第11章 常用数字器件的描述

本章对数字电路中常用器件进行逻辑描述，包括基本门电路、组合逻辑器件和时序逻辑器件。

11.1组合器件的描述

　　常用的组合逻辑电路除基本逻辑门外、有编码器、译码器、数据选择器、数据比较器、三态门和奇偶校验器等多种类型。

11.1.1 基本逻辑门

数字电路共有与门、或门、非门、与非门、或非门、异或门和同或门共七种逻辑门，因此，Verilog内置了实现与、或、非、与非、或非、异或和同或功能的七种基元。调用这些基元很容易描述基本门电路。

【例1】基本逻辑门的描述。

基本逻辑门可采用行为描述、数据流描述和结构多种方式进行描述。结构描述调用Verilog基元例化实现。

module Basic\_Gates (a,b,Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor);

// 端口描述

input a,b;

　 output Yand,Yor,Ynot,Ynand,Ynor,Yxor,Yxnor;

　 // 结构描述

　 and (Yand,a,b);

　 or (Yor,a,b);

　 not (Ynot,a);

　 nand (Ynand,a,b);

　 nor (Ynor,a,b);

　 xor (Yxor,a,b);

　 xnor (Yxnor,a,b);

　 endmodule

11.1.2 编码器

编码器有4线-2线、8线-3线、16线-4线等多种类型。74HC148为常用的8线-3线优先编码器，用于将8个高/低电平信号编成3位二进制代码。

Verilog中的条件语句if-else if-else和分支语句casex/casez本身隐含有优先级的概念，都可以用于描述优先编码器。

【例2】用条件语句描述74HC148。

module HC148a(S\_n,I\_n,Y\_n,YS\_n,YEX\_n);

// 端口说明

　 input S\_n; // 控制端，低电平有效

　 input [7:0] I\_n; // 输入端，低电平有效

　 output [2:0] Y\_n; // 编码输出，低电平有效

　 output YS\_n; // 无编码输出指示，低电平有效

　 output YEX\_n; // 有编码输出指示，低电平有效

reg [2:0] Y\_n;

reg YS\_n;

reg YEX\_n;

// 行为描述

always @(S\_n,I\_n) // 当控制信号或输入信号发生变化时

　 if (!S\_n) // 控制信号有效时

　 if (I\_n[7]==1'b0)

　 begin Y\_n=3'b000; YS\_n=1'b1; YEX\_n=1'b0; end

　 else if (I\_n[6]==1'b0)

　 begin Y\_n=3'b001; YS\_n=1'b1; YEX\_n=1'b0; end

　 else if (I\_n[5]==1'b0)

　 begin Y\_n=3'b010; YS\_n=1'b1; YEX\_n=1'b0; end

　 else if (I\_n[4]==1'b0)

　 begin Y\_n=3'b011; YS\_n=1'b1; YEX\_n=1'b0; end

　 else if (I\_n[3]==1'b0)

begin Y\_n=3'b100; YS\_n=1'b1; YEX\_n=1'b0; end

　 else if (I\_n[2]==1'b0)

　 begin Y\_n=3'b101; YS\_n=1'b1; YEX\_n=1'b0; end

　 else if (I\_n[1]==1'b0)

　 begin Y\_n=3'b110; YS\_n=1'b1; YEX\_n=1'b0; end

　 else if (I\_n[0]==1'b0)

　 begin Y\_n=3'b111; YS\_n=1'b1; YEX\_n=1'b0; end

　 else // 无编码信号输入时

　 begin Y\_n=3'b111; YS\_n=1'b0; YEX\_n=1'b1; end

　 else // 控制信号无效时

　 begin Y\_n=3'b111; YS\_n=1'b1; YEX\_n=1'b1; end

　　 endmodule

【例3】用分支语句描述74HC148。

module HC148b( S\_n,I\_n,Y\_n,YS\_n,YEX\_n);

input S\_n; // 器件使能端，低电平有效

input [7:0] I\_n; // 输入

output [2:0] Y\_n; // 编码表出

output YS\_n; // YS\_n表示无编码信号输入，低电平有效

output YEX\_n; // YEX\_n表示有编码信号输入，低电平有效

reg [2:0] Y\_n;

reg YS\_n;

reg YEX\_n;

// 行为描述

always @(S\_n,I\_n)

if (!S\_n)

casex (I\_n)

8'b0???????:

begin Y\_n=3'b000; YS\_n=1'b1; YEX\_n=1'b0; end

8'b10??????:

begin Y\_n=3'b001; YS\_n=1'b1; YEX\_n=1'b0; end

8'b110?????:

begin Y\_n=3'b010; YS\_n=1'b1; YEX\_n=1'b0; end

8'b1110????:

begin Y\_n=3'b011; YS\_n=1'b1; YEX\_n=1'b0; end

8'b11110???:

begin Y\_n=3'b100; YS\_n=1'b1; YEX\_n=1'b0; end

8'b111110??:

begin Y\_n=3'b101; YS\_n=1'b1; YEX\_n=1'b0; end

8'b1111110?:

begin Y\_n=3'b110; YS\_n=1'b1; YEX\_n=1'b0; end

8'b11111110:

begin Y\_n=3'b111; YS\_n=1'b1; YEX\_n=1'b0; end

8'b11111111:

begin Y\_n=3'b111; YS\_n=1'b0; YEX\_n=1'b1; end

endcase

else

begin Y\_n=3'b111; YS\_n=1'b1; YEX\_n=1'b1; end

endmodule

11.1.3 译码器

译码器用于将二进制码翻译为高、低电平信号。74HC138是常用的二进制译码器，用于将3位二进制代码翻译成8个高、低电平信号。

译码器可以采用行为描述、数据流描述和结构描述等多种方式描述。

【例4】 用行为描述方式描述译码器。

module HC138(s1,s2\_n,s3\_n,a,y\_n);

input s1,s2\_n,s3\_n;

input [2:0] a;

output [7:0] y\_n;

reg [7:0] y\_n;

wire s;

assign s={s1&(~s2\_n)&(~s3\_n)};

