東南大學

计算机系统综合设计

设计报告

组长：**张冠群（09003105）**

成员：**杨　俊（09003209）**

孙啸寅（09003112）

万乾坤（09003218）

李传佑（09003214）

刘　森（09003110）

黄　河（09003206）

东南大学计算机科学与工程学院

二 OO 六年九月

设计名称	基于 MIPS32 的 SOC 设计		
完成时间	2006.9	验收时间	
本组成员情况			

姓　名	学　号	承　担　的　任　务	成 绩
张冠群	09003105	**mips32CPU、CTC、PWM、WDT、BIOS、实验报告、最终测试**	
刘　森	09003110	LED、LCD	
孙啸寅	09003112	mips32CPU、I/O 模块	
黄　河	09003206	UART	
杨　俊	09003209	mips32CPU、INT32、SOC 模块	
李传佑	09003214	KEY，部分 BIOS	
万乾坤	09003218	Compilers	

注：本设计报告中各个部分如果页数不够，请大家自行扩页，原则是一定要把报告写详细，能说明本组设计的成果和特色，能够反应小组中每个人的工作。报告中应该叙述设计中的每个模块。设计报告将是评定每个人成绩的一个重要组成部分。

本组设计的功能描述（含所有实现的模块的功能）

SOC 模块

串联 CPU 和外围接口模块，实现片上系统的功能。

MEMorIO 模块

该模块功能作用是给 cpu 添加 io 接口，当读写 mem 时给 RAM 输出存储器读写信号，当读写 io 时给外设输出 io 读写信号，提供数据线和地址线与外设连接.

UART 模块

简单的串行通信模块。负责控制将 CPU 来的 8 位数据并转串，然后按照异步串行通信数据格式输出，将串口来的 8 位串行数据串转并，并在 CPU 请求的时候输入给 CPU。

LED 模块

通过向该控制电路写 32 位数据，经过译码控制共阳极的 7 段 LED 显示。32 位数每半个字节控制一位 7 段 LED，从高位到地位排列。每个 LED 灯显示十六进制数，从 0 到 F。

LED0 模块

LED 的附属模块，一个 LED 灯的输出显示。

KEY 模块

自动扫描 4 × 4 的键盘，当有键盘按下的时候扫描键值，将键值记录到键值寄存器，然后向 CPU 发出中断。CPU 响应中断，当 CPU 读出键值后，撤销中断请求。

CTC 模块

定时/计数器模块。两个定时/计数器 CNT0 和 CNT1。具有计数和定时两个功能。计数方式下可以对输入的外部脉冲进行计数，当计数到初值寄存器的值的时候，设置状态寄存器的相应位。定时方式下，在时钟作用下计时器做减 1，到 0 的时候设置状态寄存器的相应位，并在相应的 COUT 脚输出一个时钟的低电平（平时 COUT 是高电平）。状态寄存器在被读取后被清零。

PWM 模块

脉冲宽度调制模块。可以调节脉冲的频率和占空比（本模块不支持）。一个 12 位 PWM 内部一个计数器和一个对比值，计数器周而复始的加 1 计数，计数到 0FFFH 的时候转为 0 再计数。当计数器的值大于对比值，输出端输出低电平，否则输出高电平。

WDT 模块

看门狗模块。内含一个 16 位定时器，系统复位后计数值为 FFFFH，之后每时钟计数值减 1，当减到 0 的时候，向 CPU 发 4 个时钟周期的 RESET 信号，同

时计数值恢复到 FFFFH 并继续计数。通过软件不断地定期写看门狗端口来复位看门狗，使计数器重新从 FFFFH 开始计数。增加看门狗电路后，CPU 的 RESET 输入脚是系统复位信号和看门狗发出的复位信号的组合。

mips32 模块

mips32CPU 顶层模块。负责串联起 CPU 内各个控制、取指、执行、中断等模块。完成基于 mips 指令集的 32 位 CPU 功能。具有 32 位指令，16 位地址线和 32 位数据线。

