设计文档

5 控制模块VHDL语言源程序

源程序：

```
library ieee;
use ieee.std_logic_1164.all;

entity WaGaCPU is  --给WaGaCPU(我的CPU)定义输入输出量
   PORT(C,SWC,SWB,SWA,W1,W2,W3,CLRJ:IN std_logic;  --反馈，控制，节拍输入
    IR:IN std_logic_vector(7 DOWNTO 4);  --指令输入

SKIP,TJ,RAM_BUSJ,ALU_BUSJ,RS_BUSJ,SW_BUSJ,LDRi,LDIR,LDARJ,ARpp,LDPCJ,PCpp,LDDR1,LDDR2,CERJ,CELJ,LRWJ,S3,S2,S1,S0,M,CnJ:OUT std_logic;  --控制输出
    kami:OUT std_logic_vector(0 TO 5));  --中间信号察看输出，可以察看con向量，以及控制con向量改变的中间信号con01,const

--开始固定管脚，其依据来于曾经成功过的一次自动分布，固定管脚使得以后编辑代码再烤入isp1032之时无需重新拔插导线
attribute LOC:string;
--输入管脚
--反馈输入管脚
attribute LOC of C:signal is "P37";
--清零信号管脚，CLR#，平时为1，按下为0
attribute LOC of CLRJ:signal is "P18";
--程式指令输入管脚
attribute LOC of IR:signal is "P57P69P33P27";
--控制台输入管脚
attribute LOC of SWA:signal is "P15";
attribute LOC of SWB:signal is "P60";
attribute LOC of SWC:signal is "P16";
--节拍输入管脚
attribute LOC of W1:signal is "P9";
attribute LOC of W2:signal is "P49";
attribute LOC of W3:signal is "P14";

--输出管脚
--四大数据总线开关控制输出管脚
attribute LOC of ALU_BUSJ:signal is "P41";
attribute LOC of RAM_BUSJ:signal is "P71";
attribute LOC of RS_BUSJ:signal is "P70";
attribute LOC of SW_BUSJ:signal is "P28";
--数据通路四寄存器控制输出管脚
attribute LOC of LDRI:signal is "P35";
--ALU运算控制信号输出管脚
attribute LOC of S0:signal is "P4";
attribute LOC of S1:signal is "P48";
```

```
attribute LOC of S2:signal is "P47";

attribute LOC of S3:signal is "P46";

attribute LOC of M:signal is "P5";

attribute LOC of CNJ:signal is "P29";

--ALU运算数寄存器控制输出管脚

attribute LOC of LDDR1:signal is "P3";

attribute LOC of LDDR2:signal is "P6";

--双端存储器控制输出管脚

attribute LOC of CELJ:signal is "P79";

attribute LOC of CERJ:signal is "P78";

attribute LOC of LRWJ:signal is "P74";

--存储器数据与指令地址寄存器控制输出管脚

attribute LOC of LDARJ:signal is "P75";

attribute LOC of ARPP:signal is "P31";

attribute LOC of LDPCJ:signal is "P30";

attribute LOC of PCPP:signal is "P32";

--指令寄存器控制输出管脚

attribute LOC of LDIR:signal is "P26";

--时序节拍控制输出管脚

attribute LOC of SKIP:signal is "P8";

attribute LOC of TJ:signal is "P45";

--察看各中间信号，调试人员专用的输出管脚，在实际中可以没有的

attribute LOC of KAMI:signal is "P50P73P68P10P7P13";


end WaGaCPU;  --接口定义结束


architecture Operation of WaGaCPU is  --定义输出数据与输入数据之间的关系

signal con:std_logic_vector(0 TO 3);  --con向量是四个寄存器，用来记录状态，根据con状态来进行各种控制，是控制台上的大BOSS。con(0)为1的时候，循环进行控制台中的循环部分；con(1~3)其实就是记录SWC~A，为了防止在进入在最初读入SWC~A后再在未CLR#下改变SWC~A会使得控制台命令出错。详情可以参看《设计说明书》中的说明，那儿记录下了con设计的诞生

signal con01,consw:std_logic;  --con01用来控制con(0)从0跳至1，当且仅当con='0000','0001','0010','0011','0100'且W3有效的时候为1，下降沿有效；consw用来控制con(1~3)何时读写SWC~A，理论上当且仅当con='01X1'且W1有效的时候为1，下降沿有效。注意：con01,consw都跟时序无关，没有存储，不是寄存器。

begin

--先是各种输出信号以及con01,consw等非时序信号的逻辑表达式。所有的逻辑表达式都经过了卡诺图化简。卡诺图主要对于在一定的节拍下，控制台的大BOSS--con的状态所进行的控制，与程式指令执行系统中指令IR(7~4)所进行的控制，进行了化简。所得结果用con='XXXX',IR='XXXX'（7~4）表示。本段的注释就写下卡诺图化简后的结果。卡诺图化简过程，请参阅《卡诺图化简》中的记录

--SKIP仅当，W1时，控制台中con=X1XX OR XX01 OR XX10 时有效。

SKIP<=(con(1)or(not con(2)and con(3))or(not con(3)and con(2)))and W1;

--TJ仅当，控制台中，W1时，con=XX10，W2时，con=XXX1，W3时，con=X1XX OR XXX1，以及，con=1000之RP中，W3时，IR=111X 时有效。

TJ<=(con(2)and not con(3)and W1)or(con(3)and W2)or((con(1)or con(3)or(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and IR(6)and IR(5)))and W3);

--RAM_BUS#仅当，控制台中，W1时，con=XX10，以及，con=1000之RP中，W3时，IR=1011 时有效。

RAM_BUSJ<=not(con(2)and not con(3)and W1)and not(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and not IR(6)and IR(5)and IR(4)and W3);

--ALU_BUS#仅当，con=1000之RP中，W3时，IR=01XX OR 0XX1 OR X010 OR 100X 时有效。

ALU_BUSJ<=not(con(0)and not con(1)and not con(2)and not con(3)and((not IR(7)and IR(6))or(not IR(7)and IR(4))or(IR(7)and not IR(6)and not IR(5))or(not IR(6)and IR(5)and not IR(4)))and W3);

--RS_BUS#仅当，控制台中，W3时，con=11XX，以及，con=1000之RP中，W2时，IR=101X，W3时，IR=0000 OR 1100 OR 1101C OR 1111 时有效。
```

