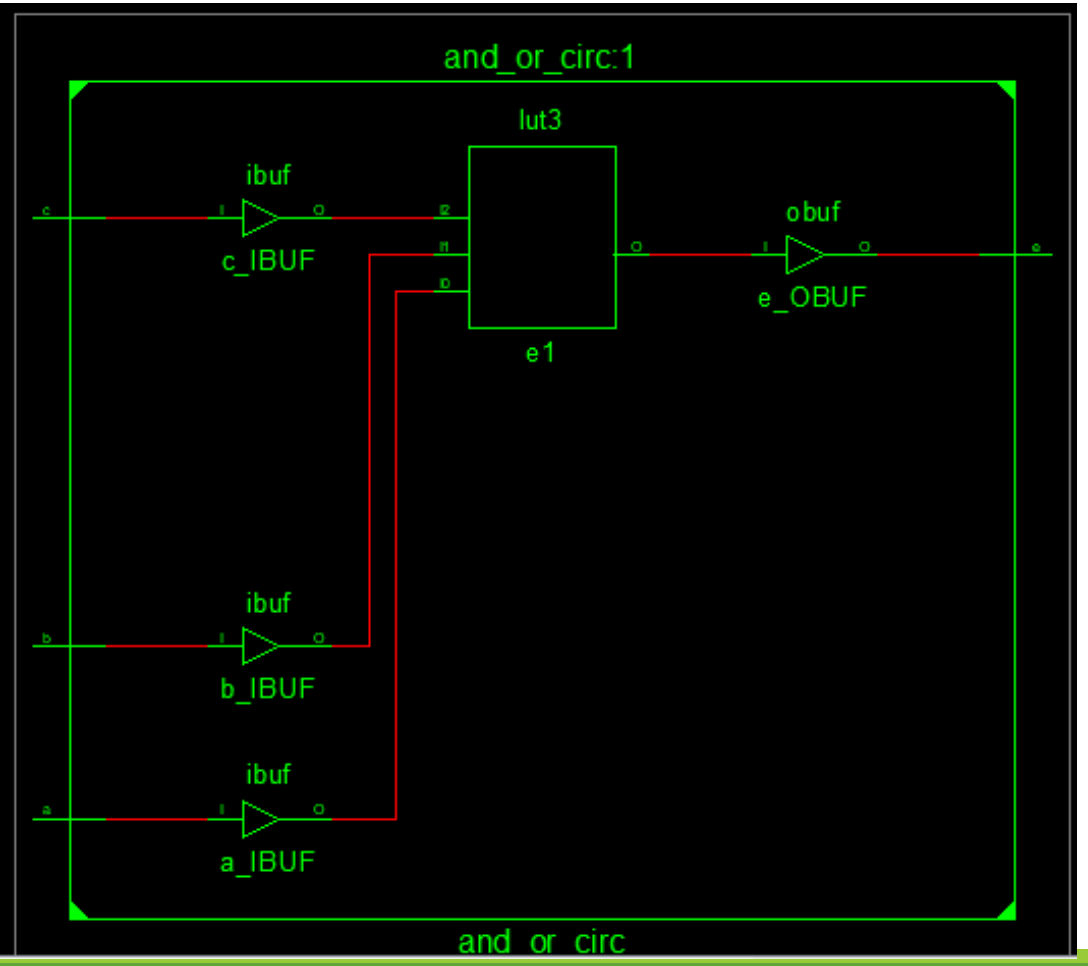
```
module and_or_circ (e,a,b,c);
input a,b,c;
output e;
wire d;
and a1 (d,a,b);
or o1(e,c,d);
endmodule
```

برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

- نتیجه سنتز با ISE (RTL schematic)



برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)



- نتیجه سنتز با ISE (Technology schematic)

برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

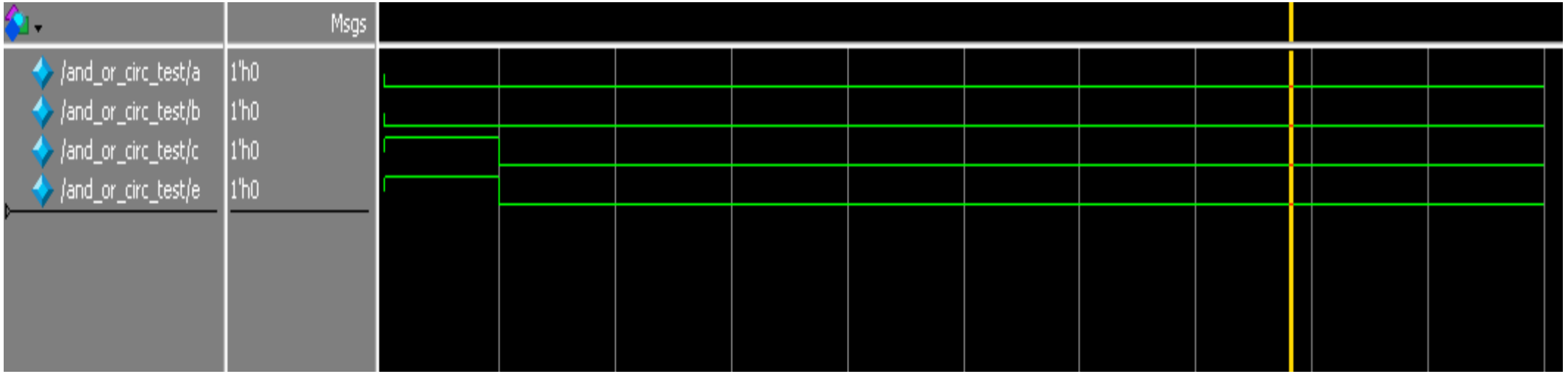
- افزودن Testbench و شبیه‌سازی برنامه با Modelsim

```
module and_or_circ (e,a,b,c);
input a,b,c;
output e;
wire d;
and a1 (d,a,b);
or o1(e,c,d);
endmodule
```

```
module and_or_circ_test;
reg a,b,c;
wire e;
and_or_circ aoc1(e,a,b,c);
initial begin
a=0; b=0; c=1;
#100;
c=0;
end
endmodule
```

برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

- نتیجه شبیه سازی با Modelsim



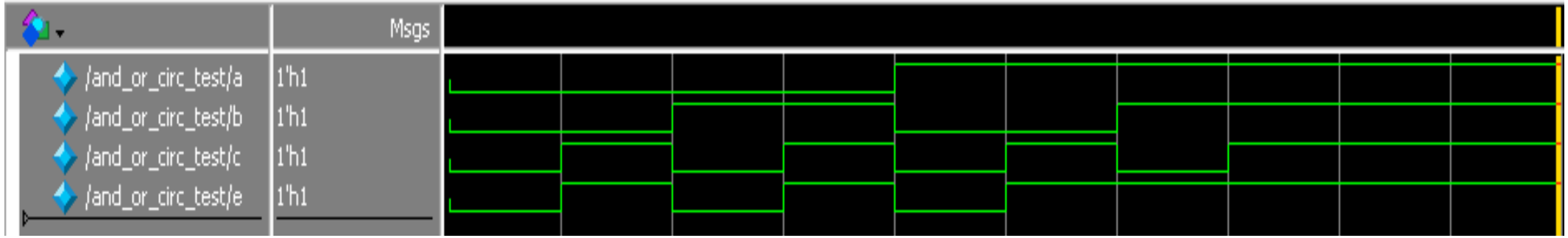
برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

- نکته: TestBench باید جامع نوشته شود. یعنی بتواند همه حالات ممکن ورودی را تا حد امکان در برگیرد تا از صحبت عملکرد سیستم، مطمئن شویم.

```
module and_or_circ_test;
reg a,b,c;
wire e;
and_or_circ aoc1(e,a,b,c);
initial begin
a=0; b=0; c=0; #100;
a=0; b=0; c=1; #100;
a=0; b=1; c=0; #100;
a=0; b=1; c=1; #100;
a=1; b=0; c=0; #100;
a=1; b=0; c=1; #100;
a=1; b=1; c=0; #100;
a=1; b=1; c=1;
end
endmodule
```

برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

- نتیجه شبیه‌سازی مثال قبل با Modelsim



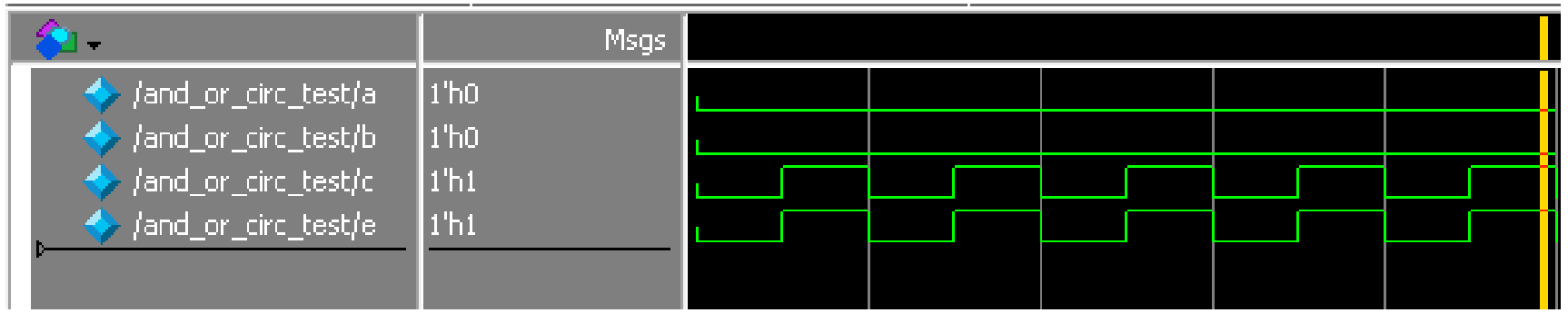
برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

```
module and_or_circ_test;
reg a,b,c;
wire e;
and_or_circ aoc1(e,a,b,c);
initial begin
a=0; b=0; c=0;
end
always
begin
#100;
c=~c;
end
endmodule
```