CountClock 模块

用于串行通信 UART 的顶层模块。

control32 模块

控制单元模块。根据指令中的指令码（op）和功能码（funct）的不同组合输出相应的控制信号。

dmemory32 模块

存储单元模块。实际完成对数据存储器(RAM)的读写操作。

executs32 模块

执行单元模块。完成逻辑运算、算术运算、移位运算、比较转移的 PC 值计算、比较后赋值。

idecode32 模块

译码单元模块。对寄存器组进行操作。根据指令译码结果，决定向其他部件（如运算器）送 1 路或两路数据。

ifetc32 模块

取指单元模块。到程序 ROM 中取指令、对 PC 值进行 + 4 处理、完成各种跳转指令的 PC 修改功能。

int32 模块

中断模块。具有两个中断功能——中断 0 和中断 1，中断 0 优先级大于中断 1，不支持同级中断嵌套。将$12 和$14 设计为两个中断返回地址存放寄存器，分别在中断 0 和中断 1 到来的时候存放返回地址。它们分别称为$i0 和$i1。

timecount 模块

时钟脉冲分频模块。共分为四个脉冲，每个脉冲的频率相同，且均为系统时钟频率的 4 倍。不同的脉冲用以控制不同的 CPU 内部模块的运行。

cs138 模块

译码模块。用于产生各个接口模块的片选信号。

本组设计的主要特色

1、可运行指定的 31 条 MIPS 指令的 RISC 型 MIPS32 微处理器，具有 32 位指令，16 位地址线和 32 数据线。

2、处理器采用哈佛结构，有独立的 2KB 的指令存储器和 4KB 的数据存储器具有 2 个中断源入口，两级中断优先级。

3、键盘采用中断方式，而非扫面方式，为中断 0 级（最高级）。

4、2 个 32 位定时/计数器。

5、CPU 内部采用分频的方法，用不同的时钟对不同的模块进行同步控制，从而提高了主频。

6、键盘和 LCD 的时钟由 CTC 提供，初始化程序在 BIOS 中。

7、具有看门狗功能，通过指令对 WDT 进行复位。

8、MEM 和 IO 统一编址。IO 的高 3 位产生片选 CS 信号，故最多支持 8 个接口，低 5 位用来选择 8 个 32 位的端口。

9、附有类 C 语言编译器，详细见 MIPS32 编译程序使用手册。

10、提供有 LCD 接口电路。

11、CPU 与接口之间的数据线有 64 条，用于 CPU 输出和读入数据。

12、CPU 内的 ROM 和 RAM 均采用同步控制。

13、采用 32 位的 mips 汇编 BIOS 程序。

14、编写有 RAM 和内部寄存器使用规则，避免因用户随意修改 RAM 和寄存器内存放的值而导致的系统瘫痪。

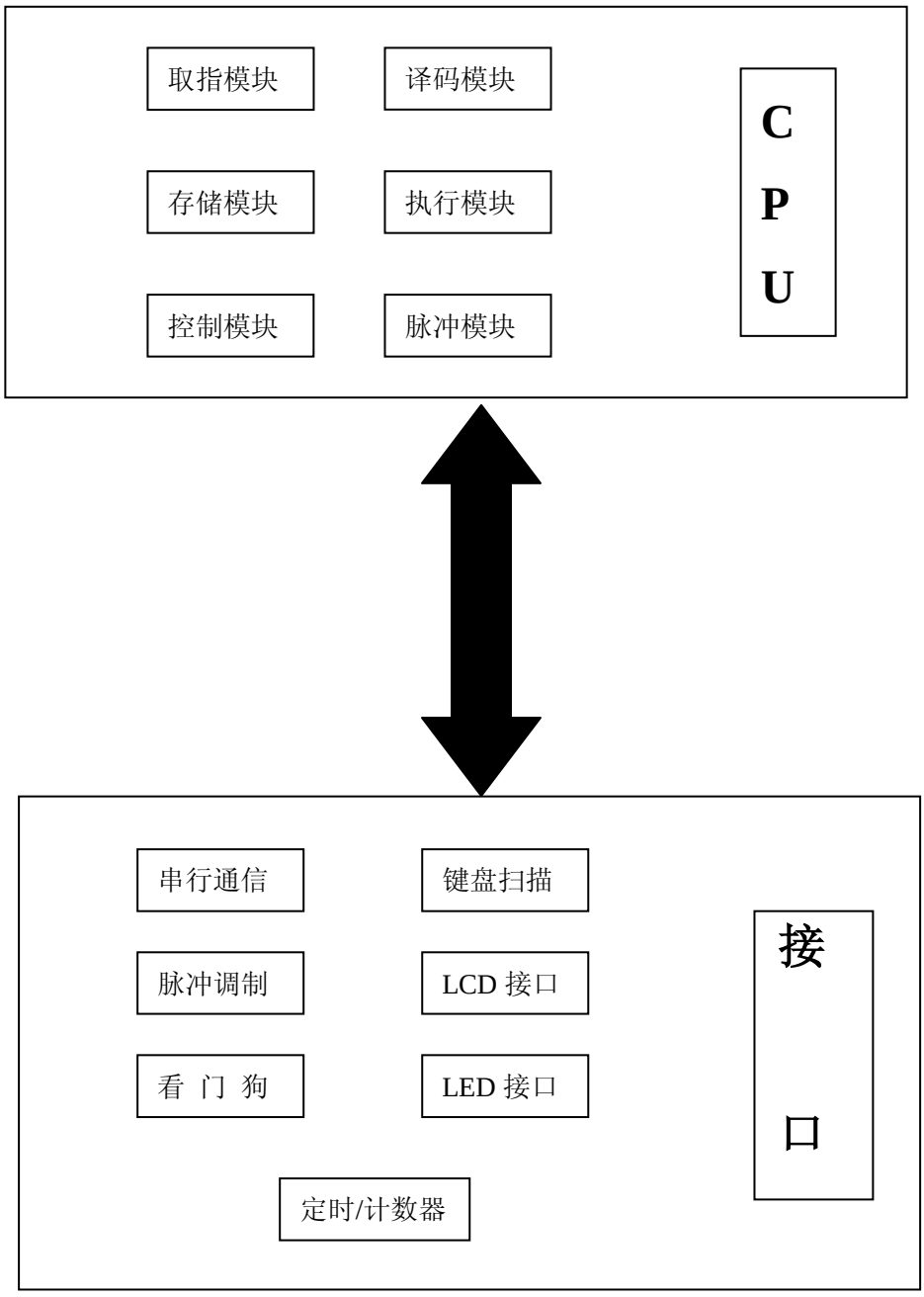