RS_BUSJ<=not(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and not IR(6)and IR(5) and W2)and not(((con(0)and con(1))or(con(0)and not con(1)and not con(2)and not con(3)and((not IR(7)and not IR(6)and not IR(5)and not IR(4))or(IR(7)and IR(6)and not IR(5)and not IR(4))or(IR(7)and IR(6)and not IR(5)and IR(4)and C)or(IR(7)and IR(6)and IR(5)and IR(4)))))and W3);

--SW_BUS#仅当，控制台中，W1时，con=11XX OR 1XX1，W1时，con=00XX OR 0XX0 OR X011 时有效。

SW_BUSJ<=not(((con(0)and con(1))or(con(0)and con(3)))and W1)and not(((not con(0)and not con(1))or(not con(0)and not con(3))or(not con(1)and con(2)and con(3)))and W3);

--LDRi仅当，控制台中，W3时，con=1X11，以及，con=1000之RP中，W3时，IR=0XXX OR X00X OR X0X1 时有效。

LDRi<=((con(0)and con(2)and con(3))or(con(0)and not con(1)and not con(2)and not con(3)and(not IR(7)or(not IR(6)and not IR(5))or(not IR(6)and IR(4))))and W3;

--LDIR仅当，控制台中，W1时，con=X000，W2时，con=XX1X，W3时，con=11XX 时有效。

LDIR<=(not con(1)and not con(2)and not con(3)and W1)or(con(2)and W2)or(con(0)and con(1)and W3);

--LDAR#仅当，控制台中，W3时，con=00X1 OR 0X10 OR 01X0，以及，con=1000之RP中，W2时，IR=101X 时有效。

LDARJ<=not(((not con(0)and not con(1)and con(3))or(not con(0)and con(1)and not con(3))or(not con(0)and con(2)and not con(3)))and W3)and not(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and not IR(6)and IR(5)and W2);

--AR+1仅当，控制台中，W3时，con=1X01 OR 1X10 时有效。

ARpp<=((con(0)and not con(2)and con(3))or(con(0)and con(2)and not con(3)))and W3;

--LDPC#仅当，控制台中，W3时，con=0X00 OR 0011，以及，con=1000之RP中，W3时，IR=1100 OR 1101C 时有效。

LDPCJ<=not(((not con(0)and not con(2)and not con(3))or(not con(0)and not con(1)and con(2)and con (3))or(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and IR(6)and((not IR(5)and not IR(4))or(not IR(5)and IR(4)and C))))and W3);

--PC+1仅当，con=1000之RP中，W2时，IR=1101，W3时，IR=0XXX OR X0XX OR XX1X 时有效。

PCpp<=(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and IR(6)and not IR(5)and IR(4)and W2)or(con(0)and not con(1)and not con(2)and not con(3)and(not IR(7)or not IR(6)or IR(5))and W3);

--LDDR1仅当，con=1000之RP中，W2时，IR=0XX1 OR 01XX OR X010 时有效。

LDDR1<=con(0)and not con(1)and not con(2)and not con(3)and((not IR(7)and IR(6))or(not IR(7)and IR(4))or(not IR(6)and IR(5)and not IR(4)))and W2;

--LDDR2仅当，con=1000之RP中，W2时，IR=00X1 OR 001X OR 0100 时有效。

LDDR2<=con(0)and not con(1)and not con(2)and not con(3)and((not IR(7)and not IR(6)and IR(4))or(not IR(7)and not IR(6)and IR(5))or(not IR(7)and IR(6)and not IR(5)and not IR(4)))and W2;