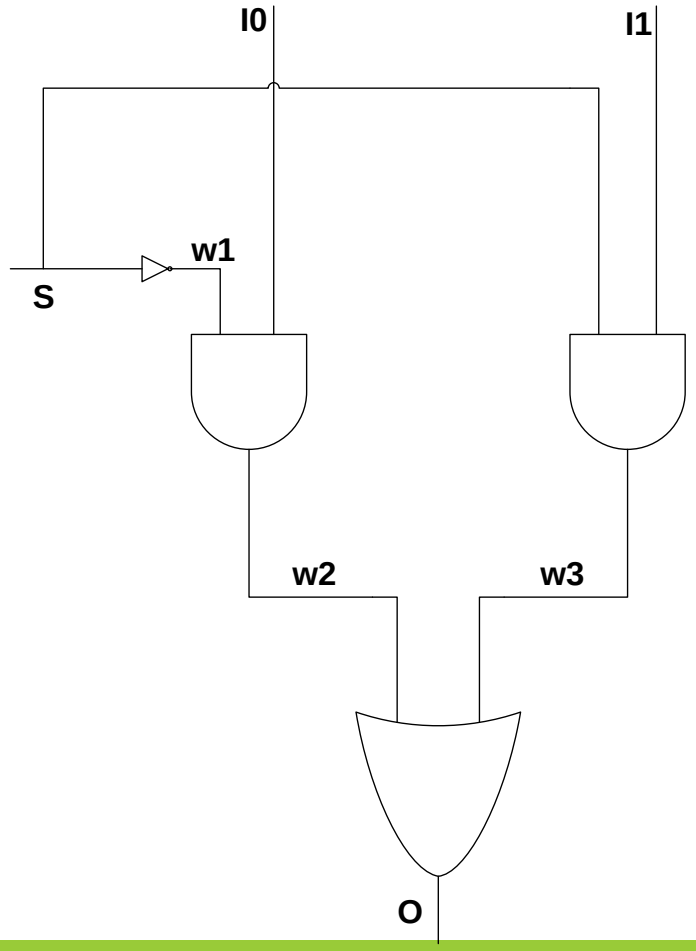
- استفاده از دستور always در Testbench
  - دستور always یک حلقه دائمی را تشکیل می‌دهد.
- در این برنامه مقدار سیگنال c هر ۱۰۰ نانوثانیه، مکمل می‌شود. مقدار اولیه سیگنال‌های a، b و c در لحظه صفر برابر صفر می‌باشد.

برنامه شماره ۲ (آشنایی با نوع wire) (ادامه)

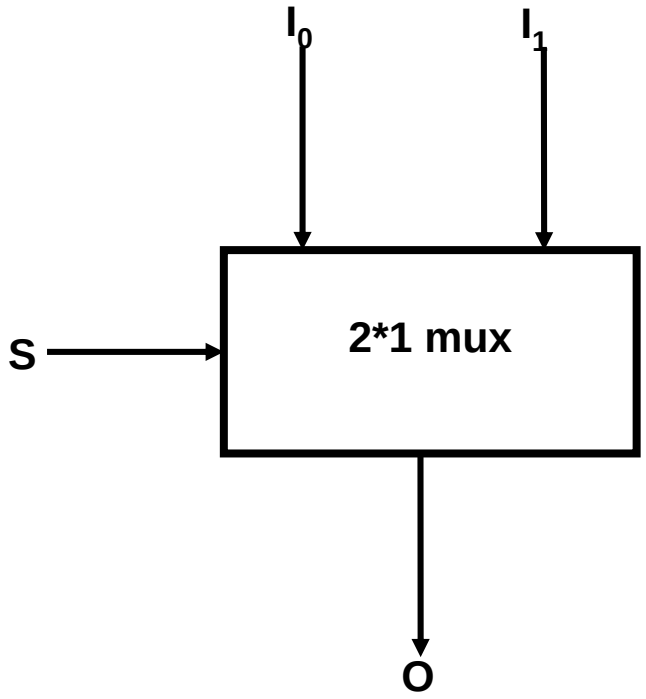
- نتیجه شبیه‌سازی برنامه صفحه قبل



برنامه شماره ۳ (طراحی mux 2*1)