15、规定了用户用 mips 汇编写代码时可用的寄存器和 RAM 地址空间。

16、按住键盘上的某个键，在屏幕上并不是连续显示键值，两个前后显示的键值之间会有一定的时间间隔。

17、当编译用户的 C 程序时，如果堆栈指针越界，编译程序会提示出错，并在 LED 上显示三条横线。

18、最多支持 7 个 LED 的显示。

本组设计的体系结构


取指模块
译码模块
存储模块
执行模块
控制模块
脉冲模块
CPU
串行通信
键盘扫描
脉冲调制
LCD 接口
看 门 狗
LED 接口
定时/计数器
接口

本组设计中各个部件的设计与特色

CPU

由六大模块构成，分别是取指模块、控制模块，译码模块，存储模块，执行模块，脉冲分频模块。可运行指定的 31 条 MIPS 指令的 RISC 型 MIPS32 微处理器，具有 32 位指令，16 位地址线和 32 数据线。处理器采用哈佛结构，有独立的 2KB 的指令存储器和 4KB 的数据存储器。具有 2 个中断源入口，两级中断优先级。采用同步的 RAM 和 ROM 代替异步。

CTC

定时/计数器模块。两个定时/计数器 CNT0 和 CNT1。具有计数和定时两个功能。计数方式下可以对输入的外部脉冲进行计数，当计数到初值寄存器的值的时候，设置状态寄存器的相应位。定时方式下，在时钟作用下计时器做减 1，到 0 的时候设置状态寄存器的相应位，并在相应的 COUT 脚输出一个时钟的低电平（平时 COUT 是高电平）。状态寄存器在被读取后被清零。

PWM

脉冲宽度调制模块。可以调节脉冲的频率和占空比（本模块不支持）。一个 12 位 PWM 内部一个计数器和一个对比值，计数器周而复始的加 1 计数，计数到 0FFFH 的时候转为 0 再计数。当计数器的值大于对比值，输出端输出低电平，否则输出高电平。

WDT

看门狗模块。内含一个 16 位定时器，系统复位后计数值为 FFFFH，之后每时钟计数值减 1，当减到 0 的时候，向 CPU 发 4 个时钟周期的 RESET 信号，同时计数值恢复到 FFFFH 并继续计数。通过软件不断地定期写看门狗端口来复位看门狗，使计数器重新从 FFFFH 开始计数。增加看门狗电路后，CPU 的 RESET 输入脚是系统复位信号和看门狗发出的复位信号的组合。

UART

串行通信模块。发送器要对外部时钟 XTAL 按照波特率要求进行分频，本设计采用固定波特率 4800b/s。接收器接收数据的采用率是波特率的 16 倍（也由 XTAL 分频得到）。采用中间值采用的原则，也就是一个数据位占 16 个接收时钟宽度，在第 8 个时钟的时候采样。数据格式固定为一位起始位 0,16 位数据位,暂无校验位,1 位停止位 1,若干空闲位 1 串行输出线空闲状态为 1。

端口列表:

100:接收移位寄存器

101:发送移位寄存器

010:写入缓冲寄存器

011:内部状态寄存器

KEY

键盘扫描模块。硬件实现自动扫描 4 × 4 的键盘，当有键盘按下的时候扫描键值，将键值记录到键值寄存器，并置位状态寄存器中的“有键”标志，同时发出中断。当 CPU 读出键值后，将“有键”标志清除。采用中断的方式而不采用 CPU 扫描的方式是本设计的特色，如果长时间不按键的话可以节省很多 CPU 时间，所以这样做提高了 CPU 的工作效率。

LED

LED 灯控制模块。设计：32 位输入数据，每半个字节控制一位七段 LED，由高位到低位排列。由两个模块控制 4 个 LED 灯。

LCD

LCD 显示器接口模块。可显示 5*7 或 5*10 点字图形 20 个共 2 行，因此共可显示 40 个字图形。内部显示寄存器有 20*32bit=20 个，每一行 10 个，可用位移显示法予以显示。

本组设计的 MIPS32 编译程序使用手册

编写程序的语言的语法说明:

1、允许全局变量和函数局部变量，但一个函数内的变量在同一层次;

2、所有标识符只能是以字母打头，数字字母串，不能含有‘_’;

3、立即数可以是十进制或是以“0x”打头的十六进制;

4、循环语句允许有 for、while、do...while，可以用 continue、break;

5、不支持 goto 语句;

6、条件语句可以用 if...else...，不能用 switch...case...;

7、变量最多允许为一维，类型只能为有符号整型，不支持变量初始化;

8、函数可以有申明，返回类型为 void 或 int，可以嵌套、递归调用;

9、允许空语句;

10、算术操作符有+、-、*、/、%、&（按位与）、|（按位或）、^（按位异与）、<<（左移）、>>（右移）、.（取变量的第几位，返回 0、1）、-（负号）;

11、逻辑操作符有&&、||、!;

12、关系运算符有<、>、<=、>=、!=、==;

13、支持变量后++，变量后--;

14、对端口操作可以用在端口号地址前加‘$’;

15、可以没有 main()函数，中断处理函数的名称分别为“interuptServer0”和“interuptServer1”;

16、不允许用头文件，所有程序只能放在同一个文件中;

17、BIOS 功能调用的函数名:

该程序生成的中间代码类似于汇编程序，但有所不同：

1、每条语句前可以有标号，必须是以‘L’打头，后面跟上十进制数；L0，L1都有特殊含义，L0是中断0跳转地址，L1是中断1跳转地址;

2、每条指令都是一个操作码加上三个操作数，中间不需要其他符号，操作数不够的后面补零;

3、只有一条伪指令“int”，相当于定义变量，可以在函数结构内定义局部变量，也可在函数外定义全局变量，以‘;’结束。操作数中出现的变量名就等价于立即数，值等于它的偏移量;

4、指令“j”“jal”“beq”“bne”后跟的是要跳转的地址处标号的值，而非跳转地址或偏移 ;

5、“lw”“sw”的后面的参数顺序与网页上的不同，将立即数放在了最后;

6、寄存器可以用‘$’后加数表示，也可以加寄存器名表示，即“$1”和“$at”等价;

7、立即数支持正、负，十进制、十六进制（以“0x”打头）;

对于关键部分的程序可以在中间代码进行人工修改后在汇编翻译。

语法和词法：

digit : [0-9]

letter : [a-zA-Z] ;

token : letter token | letter ;

decnums : digit decnums | ;

decnum : digit | [1-9] decnums ;

hexnums : [0-9a-f] hexnums | ;

hexnum : '0x' hexnums ;

number : decnum | hexnum ;

var : '$' arifif | token | token '[' ariexp ']' ;

args : ariexp ',' args | ariexp | ;

funcal : token '(' args ')' ;

arifif : var | funcal | '(' ariexp ')' | number ;

arifor : '-' arifif |arifif;

arithi : arithi '&' arifor | arithi '|' arifor | arithi '^' arifor | arithi '.' arifor | arithi '>>' arifor | arithi '<<' arifor | arifor;

arisec : arisec '*' arithi | arisec '/' arithi | arisec '%' arithi | arithi;

ariexp : ariexp '+' arisec | ariexp '-' arisec | arisec;

relation : ariexp '<' ariexp | ariexp '>' ariexp | ariexp '<=' ariexp | ariexp '>=' ariexp | ariexp '==' ariexp | ariexp '!=' ariexp |'true' ;

logthi : '!' logthi | '(' logexp ')' | relation ;

logsec : logsec '&&' logthi | logthi ;