// 行为描述

always @(s,a)

if ( s)

case (a)

3'b000: y\_n=8'b11111110;

3'b001: y\_n=8'b11111101;

3'b010: y\_n=8'b11111011;

3'b011: y\_n=8'b11110111;

3'b100: y\_n=8'b11101111;

3'b101: y\_n=8'b11011111;

3'b110: y\_n=8'b10111111;

3'b111: y\_n=8'b01111111;

default: y\_n=8'b11111111;

endcase

else

y\_n=8'b11111111;

endmodule

显示译码器是特殊的译码器，用于将BCD或二进制码译成七段码，用于驱动半导体数码管。CD4511是常用的BCD显示译码器，具有灯测试、灭灯和锁存三种附加功能，输出高电平有效。

【例5】CD4511逻辑描述。

module CD4511(LE,BI\_n,LT\_n,D,SEG7);

input LE,BI\_n,LT\_n;

input [3:0] D;

　 　 output [6:0] SEG7;

reg [6:0] SEG7;

　　 always @(LE,BI\_n,LT\_n,D)

　　 if (!LT\_n) // 灯测试信号有效时

　　 SEG7<=7'b1111111; // SEG7: gfedcba

　　 else if (!BI\_n) // 灭灯输入有效时

　　 SEG7<=7'b0000000;

　　 else if (!LE) // 锁存信号无效时

　　 case (D) // dcba

　　 4'b0000: SEG7<=7'b0111111; // 显示 0

　　 4'b0001: SEG7<=7'b0000110; // 显示 1

　　 4'b0010: SEG7<=7'b1011011; // 显示 2

　　 4'b0011: SEG7<=7'b1001111; // 显示 3

　　 4'b0100: SEG7<=7'b1100110; // 显示 4

　　 4'b0101: SEG7<=7'b1101101; // 显示 5

　　 4'b0110: SEG7<=7'b1111100; // 显示 6

　　 4'b0111: SEG7<=7'b0001111; // 显示 7

　　 4'b1000: SEG7<=7'b1111111; // 显示 8

　　 4'b1001: SEG7<=7'b1100111; // 显示 9

　　 default: SEG7<=7'b0000000; // 不显示

　　 endcase

endmodule

11.1.4 数据选择器

数据选择器是在地址信号的作用下，从多路输入数据中选择其中一路数据输出，有2选一 、4选一、8选一等多种类型。

74HC151为常用的8选一数据选择器，具有互补型输出。

【例6】74HC151逻辑描述。

module HC151(s\_n,a,d,y,w\_n);

input s\_n;

input [2:0] a;

input [7:0] d;

output y,w\_n;

reg y;

assign w\_n=~y;

always @(s\_n,a,d)

begin

if (!s\_n)

case (a)

3'b000: y=d[0];

3'b001: y=d[1];

3'b010: y=d[2];

3'b011: y=d[3];

3'b100: y=d[4];

3'b101: y=d[5];

3'b110: y=d[6];

3'b111: y=d[7];

default: y=d[0];

endcase

else

y=1'b0;

end

endmodule

用EDA技术设计数字电路时，可以不受具体器件的限制，可以描述任何需要的功能电路。

【例7】设计四位4选一数据选择器，用于从四路4位信号中选择其中一路输出。

module MUX4b4to1(s\_n,a,d0,d1,d2,d3,y);

input s\_n;

input [1:0] a;

input [3:0] d0,d1,d2,d3; // 四位4路数据

output [3:0] y;

reg [3:0] y;

always @(s\_n,a,d0,d1,d2,d3)

begin

if (!s\_n)

case (a)

2'b00: y=d0;

2'b01: y=d1;

2'b10: y=d2;

2'b11: y=d3;

default: y=d0;

endcase

else

y=4'b0;

end

endmodule

11.1.5 数值比较器

数值比较器用于比较数值的大小，一位比较器和多位比较器之分。

74HC85是四位数值比较器，用于比较两个四位二进制数码的大小。考虑到功能扩展方便，74HC85还附加有三个来自低位比较结果输入端。

【例8】74HC85逻辑描述。

module HC85(a,b,Ia\_gt\_b,Ia\_eq\_b,Ia\_lt\_b,ya\_gt\_b,ya\_eq\_b,ya\_lt\_b);

input [3:0] a,b;

input Ia\_gt\_b,Ia\_eq\_b,Ia\_lt\_b;

output ya\_gt\_b,ya\_eq\_b,ya\_lt\_b;

reg ya\_gt\_b,ya\_eq\_b,ya\_lt\_b;

// gt=greater than, eq=equal,lt=less than.