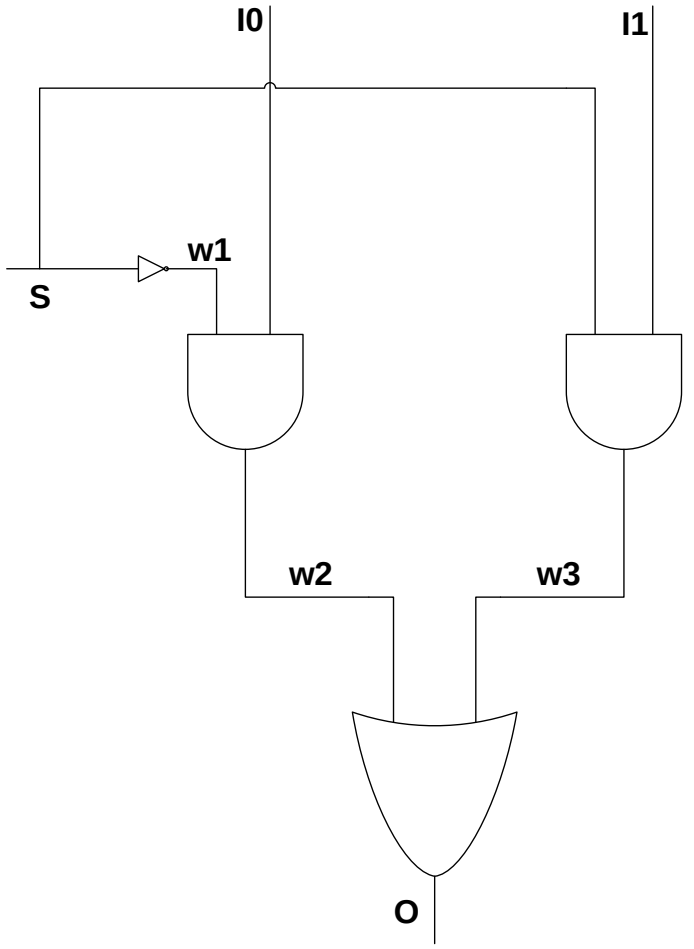


- برنامه ای بنویسید که یک mux 2*1 را پیاده‌سازی کند.