logexp : logexp '||' logsec | logsec ;

defvar : token | token '[' number ']' ;

defvars : defvar ',' defvariables | defvar ;

defsen :'int' defvars ;

empsen : ';' ;

retsen : 'return' ariexp ;

assen : var '=' ariexp | var '++' | var '--' ;

forini : asssen | ;

forjudge : logexp | ;

fordo : asssen | ;

forsen : 'for' '(' forini ';' forjudge ';' fordo ')' block ;

whisen : 'while' '(' logexp ')' block ;

dowsen : 'do' block 'while' '(' logexp ')' ';' ;

iffsen : 'if' '(' logexp ')' block | 'if' '(' logexp ')' block 'else' block ;

sentence : empsen | asssen ';' | forsen | whisen | dowsen | iffsen | defsen | funcal ';' |'break' ';' |'continue' ';' | retsen ;

block : '{' sentence block '}' | sentence ;

fundefvars : 'int' token ',' fundefvars | 'int' token | ;

fundec : rettype token '(' fundefvars ')' ';' ;

fundef : rettype token '(' fundefvars ')' '{' block '}' ;

function : fundef | fundec ;

wsentence : defsen | function ;

program : wsentence program ;

本组设计中的 BIOS 使用手册

BIOS 功能调用:

1、KEY

入口地址 498

23 号寄存器存放键值 特权

用户从 19 号寄存器取数据

2、LED

入口地址 489

规定将要写的内容放到 20 号寄存器中

3、rUART

入口地址 478

特权寄存器 22 用来存放从 UART 读来的数据,用户从 18 号寄存器取出数据。

UART 接收了 CPU 发出的数据，当接收缓冲寄存器为空时将$24 清零

4、wUART

入口地址 467

用户用 21 号寄存器来写

21 号寄存器的内容赋给 24 号寄存器

24 号寄存器是中断保留使用的

5、WRONG（不对用户开放）

入口地址 509

使用特权寄存器$25 保存是否越界的标志

INT 中断功能:

1、KEY

中断号: 0

中断向量表地址: 510

中断服务程序首址: 36

2、UART

中断号: 1

中断向量表地址: 511

中断服务程序首址: 46

初始化程序:

1、初始化 LCD

2、初始化 KEY

本组设计中的 RAM 和寄存器使用手册

RAM 使用规则:

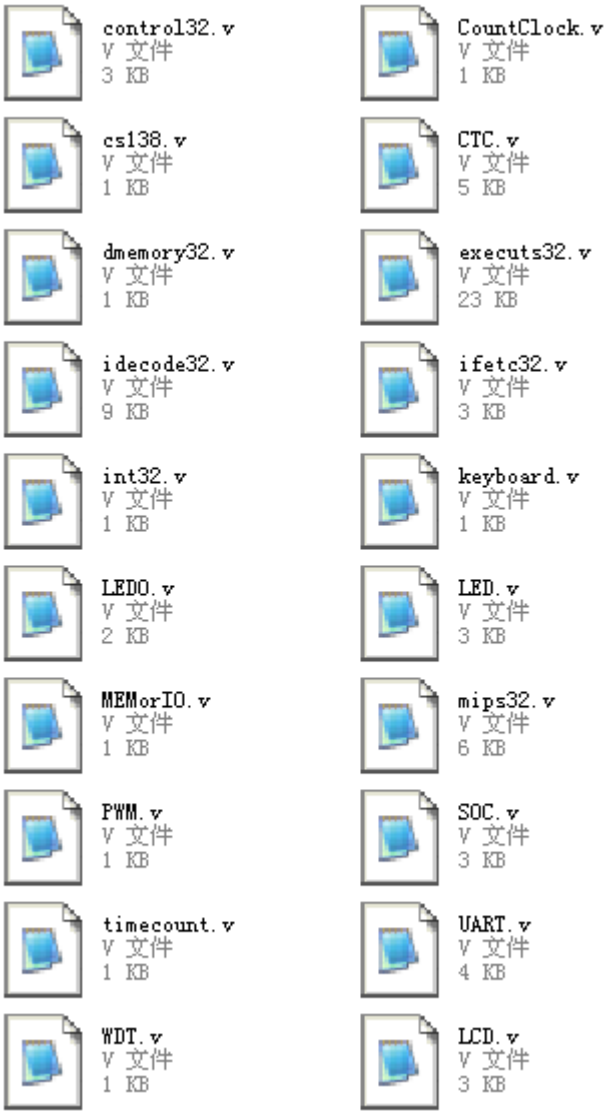
系统使用的 RAM 地址范围: 03F7~03FF

用户使用的 RAM 地址范围: 0000~03F6

寄存器使用规则:

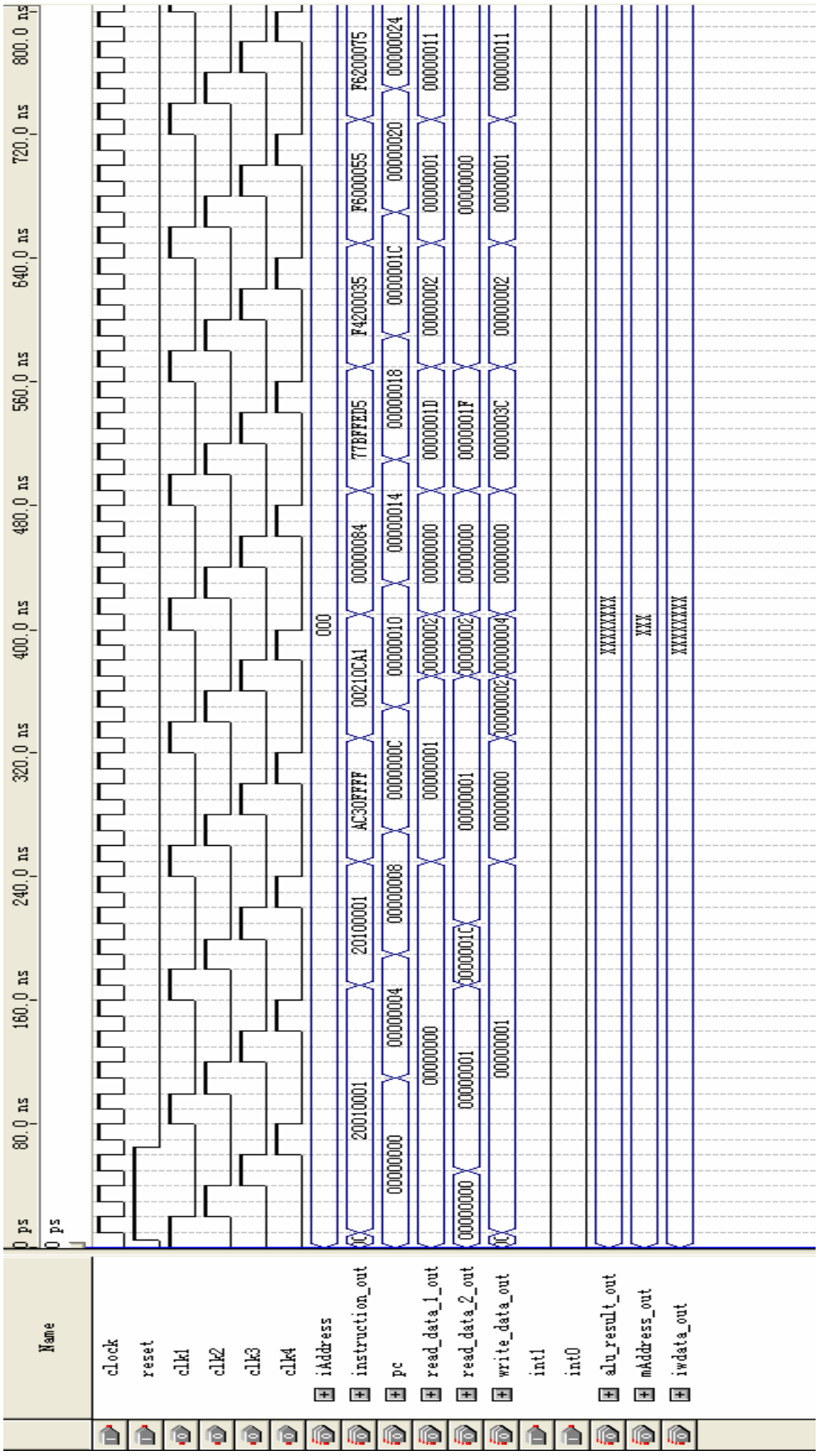
$0: 常量 0 寄存器
$1: 留做汇编使用
$2~$3: 用于表达式和函数值的存放
$4~$7: 存放参数 1~3
$8~$15: 临时变量寄存器，调用时不用保存压栈的
$16~$21: 变量寄存器，需要压栈保存。
    $18 用户从该寄存器里读 UART 的内容;
    $19 用户从该寄存器里读键盘的键值;
    $20 存放用户将要写到 LED 内的内容;
    $21 存放用户将要写到 UART 内的内容
$22: 特权寄存器，中断程序用它存放从 UART 读来的数据
$23: 特权寄存器，中断程序用它存放寄存器的键值
$24: 特权寄存器，UART 中断使用
$25: 特权寄存器，判断堆栈指针是否越界时使用
$26~$27: 为操作系统内核保留
$28: 指向全局变量的寄存器
$29: 存放堆栈指针
$30: 存放中断的返回地址
$31: 存放函数调用的返回地址