wire [2:0] Iin;

assign Iin={Ia\_gt\_b,Ia\_eq\_b,Ia\_lt\_b}; // 并置

always @(a,b,Iin)

if (a>b)

begin ya\_gt\_b<=1'b1;ya\_eq\_b=1'b0;ya\_lt\_b<=1'b0; end

else if (a<b)

begin ya\_gt\_b<=1'b0;ya\_eq\_b=1'b0;ya\_lt\_b<=1'b1; end

else if (Iin==3'b100)

begin ya\_gt\_b<=1'b1;ya\_eq\_b=1'b0;ya\_lt\_b<=1'b0; end

else if (Iin==3'b001)

begin ya\_gt\_b<=1'b0;ya\_eq\_b=1'b0;ya\_lt\_b<=1'b1; end

else

begin ya\_gt\_b<=1'b0;ya\_eq\_b=1'b1;ya\_lt\_b<=1'b0; end

endmodule

11.1.6 三态缓冲器

三态缓冲器用于总线驱动或双向数据总线的构建，有三态反相器和三态驱动器两种类型。三态缓冲器的输出有低电平、高电平和高阻三种状态。

74HC240/244是双四位三态缓冲器，74HC240为三态反相器，74HC244为三态驱动器。74HC240/244的功能表如表11-1所示。

表11-1 74HC240/244功能表

|  |  |  |  |
| --- | --- | --- | --- |
| 输入 | | 输出 | |
| *G'* | *A* | 74HC240 | 74HC244 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | z | z |
| 1 | 1 | z | z |

【例9】74HC244逻辑描述。

module HC244(g1\_n,a1,y1,g2\_n,a2,y2);

input g1\_n,g2\_n;

input [3:0] a1,a2;

output [3:0] y1,y2;

assign y1=(!g1\_n)? a1:4'bz; // 第一组

assign y2=(!g2\_n)? a2:4'bz; // 第二组

endmodule

74HC245为八位双向缓冲驱动器，功能如表5-7所示。

表11-2 74HC245功能表

|  |  |  |  |
| --- | --- | --- | --- |
| 输 入 | | 输入/输出 | |
| *OE'* | *DIR* | *A*n | *B*n |
| 0 | 0 | *A=B* | 输入 |
| 0 | 0 | 输入 | *B=A* |
| 1 | X | Z | Z |

【例10】74HC245逻辑描述。

module HC245(a,b,dir,oe\_n);

inout [7:0] a,b;

input dir,oe\_n;

reg [7:0] a,b;

always @(oe\_n,dir,a,b)

if (!oe\_n)

if (dir==1'b0)

begin a<=b; b<=8'bz; end

else

begin b<=a; a<=8'bz; end

else

begin a<=8'bz; b<=8'bz; end

endmodule

11.1.7 奇偶校验器

奇偶校验是数字系统中最基本的检错方法，分为奇校验和偶校验两种。奇偶校验码既可以用软件产生，也可以用硬件实现。

【例11】八位奇偶校验器的逻辑描述。

module parity\_check(din,y\_odd,y\_even);

input [7:0] din; // 数据输入

output y\_odd,y\_even; // 奇校验输出端，偶校验输出端

assign y\_odd=~^din;　　// 奇校验输出

assign y\_even=^din;　　 // 偶校验输出

endmodule

11.2 时序器件的描述

时序逻辑电路是任一时刻的输出不但与当时的输入信号有关，而且与电路的状态也有关系。时序逻辑器件主要有寄存器和计数器两大类，两者均以存储电路为核心构成。

11.2.1 锁存器与触发器

锁存器与触发器是两种基本的存储电路。锁存器在时钟脉冲的有效电平期间工作，触发器在时钟脉冲的有效沿工作。为了使用起来灵活方便，商品化的锁存器/触发器一般都提供附加的复位端和置位端，分为异步和同步两类。

异步复位/置位是与时钟脉冲无关。当复位/置位信号有效时立即将锁存器/触发器置为0或1。

异步置位/复位用always语句实现时，需要将复位/置位信号列入always语句的敏感事件列表中，当复位/置位有效时就能立即执行指定的操作。

【例12】½74HC74功能描述。

module HC74(clk,rd\_n,sd\_n,d,q);

input clk,rd\_n,sd\_n,d;

output reg q;

always @(posedge clk or negedge rd\_n or negedge sd\_n)

if ( !rd\_n )

q<=1'b0;

else if (!sd\_n)

q<=1'b1;

else

q<=d;

endmodule

【例13】½74HC112功能描述。

module HC112(clk,rd\_n,sd\_n,j,k,q);

input clk,rd\_n,sd\_n,j,k;

output reg q;

always @(posedge clk or negedge rd\_n or negedge sd\_n)

if (!rd\_n)

q<=1'b0;

else if (!sd\_n)

q<=1'b1;

else

case ({j,k})

2'b00: q<=q; // 保持

2'b01: q<=1'b0; // 置0

2'b10: q<=1'b1; // 置1

2'b11: q<=~q; // 翻转

endcase

endmodule

同步复位/置位只有当时钟脉冲的有效沿到来时才能使触发器复位或置位。同步复位/置位用always语句实现时，always语句只对时钟有效沿敏感，然后在always内部语句块中检测置位/复位是否有效。例如，同步复位*D*触发器的功能描述如下：

module dff\_sync\_reset(clk,rst\_n,d,q);

input clk,rst\_n,d;

output reg q;

always @(posedge clk)

if (!rst\_n)

q<=1'b0;

else

q<=d;

endmodule

74HC573是8位三态寄存器，内部由*D*锁存器构成，用于数据或地址信号的锁定。

【例14】74HC573功能描述。

module HC573(D,LE,OE\_n,Q);

input [7:0] D;

input LE,OE\_n;

output [7:0] Q;

reg [7:0] Q;

reg Qtmp;

always @(D,LE)

if (LE)