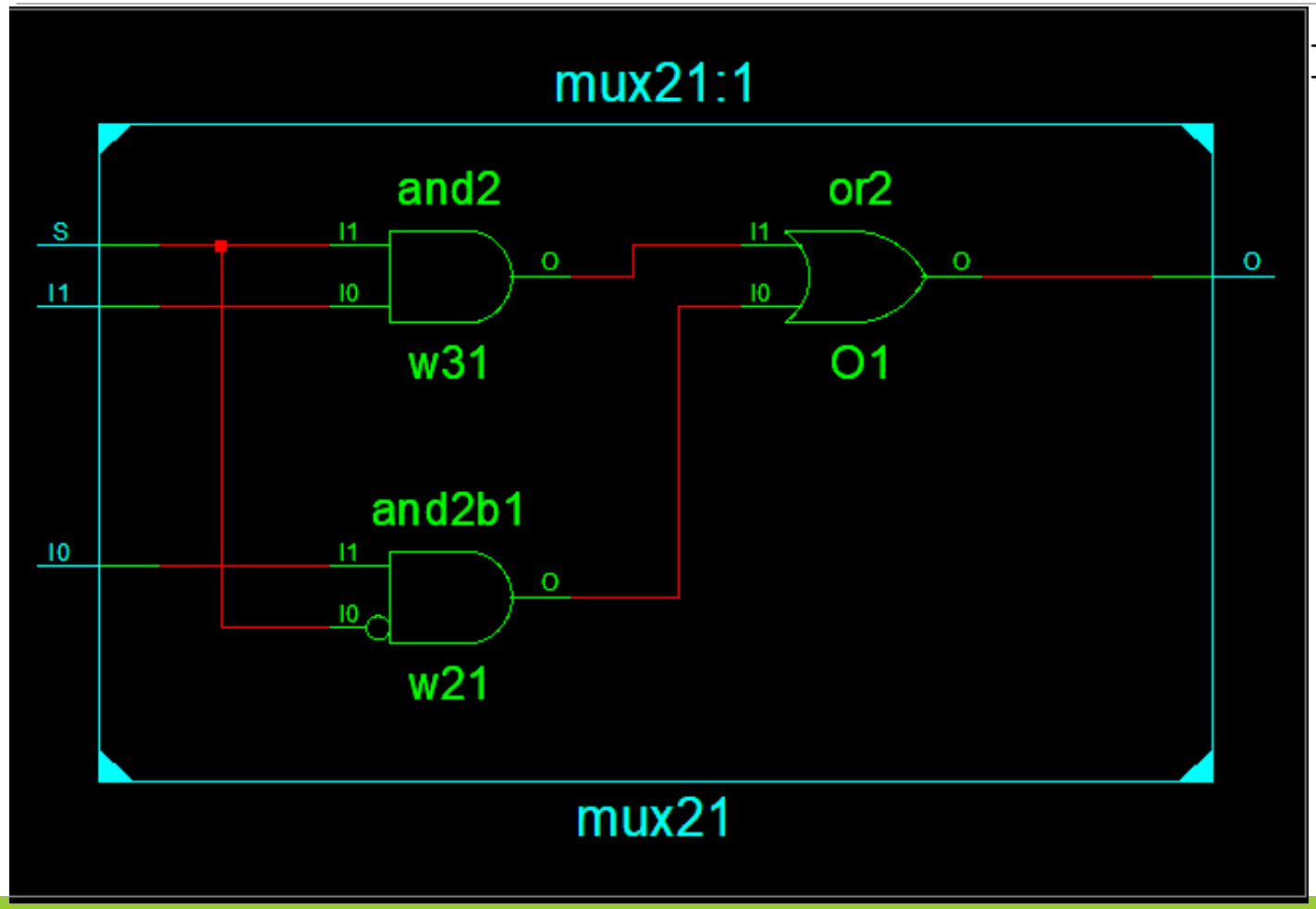


برنامه شماره ۳ (طراحی mux 2*1) (ادامه)

```
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
```

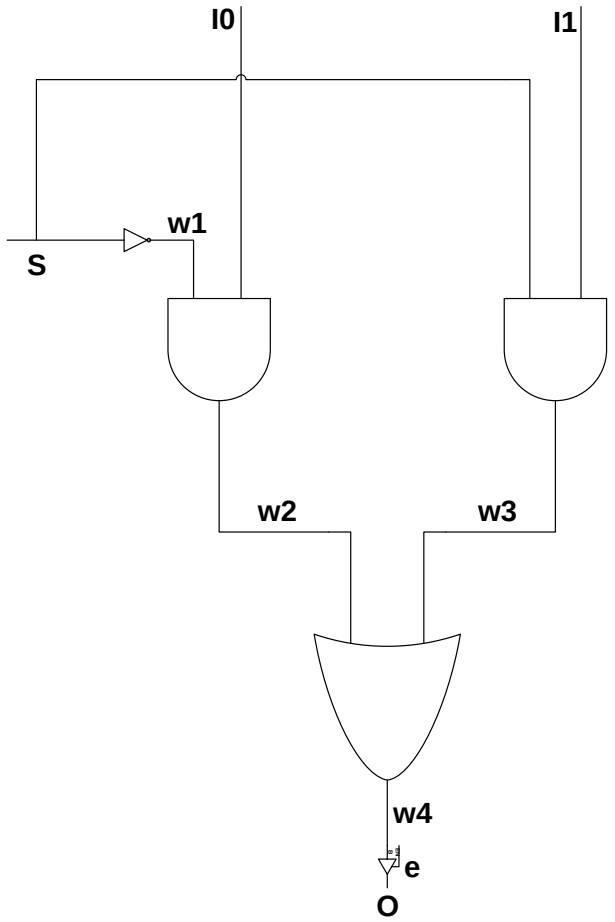


برنامه شماره ۳ (طراحی mux 2*1) (ادامه)



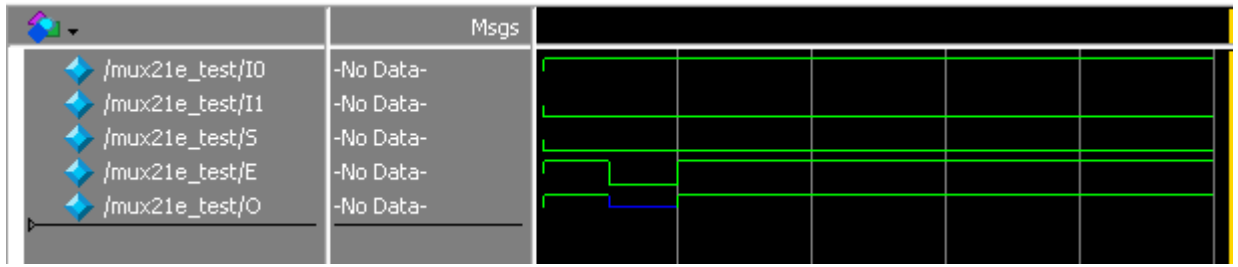
- سنتز برنامه شماره ۳ (RTL shematic)