--CER#仅当，控制台中，W1时，con=X000，W2时，con=XX1X，W3时，con=11XX 时有效。

CERJ<=not(not con(1)and not con(2)and not con(3)and W1)and not(con(2)and W2)and not(con(0)and con(1)and W3);

--CEL#仅当，控制台中，W1时，con=11XX OR 1XX1 OR 1X1X，以及，con=1000之RP中，W3时，IR=101X 时有效。

CELJ<=not(((con(0)and con(1))or(con(0)and con(2))or(con(0)and con(3)))and W1)and not(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and not IR(6)and IR(5)and W3);

--LRW#仅当，控制台中，W1时，con=11XX OR 1XX1，以及，con=1000之RP中，W3时，IR=1010 时有效。

LRWJ<=not(((con(0)and con(1))or(con(0)and con(3)))and W1)and not(con(0)and not con(1)and not con(2)and not con(3)and IR(7)and not IR(6)and IR(5)and not IR(4)and W3);

--S3仅当，con=1000之RP中，IR=X100 OR XX11 OR X0X1 OR 1X1X 时有效。

S3<=con(0)and not con(1)and not con(2)and not con(3)and((IR(6)and not IR(5)and not IR(4))or(not IR(6)and IR(4))or(IR(5)and IR(4))or(IR(7)and IR(5)));

--S2仅当，con=1000之RP中，IR=X100 OR X010 OR 1XX1 OR X111 时有效。

S2<=con(0)and not con(1)and not con(2)and not con(3)and((IR(6)and not IR(5)and not IR(4))or(IR(6)and IR(5)and IR(4))or(not IR(6)and IR(5)and not IR(4))or(IR(7)and IR(4)));

--S1仅当，con=1000之RP中，IR=XX00 OR XX11 OR X01X 时有效。

S1<=con(0)and not con(1)and not con(2)and not con(3)and((not IR(5)and not IR(4))or(IR(5)and IR(4))or(not IR(6)and IR(5)));

--S0仅当，con=1000之RP中，IR=XX11 OR 1XX0 OR 00X1 时有效。

S0<=con(0)and not con(1)and not con(2)and not con(3)and((not IR(7)and not IR(6)and IR(4))or(IR(5)and IR(4))or(IR(7)and not IR(4)));

--M仅当，con=1000之RP中，IR=1XXX OR X10X OR X011 时有效。

M<=con(0)and not con(1)and not con(2)and not con(3)and((not IR(6)and IR(5)and IR(4))or(IR(6)and not IR(5))or IR(7));

--Cn#仅当，con=1000之RP中，IR=XXX0 时有效。

CnJ<=not(con(0)and not con(1)and not con(2)and not con(3)and not IR(4));

```
--con01仅当，控制台中，W1时，con=00XX OR 0XX0 时有效。采用其下降沿的时候，把con(0)从0置1。
con01<=((not con(0)and not con(1))or(not con(0)and not con(3)))and W3;
--consw仅当，控制台中，W1时，con=X1X1 时有效。采用其下降沿的时候，把con(1~3)与SWC~A同步。
consw<=con(1)and con(3)and W1;
--kami(0~3)输出管脚，用来查看con(0~3)的值，仅仅是方便调试而已。
kami(0)<=con(0);
kami(1)<=con(1);
kami(2)<=con(2);
kami(3)<=con(3);
--kami(4)输出管脚，用来查看con01的值，仅仅是方便调试而已。
kami(4)<=con01;
--kami(5)输出管脚，用来查看consw的值，仅仅是方便调试而已。
kami(5)<=consw;


--进入时序，con01,consw,CLR#敏感。
PROCESS(con01,consw,CLRJ)
BEGIN
 IF(CLRJ='0')THEN  --当按下CLR#的时候，CLRJ=0，此时需要让con(0~3)跳入01X1这个初始的特殊状态，因为只有它之后的节拍才有consw=1，才有机会让con(1~3)与SWC~A同步。
  con(0)<='0';con(1)<='1';con(3)<='1';  --让con(0~3)跳入01X1这个特殊状态。
  ELSE  --当没有按CLR#的时候，才进行下面语句，从而明主次，定尊卑，使模块所发命令有定。
  IF(con01'event AND con01='0')THEN  --当con01处在下降沿，根据设计，需要将con(0)从0置1。
    con(0)<='1';  --将con(0)置1。
  END IF;
  IF(consw'event AND consw='0')THEN  --当consw处在下降沿，根据设计，需要将con(1~3)与SWC~A同步。因为这个修改的与前面那个IF语句修改的目标不同，从而可以并列。
  con(1)<=SWC;con(2)<=SWB;con(3)<=SWA;  --将con(1~3)与SWC~A同步。
  else null;  --当非下降沿的时候就不同步。
  END IF;
 end if;
END PROCESS;
end Operation;
```

至于原理图，就参看附录《设计说明书》吧。

上一页　　下一页

返回