Qtmp<=D;

always @(OE\_n)

if (!OE\_n)

Q<=Qtmp;

else

Q<=8'bz;

endmodule

74HC574是8位三态寄存器，内部由*D*触发器构成。与74HC573作用类似，用于数据或地址信号的锁定。

【例15】74HC574功能描述。

module HC574(D,Clk,OE\_n,Q);

input [7:0] D;

input Clk,OE\_n;

output [7:0] Q;

reg [7:0] Q;

reg Qtmp;

always @(posedge Clk)

Qtmp<=D;

always @(OE\_n)

if (!OE\_n)

Q<=Qtmp;

else

Q<=8'bz;

endmodule

74HC194是四位双向移位寄存器，具有异步复位，同步左移、右移、并行输入和保持功能。

【例16】74HC194功能描述。

module HC194(clk,Rd\_n,s,d,dil,dir,q);

input clk,Rd\_n,dil,dir;

input [0:3] d;

input [0:1] s;

output [0:3] q;

reg [0:3] q;

always @(posedge clk or negedge Rd\_n)

if (!Rd\_n)

q<=4'b0000;

else

case (s)

2'b00: q<=q; // 保持

2'b01: q[0:3]<={q[1:3],dil}; // 左移

2'b10: q[0:3]<={dir,q[0:2]}; // 右移

2'b11: q<=d; // 并行输入

default: q<=q;

endcase

endmodule

11.2.2 计数器

计数器是应用最广泛的时序逻辑器件，分为同步计数器和异步计数器两大类。根据计数容量又可分为二进制、十进制和其它进制计数器，根据计数方式又可分加法、减法和加/减计数器三种类型。

HC160/162为同步十进制计数器，HC161/163为同步十六进制计数器。HC160/161与HC162/163管脚排列完全相同，所不同的是，前两者为异步复位，后两者为同步复位。

【例17】74HC160功能描述。

module HC160(CLK,Rd\_n,LD\_n,EP,ET,D,Q,CO);

input CLK；

input Rd\_n,LD\_n,EP,ET;

input [3:0] D;

output reg [3:0] Q;

output reg CO;

always @(posedge CLK or negedge Rd\_n) // 计数逻辑

if (!Rd\_n)

Q<=4'b0000;

else if (!LD\_n)

Q<=D;

else if (EP & ET)

if (Q==4'b1001)

Q<=4'b0000;

else

Q<=Q+1'b1;

always @(Q,ET) // 进位逻辑

if ((Q==4'b1001) & ET)

CO<=1'b1;

else

CO<=1'b0;

endmodule

【例18】74HC163功能描述。

module HC163(CLK,CLR\_n,LD\_n,EP,ET,D,Q,CO);

input CLK,CLR\_n,LD\_n,EP,ET;

input [3:0] D;

output reg [3:0] Q;

output reg CO;

always @(posedge CLK )

if (!CLR\_n) // 同步复位

Q<=4'b0000;

else if (!LD\_n)

Q<=D;

else if (EP & ET)

Q<=Q+1'b1;

always @(Q,ET)

if ((Q==4'b1111) & ET)

CO<=1'b1;

else

CO<=1'b0;

endmodule

加/减计数器在时钟脉冲下既能实现加法计数，也能实现减法计数，分为单时钟和双时钟两种。单时钟加/减计数器的计数方式由加/减控制端控制，双时钟加/减计数器则采用不同的时钟输入控制加法计数或减法计数。

74HC191是单时钟16进制加/减计数器，*U'/D*是计数控制端，当*U'/D*=0时实现加法计数，*U'/D*＝1时实现减法计数。

【例19】74HC191功能描述。

module HC191(clk,S\_n,LD\_n,UnD,D,Q);

input clk,S\_n,LD\_n,UnD;

input [3:0] D;

output reg [3:0] Q;

always @(posedge clk or negedge LD\_n)

if (!LD\_n)

Q<=D;

else if (!S\_n)

if (!UnD)

Q<=Q+1'b1;

else

Q<=Q-1'b1;

endmodule

11.2.3 分频器

分频器是用于降低信号的频率。设分频器输入时钟信号clk的频率为*f*clk，分频输出信号的频率为*f*fpout，则*N*分频器的输出信号fpout与时钟脉冲clk的频率关系为

*f*fpout=*f*clk/*N*

分频器有偶分频器和奇分频器两种。偶分频器的分频系数*N*为偶数，其实现原理为：将待分频的信号作为计数器的时钟触发计数器计数，当计数值从0计到*N*/2-1时，分频输出信号翻转，同时将计数器清零，下一个时钟到来时重新开始计数。如此循环反复，可以实现任意偶数分频。