برنامه شماره ۴ (طراحی mux 2*1 دارای خط enable)



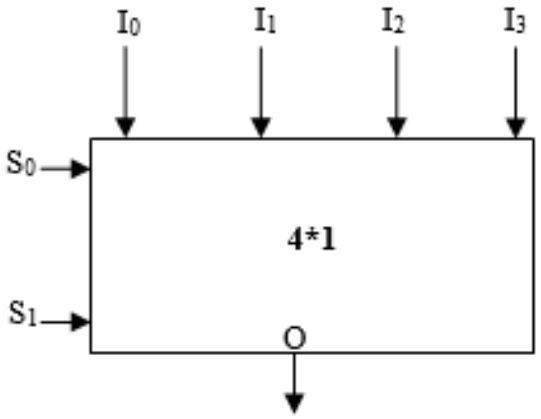
برنامه شماره ۴: مطلوبست طراحی یک mux 2*1 دارای قابلیت enable

```
module mux21e (O,I0,I1,S,E);
input I0,I1,S,E;
output O;
wire w1,w2,w3;
not n1(w1,S);
and a1(w2,w1,I0);
and a2(w3,I1,S);
or o1(w4,w2,w3);
bufif1 bf1(O,w4,E);
endmodule
module mux21e_test;
reg I0,I1,S,E;
wire O;
mux21e mx21e(O,I0,I1,S,E);
initial begin
E=1; I0=1; I1=0; S=0;
#100;E=0;#100;E=1;
end
endmodule
```



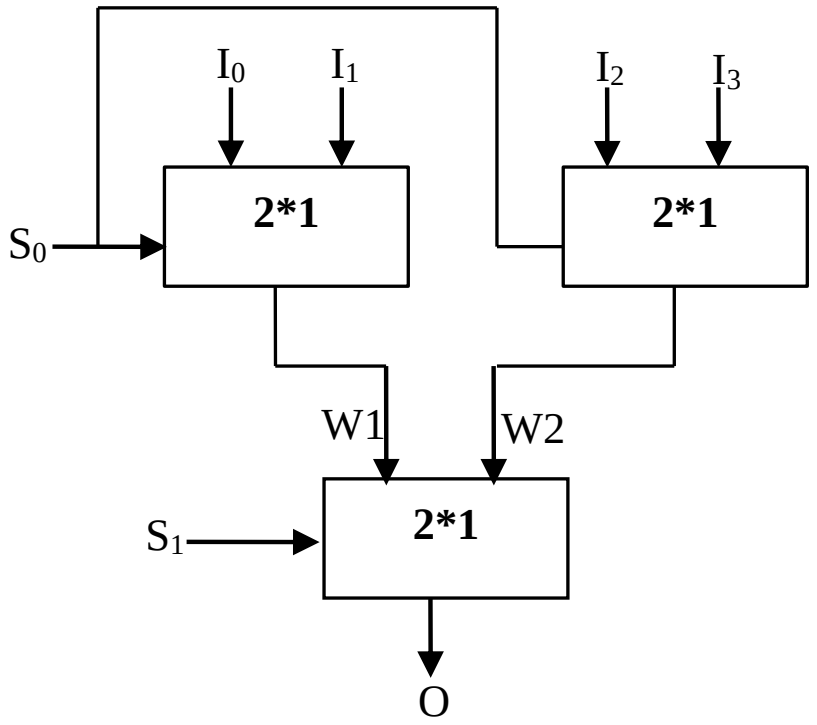
- رنگ آبی در بالا، بیانگر قطع (HZ) بودن O می‌باشد.

برنامه شماره ۵ (طراحی mux 4*1)



- برنامه شماره ۵ (طراحی mux 4*1 معمولی (بدون خط enable))
- mux 4*1 به دو روش قابل طراحی می‌باشد: طراحی مستقیم با گیت‌های and و or.
- پیاده سازی با استفاده از mux 2*1 (شکل صفحه بعد).

برنامه شماره ۵ (طراحی mux 4*1)(ادامه)



- طراحی mux 4*1 با استفاده از چند mux 2*1
- در Verilog امکانی به نام احضار ماژول وجود دارد که با کمک آن می‌توان طراحی ماژول mux 4*1 را با استفاده از mux 2*1 انجام داد.
- در برنامه صفحه بعد، این روش طراحی مورد استفاده قرار گرفته است.

