زبان های توصیف سخت افزار و مدارها

امیر خورسندی

زمستان ۱۴۰۱

دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

جملات شرطی و حلقه ها

امیر خورسندی

ساختار if – else

- فرمت کلی :

```
if(<expression>)
    <True_Statement>;
else
    <False_Statement>;
```

- اگر عبارت شرطی برابر یک منطقی باشد عبارت اول و گر نه عبارت بعد از else اجرا می شود.
- باید حتماً درون بلوک initial یا always به کار رود.

جملات شرطی

امیر خورسندی

ساختار if – else (ادامه)

```
always @ (posedge Clk)
   if(T)
      Q=~Q;
   else
      Q=Q;
```

```
always @ (posedge Clk)
   if(T)
      Q=~Q;
```

- قسمت else به بعد اختیاری است.
- اگر برای هر یک از عبارات درست و یا غلط بیش از یک جمله داشته باشیم از begin و end استفاده می کنیم.

جملات شرطی

امیر خورسندی

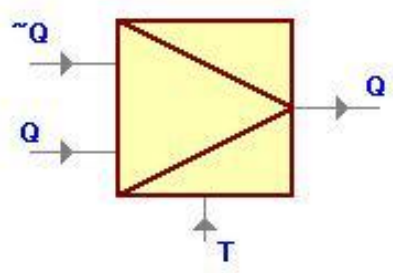
ساختار if – else (ادامه)

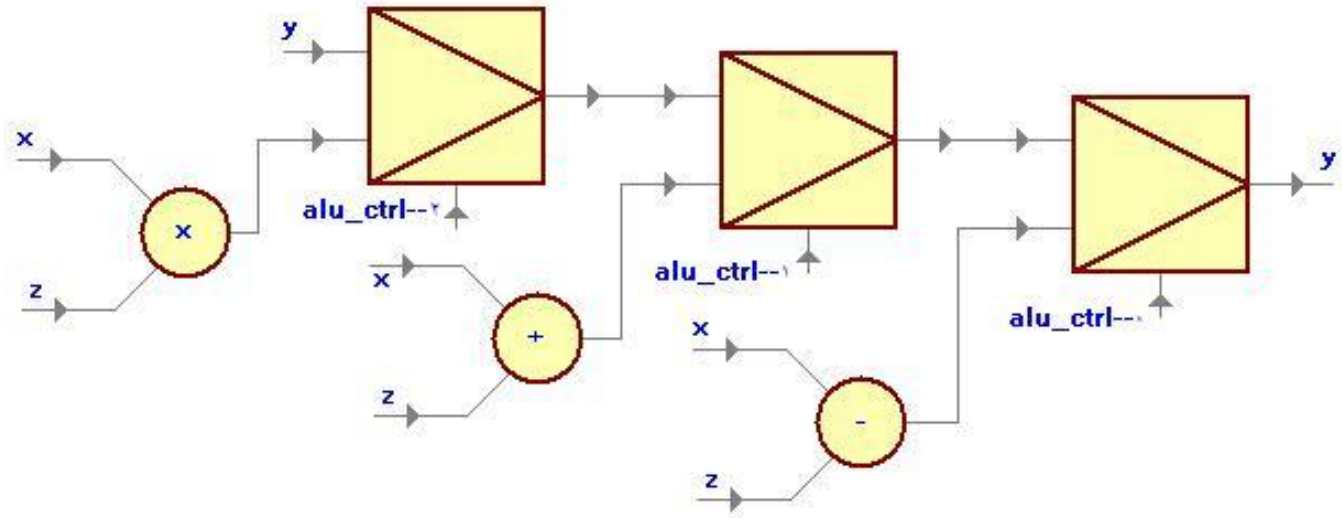
```
always @ (posedge Clk)
   if(alu_ctrl==0)
      y=x-z;
   else if(alu_ctrl==1)
      y=x+z;
   else if(alu_ctrl==2)
      y=x*z;
   else
      $display
("Invalid ALU Ctrl Code")
```

- این ساختار به صورت تو در تو هم می تواند استفاده شود.