【例20】偶分频器功能描述。

module fp\_even(clk,rst\_n,fp\_out);

input clk,rst\_n;

output fp\_out;

reg fp\_out;

reg [3:0] count; // n位计数器，容量根据分频倍数进行调整

parameter N = 10; // 分频系数N，计数器容量应满足2n≥(N/2)2n-1>N

always @ (posedge clk)

if (!rst\_n)

begin count <= 1'b0; fp\_out<= 1'b0; end

else

if ( count < N/2-1)

count <= count + 1'b1;

else

begin count <= 1'b0; fp\_out<= ~fp\_out; end

endmodule

奇分频器的分频系数*N*为奇数，实现相对复杂一些。对于占空比为50%的奇分频器，具体的实现方法为：将待分频的信号作为计数器的时钟，在时钟脉冲的上升沿和下降沿同时进行*N*进制计数，当计数值从0计到(*N*-1)/2时分频输出信号翻转，然后再计（*N*-1）/2个时钟分频输出信号再次翻转，分别得到一个占空比非50%的分频信号，将两个分频输出信号相或，可得到占空比为50%的奇分频信号。

【例21】奇分频器功能描述。

module fp\_odd(clk,rst\_n,fp\_out);

input clk,rst\_n;

output fp\_out;

reg [3:0] count1,count2; // 内部计数器，容量根据分频倍数进行调整

reg clk1,clk2;

parameter N = 11; // 分频系数N，计数器容量应满足2n>N

assign fp\_out= clk1 | clk2;

always @(posedge clk)

if (!rst\_n)

begin count1 <= 1'b0; clk1 <= 1'b0; end

else

if (count1 < (N - 1))

begin

count1 <= count1 + 1'b1;

if (count1 == (N-1)/2) clk1<= ~clk1;

end

else

begin clk1<= ~clk1; count1 <= 1'b0; end

always @ (negedge clk)

if (!rst\_n)

begin count2 <= 1'b0; clk2 <= 1'b0; end

else if (count2 < (N - 1))

begin

count2 <= count2 + 1'b1;

if (count2 == (N-1)/2) clk2 <= ~clk2;

end

else

begin clk2 <= ~clk2; count2 <= 1'b0; end

endmodule

11.2.4 双口RAM

存储器分为RAM和ROM两类，RAM为时序电路，ROM为组合电路。在Verilog中，ROM一般采用FPGA中的RAM块实现，只需要预先将数据存入RAM块中即可。

存储器可以看作是存储单元的集合，在Verilog中由寄存器数组描述。

双口是指RAM的读操作和写操作在不同的端口进行，框图如图11-1所示，其中clock为时钟端，wren为写控制端，wraddr为写地址端，rdaddr为读地址端，data为数据输入端，q为数据输出端。

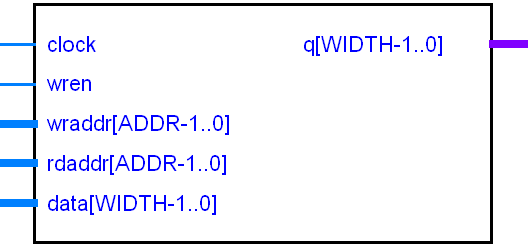


　　　　　 　图11-1　双向RAM框图

【例22】16×8双口RAM功能描述。

module dpram16x8 (clock,wren,wraddr,rdaddr,data,q);

parameter WIDTH=8,DEPTH=16,ADDR=4;

input clock;

input wren;

input [ADDR-1:0] wraddr,rdaddr;

input [WIDTH-1:0] data;

output [WIDTH-1:0] q;

reg [WIDTH-1:0] mem\_data [DEPTH-1:0]; // 定义存储器

always @(posedge clock) // 写过程

if (wren)

mem\_data[wraddr]=data;

assign q=mem\_data[rdaddr]; // 读操作

endmodule

11.3 设计项目

基于FPGA设计数字系统，不再受具体器件功能与性能的限制，可以根据需要定制单元电路。同时，由于FPGA的高速性和易构性，基于FPGA设计的数字系统与传统的基于ASIC设计有着无可比拟的优点。

11.3.1 100MHz频率计设计

集成计数器74HC160从时钟到输出的典型传输延迟时间约为18ns，因此最高工作速度只能达到50MHz左右，无法对100MHz频率信号进行计数。目前，FPGA的传输延迟时间普遍小于8ns，因此工作速度很容易达到100MHz以上。

100MHz频率计的总体设计方案与6.7.2节中的方案相同，只是模块在EDA软件中设计，用FPGA实现。具体设计方法是：对100MHz以下信号进行测频需要用8位BCD码计数器，在EDA软件直接调用8个74160计数器级联实现（如图11-2所示），然后封装成计数器模块（cnt10p8x1）以便在顶层设计模块中调用。