برنامه شماره ۵ (طراحی mux 4*1)(ادامه)

```
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; wire W1,W2;
mux21 m0 (W1,I0,I1,S0);
mux21 m1 (W2,I2,I3,S0);
mux21 m2 (O,W1,W2,S1);
endmodule
```

- همان‌گونه که مشاهده می‌شود برای پیاده‌سازی mux 4*1 با استفاده از mux 2*1، ابتدا ماژول mux21 نوشته شده است. سپس، ماژول mux41 شروع می‌شود. در این ماژول، mux21 سه بار احضار شده است. در هر احضار، کلمه mux21 و به دنبال آن یک نام دلخواه برای ماژول احضار شده ذکر می‌گردد. در ادامه، ورودی‌ها و خروجی‌ها، بیان می‌گردند.

برنامه شماره ۵ (طراحی mux 4*1)(ادامه)

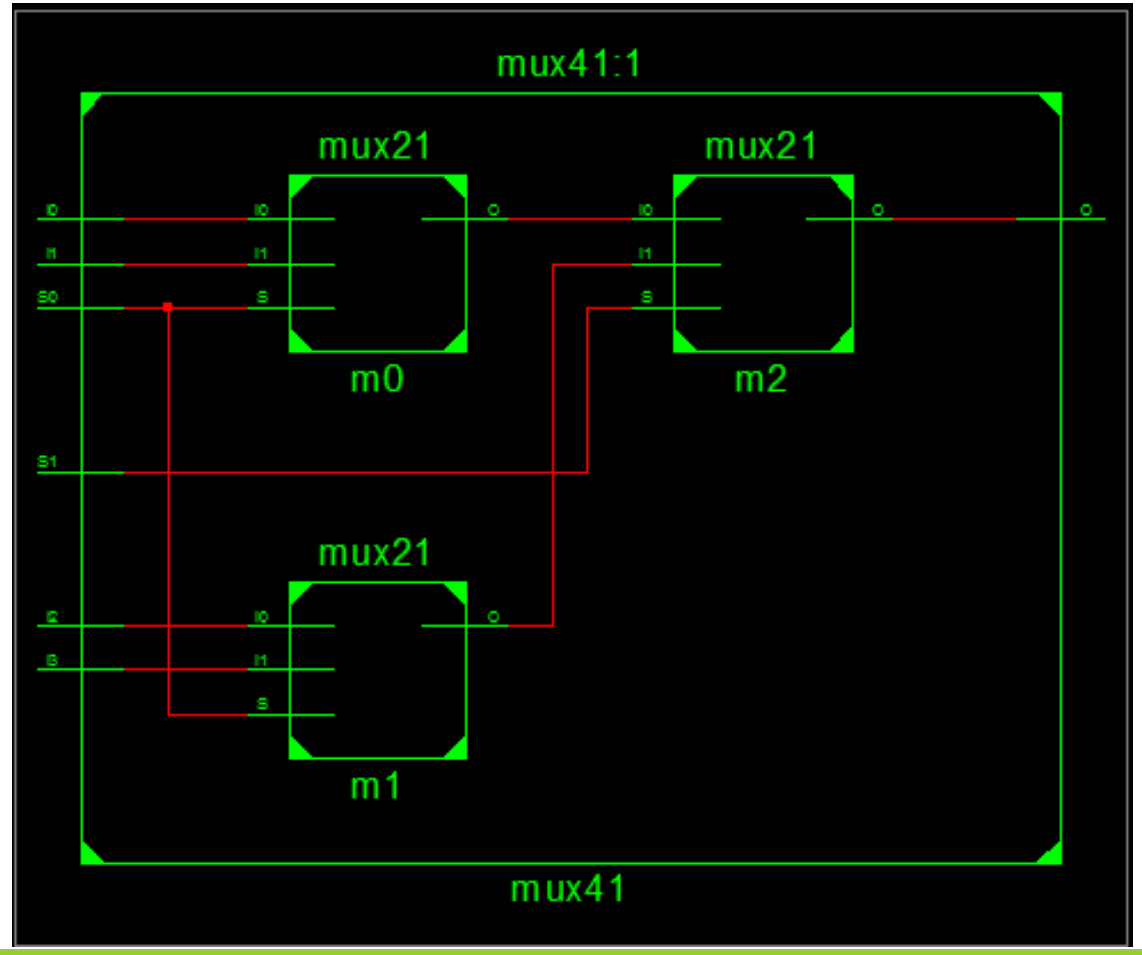
```
module mux21 (O,I0,I1,S);
input I0,I1,S;
output O;
wire w1,w2,w3;
not n1(w1,S);
and a1(w2,w1,I0);
and a2(w3,I1,S);
or o1(O,w2,w3);
endmodule
```

```
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
```

چند نکته:

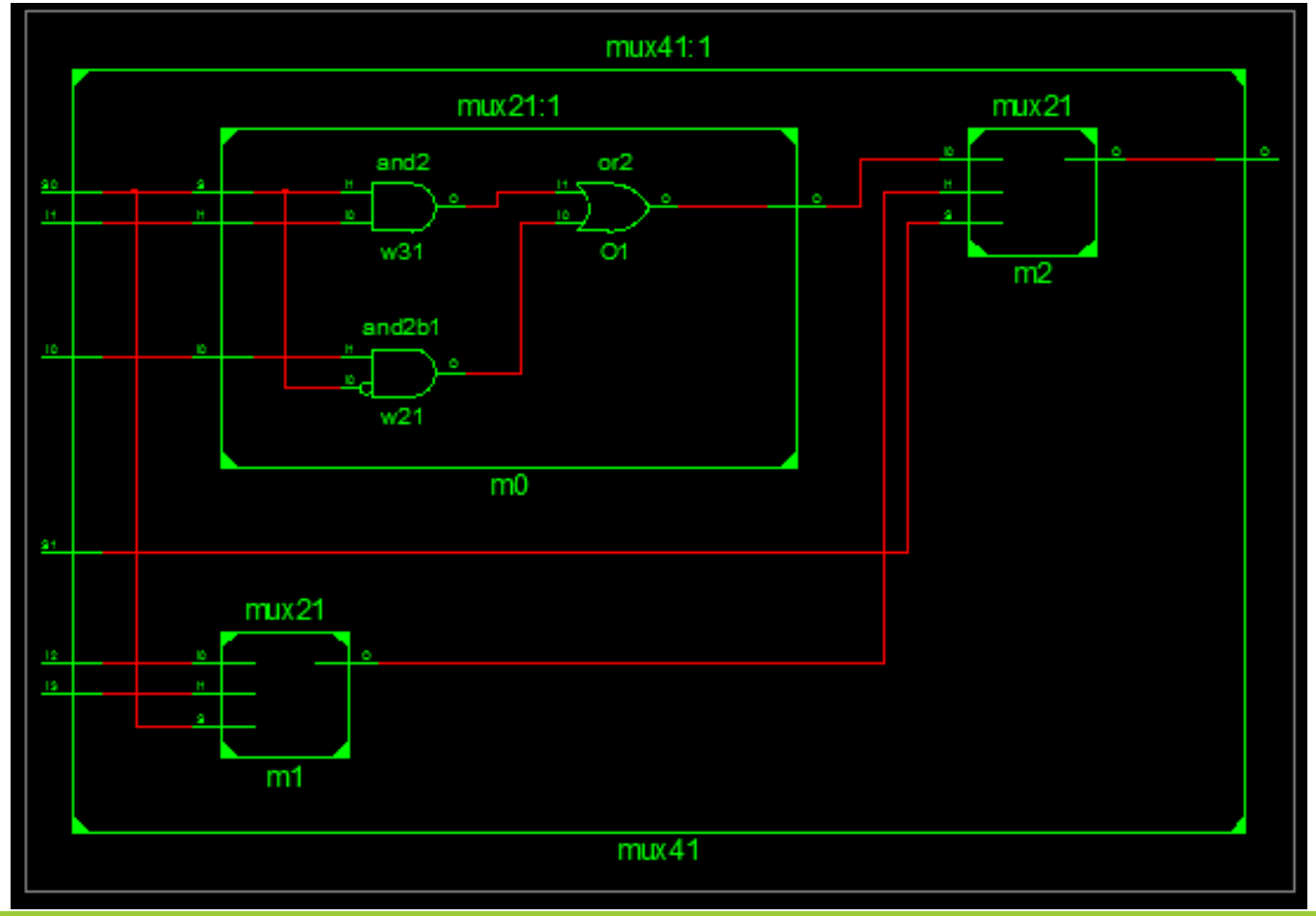
- مشابه یک زبان برنامه‌نویسی نرم‌افزاری مثل C، سیگنال‌های موجود در یک ماژول، محلی (Local) آن ماژول می‌باشند. به عنوان مثال، I0 تعریف شده در mux21 ارتباطی با I0 تعریف شده در mux41 ندارد.
- استفاده از ماژول‌های از قبل طراحی شده، موجب می‌شود برنامه‌نویسی سریع‌تر و به صورت ساده‌تری انجام گیرد. به این روش طراحی اصطلاحا طراحی سلسله‌مراتبی (Hierarchical) گفته می‌شود.

برنامه شماره ۵ (طراحی mux 4*1)(ادامه)



- نتیجه سنتز با ابزار XST ، ابتدا به صورت زیر خواهد بود. اگر روی هر یک از mux21ها کلیک کنیم، داخل mux21 ظاهر می‌شود (شکل صفحه بعد). این امر، خود گویای سلسله مراتبی بودن طراحی می‌باشد.

برنامه شماره ۵ (طراحی mux 4*1)(ادامه)



- در این شکل، روی m0 کلیک شده است و در نتیجه، داخل آن در ابزار شبیه‌ساز نشان داده شده است.