ساختار if – else (ادامه)

✓تحقق مداری مثال های قبل بدین صورت است :





جملات شرطی

امیر خورسندی

انشعاب چند گانه

- برای پرهیز از لایه های زیاد در ساختارهای شرطی تو در تو
- فرمت کلی :

```
case(<expression>)
    alter#1: Statement_1;
    ...
    alter#n: Statement_n;
    default: Statement_def;
endcase
```

```
case (exp)
        1: x = 1;
        2: x = 2;
        default : x = 4;
endcase
```

جملات شرطی

امیر خورسندی

انشعاب چند گانه (ادامه)

- ابتدا expression ارزیابی می شود.
- مقدار به دست آمده به ترتیب با alter#1 تا alter#n مقایسه می شود.
- عبارت مربوط به نخستین شرط محقق شده اجرا می شود.
- قسمت default اختیاری است.
- تک تک بیت ها با در نظر گرفتن 0,1,x,z اجرا می شوند.
- اگر تعداد بیت های expression و alter برابر نباشند، عبارت کوچکتر از سمت چپ با صفر پر می شود.

انشعاب چند گانه (ادامه)

```
always @(posedge Clk)
        case (exp)
                2'b10:Out = 1;
                3:Out = 2;
                2:Out = 3;
                default : Out = 4;
        endcase
```

casex - casez

- از لحاظ ساختار کاملاً مشابه case هستند.
- در casez مقادیر z و ؟ در عبارات به صورت Don’t care در نظر گرفته می شوند.
- در casex مقادیر x در عبارات به صورت Don’t care در نظر گرفته می شوند.

casex – casez

مقدار در بخش alter	مقادیر سیگنال قابل تطبیق در case	مقادیر سیگنال قابل تطبیق در casez	مقادیر سیگنال قابل تطبیق در casex
0	0	0	0
1	1	1	1
x	x	x	0, 1, x, z
z	z	0, 1, x, z	z
?	unused	0, 1, x, z	unused
default	0, 1, x, z	0, 1, x, z	0, 1, x, z

حلقه ها

- برای تکرار در اجرای یک بخش به صورت متناهی و یا نامتناهی استفاده می شوند.
- انواع حلقه ها عبارتند از:

while, for, repeat, forever

حلقه while

- فرمت کلی :

```
while <expression>
    <statement>;
```

- تا زمانی که عبارت expression درست باشد، عبارت statement اجرا می شود.

حلقه while (ادامه)

```
always @(clkR)
   while(cntR != 0)
      begin
         cntR = cntR-1;
        Out = cntR;
      end
```

حلقه for

- فرمت کلی :

for (<initialization>;<end_condition>;<index_ctrl>)

- مقدار دهی اولیه
- بررسی شرط ادامه حلقه
- عبارت تغییر مقدار متغیر کنترلی حلقه

for برای حلقه های دارای مرتبه تکرار مشخص و while برای حلقه های منوط به برقراری یک شرط خاص به کار می رود.

حلقه for (ادامه)

```
for(cntW=0;cntW<=14; cntW=cntW+1)
        begin
                $display (cntW,$time);
                Out = cntW;
        end
```

اگر در قسمت initializing حلقه for فقط مقدار ثابت به کار رود و سیگنال استفاده نشود، آن گاه این حلقه قابل سنتز خواهد بود.

حلقه repeat

- فرمت کلی :

```
repeat (<repeat_number>)
    <statement>;
```

- تعداد تکرار می تواند یک ثابت، مقدار یک متغیر و یا یک سیگنال باشد.
- تعداد تکرار فقط یک بار و در ابتدای حلقه محاسبه می شود.

حلقه repeat (ادامه)

```
parameter iter = 8;
initial
    begin
        Cnt=0;
        repeat(iter)
            Cnt=Cnt+1;
    end
```

حلقه forever

- فرمت کلی :

```
forever <statement>;
```

- برای ساخت حلقه های بی نهایت به کار می رود.
- شامل هیچ عبارت کنترلی نیست.
- مثل یک حلقه while است که شرط آن همواره درست باشد.
- تا انتهای شبیه سازی و یا فراخوانی تابع $finish اجرا می شود.
- با دستور disable می تواند متوقف شود.
- معمولا همراه ساختارهای زمانی و کنترل زمانی به کار می رود.

حلقه forever (ادامه)

```
Initial
   begin
      Clk=1'b0;
      forever #10 Clk=~Clk;
   end
```

ثابت ها و پارامترها

- تعریف ثابت ها با `define

```
`define WORD_SIZE 32

reg [`WORD_SIZE-1:0] Data;
```

- تعریف پارامترها با parameter

```
parameter port_id = 0;
//
defparam m1.port_id = 1;
defparam m2.port_id = 2;
```

فراخوانی آرایه ای از ماژول ها

```
wire [3:0] in, out;

generate
        genvar i;
        for (i = 0; i < 4; i = i + 1);
                begin :
                        dff my_dff(clk, in[i], out[i]);
                end
endgenerate
```