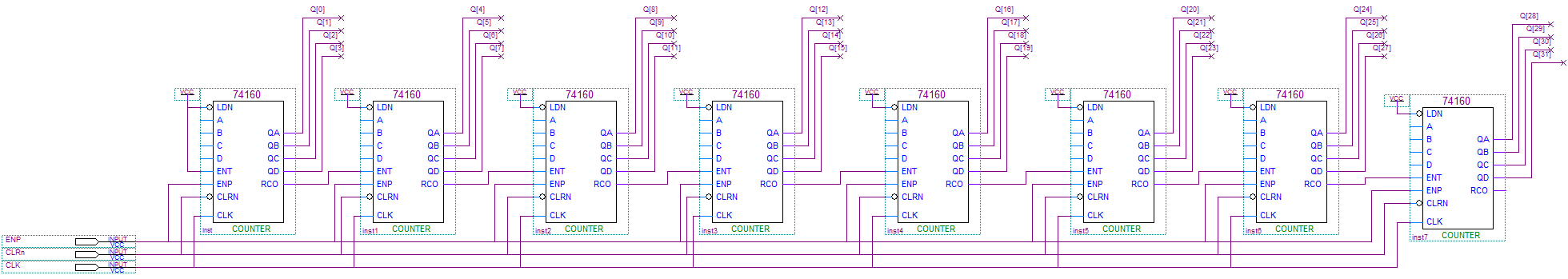


图11-2 8位BCD码计数器

8位BCD码计数器共有32位输出，因此需要设计一个32位锁存器，在主控模块的控制下锁存计数结果，供译码显示电路使用。锁存器的设计代码如下：

module latch32 ( en, // 锁存使能信号

din, // 数据输入

dout // 数据输出

);

input en;

input [31:0] din;

output [31:0] dout;

reg [31:0] dout;

always @(en,din)

if (en==1'b1) dout <= din;

endmodule

显示译码器CD4511按本章例5所示代码设计，然后用结构描述方式构成8位数码管译码模块电路，再封装成模块以便在顶层设计中调用。具体代码如下：

module SEG7\_LUT\_8 (oSEG0, oSEG1,oSEG2,oSEG3,oSEG4, oSEG5,oSEG6,oSEG7,

iDIG );

input [31:0] iDIG;

output [6:0] oSEG0,oSEG1,oSEG2,oSEG3,oSEG4,oSEG5,oSEG6,oSEG7;

// 结构描述

CD4511 u0 ( oSEG0,iDIG[3:0] );

CD4511 u1 ( oSEG1,iDIG[7:4] );

CD4511 u2 ( oSEG2,iDIG[11:8] );

CD4511 u3 ( oSEG3,iDIG[15:12] );

CD4511 u4 ( oSEG4,iDIG[19:16] );

CD4511 u5 ( oSEG5,iDIG[23:20] );

CD4511 u6 ( oSEG6,iDIG[27:24] );

CD4511 u7 ( oSEG7,iDIG[31:28] );

endmodule

主控电路不再需要用集成计数器附加门电路构成，可以直接根据功能要求进行描述，然后封装成模块以便在顶层设计中调用。具体代码如下：

module ftctrl ( clk, clr\_n, cl, le );

input clk; // 8Hz

output clr\_n;

reg clr\_n;

output cl;

reg cl;

output le;

reg le;

reg [3:0] q;

always @(posedge clk)

begin

if (q>=4'b1001)

q<=4'b0000;

else

q<=q+1'b1;

end

always @(q)

begin

case (q)

4'b0000: begin clr\_n<=1'b0; cl<=0; le<=0; end

4'b0001: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b0010: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b0011: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b0100: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b0101: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b0110: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b0111: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b1000: begin clr\_n<=1'b1; cl<=1; le<=0; end

4'b1001: begin clr\_n<=1'b1; cl<=0; le<=1; end

default: begin clr\_n<=1'b1; cl<=0; le<=0; end

endcase

end

endmodule

其中clr\_n为计数器清零信号，cl为闸门控制信号，le为显示控制信号。

顶层模块采用原理图方法描述，整体设计如图11-3所示，其中分频模块fpdiv用于将开发板提供的50MHz晶振信号分频为8Hz，作为主控电路ftctrl的时钟脉冲。

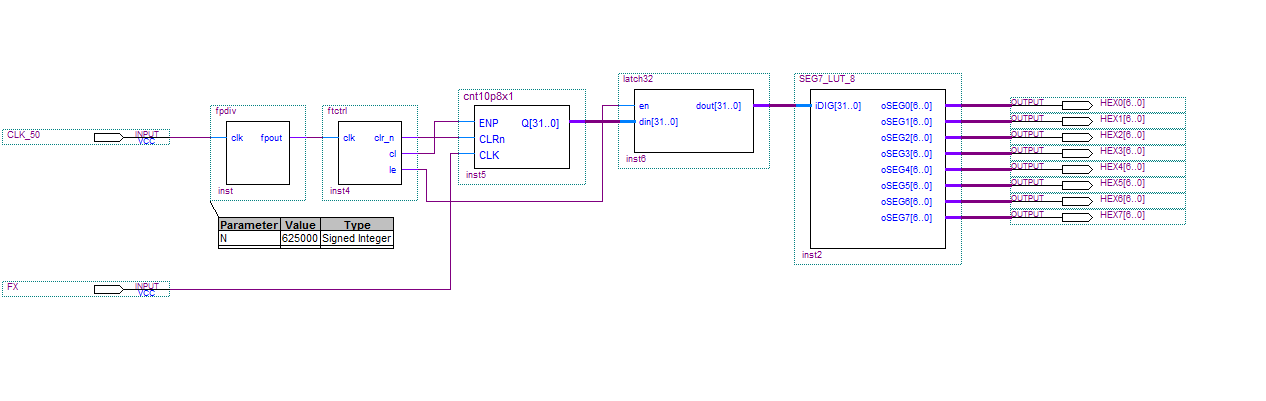


图11-3顶层设计图

11.3.2 正弦波信号源设计

基于集成器件设计DDS信号源时，仍然会受到芯片性能的限制，难以实现高速度和高分辨率。基于EDA技术设计时，相位累加器可直接采用Verilog代码描述，正弦波ROM可直接调用ROM模块实现，而不必拘泥于具体器件。

为了节约有限的FPGA存储资源，本方案只存储1/4周期的正弦波，再利用正弦波波形的对称性，恢复出完整的正弦波形。

相位累位器设计代码如下：

module padder\_addr ( clk, fword, addrout, datinv );

input clk;

input [14:0] fword; // 15位控制字输入

output reg [14:0] addrout;

output reg datinv; // 数据反相标志

reg [16:0] qq; // 15+2位相位控制字

always @ (posedge clk)

begin

qq<=qq+fword; // 相位累加

case (qq[16:15])

2'b00: begin addrout[14:0]<= qq[14:0]; datinv=0; end

2'b01: begin addrout[14:0]<=~qq[14:0]; datinv=0; end

2'b10: begin addrout[14:0]<= qq[14:0]; datinv=1; end

2'b11: begin addrout[14:0]<=~qq[14:0]; datinv=1; end

endcase

end

endmodule

正弦波ROM可以直接定制ROM模块实现，容量为32k×10位，存储1/4周期正弦波采样值。正弦波采样程序（C语言）如下：

#include <math.h>

#define PI 3.1415926

void main (void)

{

unsigned int dat;

float x;

unsigned int i;

for (i=0;i<32768;i++) // 采样1/4正弦波, 215=32768

{

x=sin(2\*PI/131072\*i); // 217=131072

dat=((x+1)/2\*1023);

printf("%d : %d; \n",i,dat);

}

}

由于只存储了1/4正弦波，所以需要在相位累加器和输出数据代码控制代码的共同作用下，恢复成一个完整的正弦波。输出正弦数据控制代码如下：

module datainv (

din,

datflag,

dout

);

input [9:0] din;

input datflag;

output [9:0] dout;

assign dout=(datflag? ~din : din);

endmodule

DDS整体设计如图11-4所示，外接10位D/A转换器即可输出正弦波。当相位累加器的时钟频率可取128kHz时，输出正弦波的频率范围为1Hz~32kHz，分频率为1Hz。

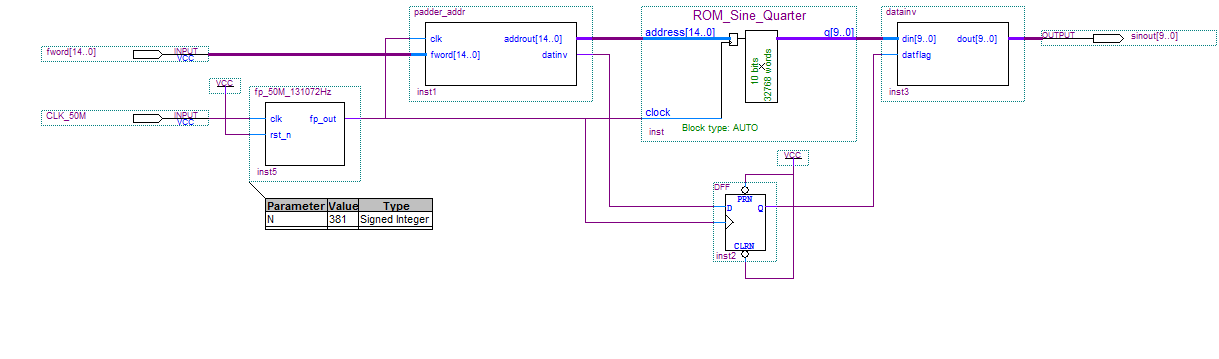


图11-4 正弦波信号源顶层设计图

11.3.3 VGA彩格控制电路设计

VGA（Video Graphics Array）是IBM推出的视频显示标准接口，具有分辨率高、显示速率快和色彩丰富等优点，目前仍广泛应用于使用VGA显卡的计算机、笔记本电脑、投影仪和液晶电视等电子产品。VGA采用D-SUB 15接口，如图11-5所示。



(a) 公头 (b) 母头

图11-5 VGA接口

VGA主要有5个信号：行同步信号和场同步信号，以及红、绿、蓝三基色模拟信号。要能正确地显示图像，必须提供精确的行同步和场同步信号。

行、场同步信号的时序如图11-6所示，均分为前沿、同步头、后沿和显示四个段，不同的是行同步信号以像素（Pixel）为单位，而场同步信号则以行（Line）为单位。同步脉冲头低电平有效，b、c和d段时则为高电平，c段时显示三基色信号，其余时段则处于消隐状态。



(a) 行时序图



(b) 场时序图

图11-6 VGA标准参考时序图

对于分辨率为640×480、刷新频率为60Hz的图像来说，每行的总像素点为800个，其中有效像素为c段的640个；每场的总行数为525行，其中有效行数为c段的480行，如表11-3所示。

表11-3 640×480@60Hz模式行、场同步信号参数值

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 显示模式 | 行同步信号（Pixels） | | | | |
| 640×480  60Hz | a段 | b段 | c段 | d段 | 总像数 |
| 96 | 48 | 640 | 16 | 800 |
| 显示模式 | 场同步信号（Lines） | | | | |
| 640×480  60Hz | a段 | b段 | c段 | d段 | 总行数 |
| 3 | 32 | 480 | 10 | 525 |

设计任务：设计一个VGA显示控制器，能够在640480@60Hz模式下显示8×8 彩格图像。

设计过程：VGA显示设计与VGA硬件电路有关。若VGA接口电路如图11-7所示，用Cyclone III系列FPGA输出四位红、绿、蓝三基色数字信号，通过权电阻网络转换为模拟信号，提供给VGA接口输出。

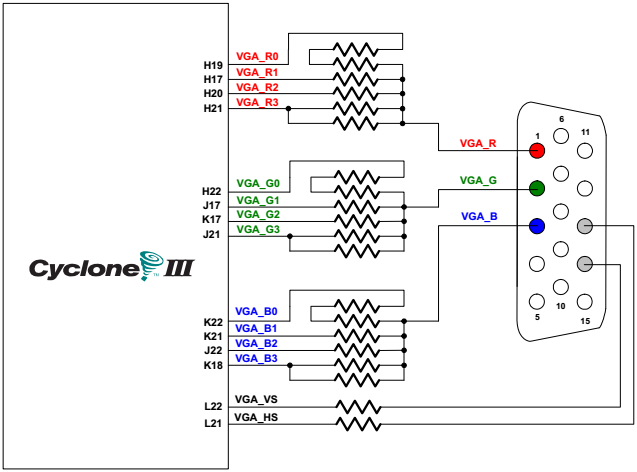


图11-8 VGA接口电路

VGA彩格显示控制Verilog代码如下：

module VGA\_Pattern (

OSC\_50, // 板载50 MHz晶振信号

VGA\_HS, // 行同步信号

VGA\_VS, // 场同步信号

VGA\_R, // 红色分量，4位

VGA\_G, // 绿色分量，4位

VGA\_B // 蓝色分量，4位

);

input OSC\_50;

output reg VGA\_HS;

output reg VGA\_VS;

output reg [3:0] VGA\_R;

output reg [3:0] VGA\_G;

output reg [3:0] VGA\_B;

// 640×480 行参数值(pixels)

parameter H\_FRONT=16; // d段，前沿

parameter H\_SYNC=96; // a段，同步头

parameter H\_BACK=48; // b段，后沿

parameter H\_ACT=640; // c段，显示段

parameter H\_BLANK=H\_FRONT+H\_SYNC+H\_BACK;

parameter H\_TOTAL=H\_FRONT+H\_SYNC+H\_BACK+H\_ACT;

// 640×480 场参数值 (lines)

parameter V\_FRONT=10;

parameter V\_SYNC=3;

parameter V\_BACK=32;

parameter V\_ACT=480;

parameter V\_BLANK=V\_FRONT+V\_SYNC+V\_BACK;

parameter V\_TOTAL=V\_FRONT+V\_SYNC+V\_BACK+V\_ACT;

// 信号及变量定义

reg VGA\_CLK; // VGA时钟，640×480@60Hz时应为25MHz

reg [10:0] H\_Cont; // 行计数器

reg [10:0] V\_Cont; // 列计数器

reg [10:0] X; // 行坐标

reg [10:0] Y; // 列坐标

// 用板载晶振二分频产生25MHz VGA时钟

always @(posedge OSC\_50)

VGA\_CLK=~VGA\_CLK;

// 行处理过程

always@(posedge VGA\_CLK)

begin

if(H\_Cont<H\_TOTAL) // 行计数器

H\_Cont<= H\_Cont+1'b1;

else

H\_Cont<= 0;

if(H\_Cont==H\_FRONT-1) // 生成行同步信号VGA\_HS

VGA\_HS <= 1'b0;

if(H\_Cont==H\_FRONT+H\_SYNC-1)

VGA\_HS <= 1'b1;

if(H\_Cont>=H\_BLANK) // 计算行像素坐标

X<=H\_Cont-H\_BLANK;

else

X<=0;

end

// 场处理过程

always@(posedge VGA\_HS )

begin

if(V\_Cont<V\_TOTAL) // 场计数器

V\_Cont<=V\_Cont+1'b1;

else

V\_Cont<=0;

if(V\_Cont==V\_FRONT-1) // 生成场同步信号

VGA\_VS <= 1'b0;

if(V\_Cont==V\_FRONT+V\_SYNC-1)

VGA\_VS <= 1'b1;

if(V\_Cont>=V\_BLANK) // 计算场（行数）坐标

Y<=V\_Cont-V\_BLANK;

else

Y<=0;

end

// 彩格图像生成

always@(posedge VGA\_CLK)

begin

VGA\_R<= (Y<120) ? 4 : // 红色分量定义

(Y>=120 && Y<240) ? 8 :

(Y>=240 && Y<360) ? 12 :

15;

VGA\_G<= (X<80) ? 2 : // 绿色分量定义

(X>=80 && X<160) ? 4:

(X>=160 && X<240) ? 6:

(X>=240 && X<320) ? 8:

(X>=320 && X<400) ? 10 :

(X>=400 && X<480) ? 12 :

(X>=480 && X<560) ? 14:

15;

VGA\_B<=(Y<60) ? 15: // 蓝色分量定义

(Y>=60 && Y<120) ? 14 :

(Y>=120 && Y<180) ? 12 :

(Y>=180 && Y<240) ? 10 :

(Y>=240 && Y<300) ? 8 :

(Y>=300 && Y<360) ? 6 :

(Y>=360 && Y<420) ? 4 :

2 ;

end

endmodule

将上述代码在EDA软件中完成编译、综合、适配和引脚锁定后下载到开发板，能够显示640×480@60Hz棋盘格图像。