本组设计中的 Verilog HDL 程序清单及 VWF 图

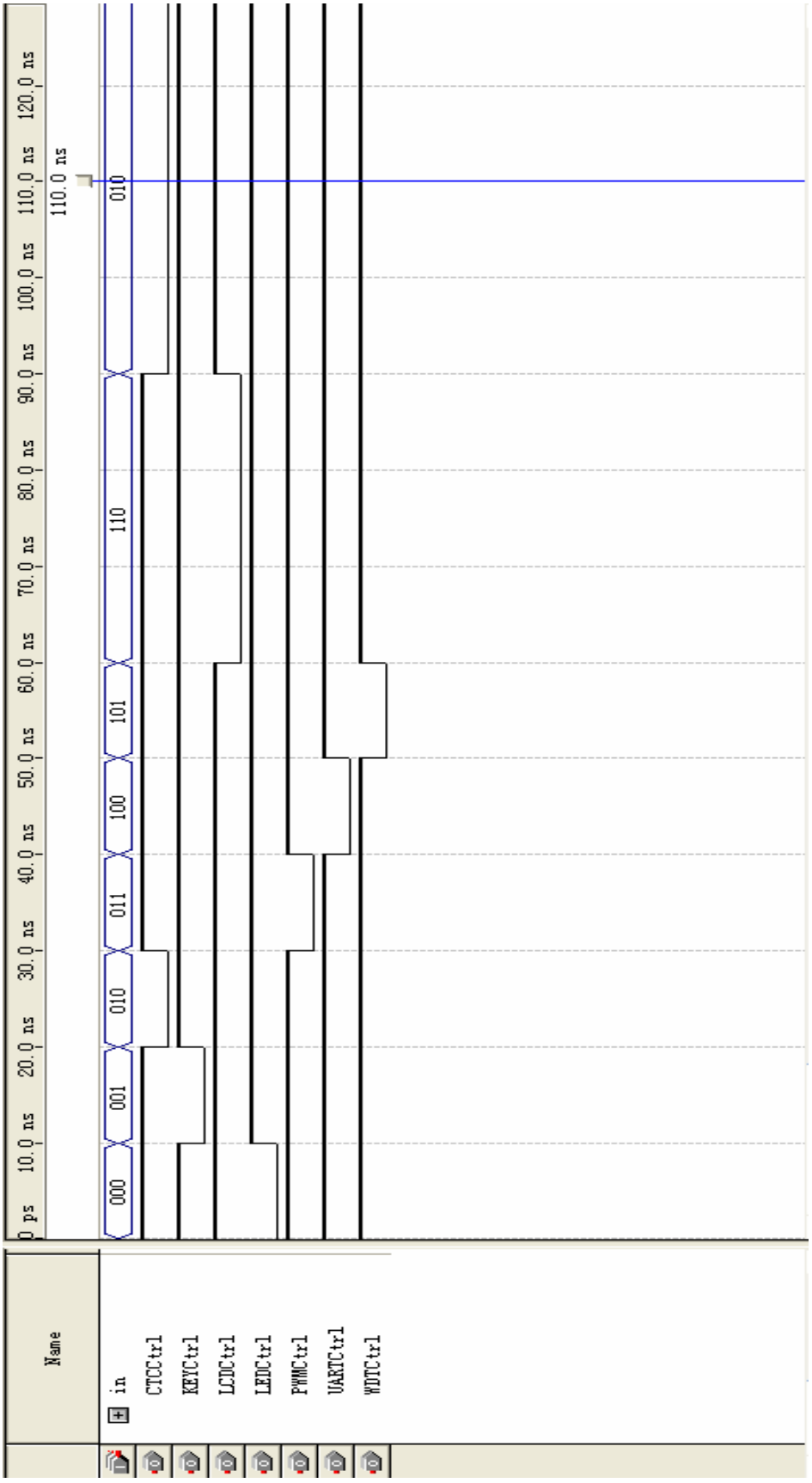


注：这里只列出文件的列表，BIOS 程序及.v 文件具体代码请参见光盘。

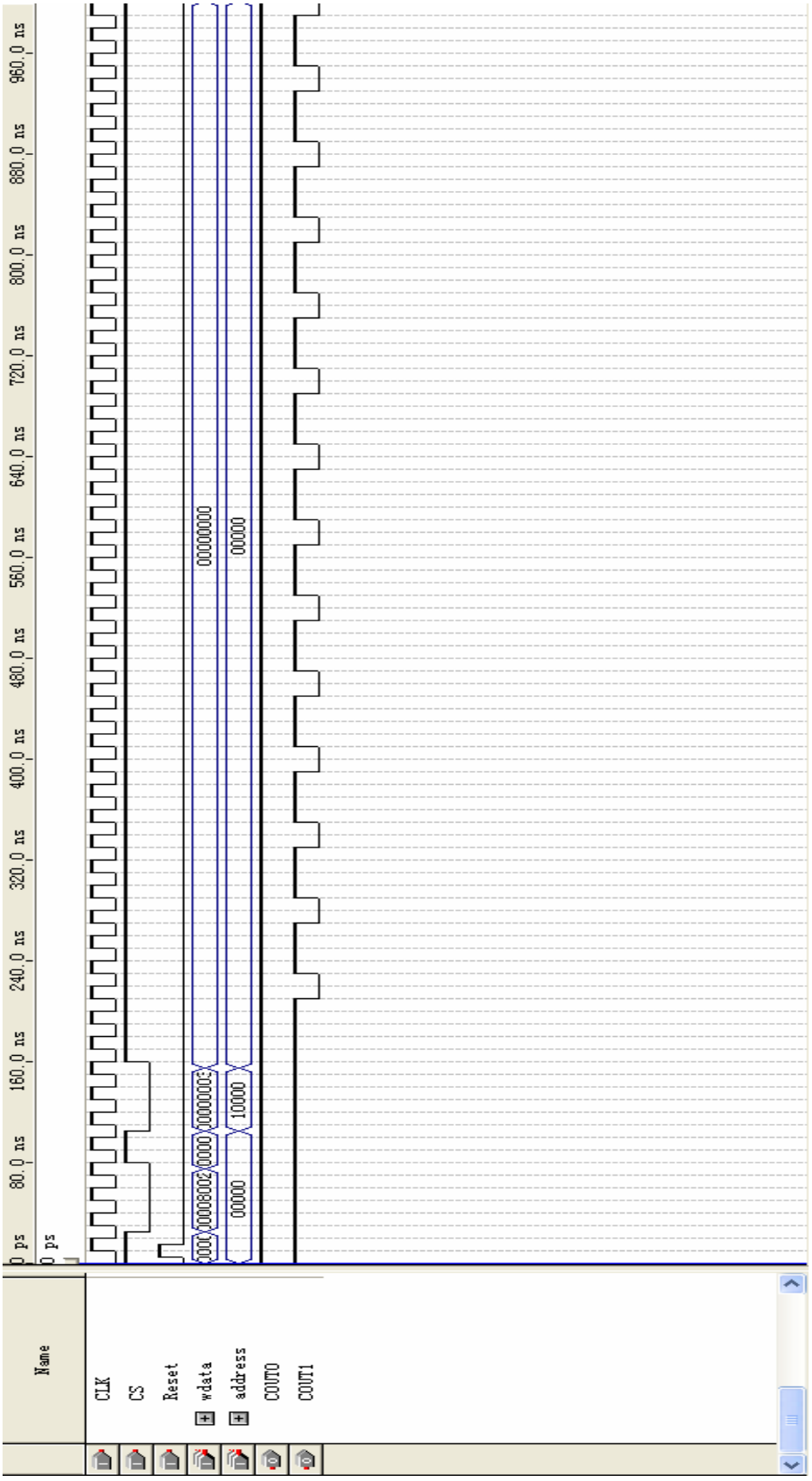
CPU 的 vwf 图


Name
clock
reset
clk1
clk2
clk3
clk4
iAddress
instruction_out
pc
read_data_1_out
read_data_2_out
write_data_out
int1
int0
alu_result_out
mAddress_out
iwdata_out
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
000
20010001
20100001
AC30FFFF
00210CA1
00000084
77BFFFD5
F4200035
F6000055
F6200075
00000000
00000004
00000008
0000000C
00000010
00000014
00000018
0000001C
00000020
00000024
00000000
00000001
00000002
00000001
00000002
00000000
00000001
00000011
00000000
00000001
0000001C
00000001
00000002
00000000
0000001F
00000000
00000000
00000001
00000000
00000002
00000004
00000000
0000003C
00000002
00000001
00000011
XXXXXXXX
XXX
XXXXXXXX

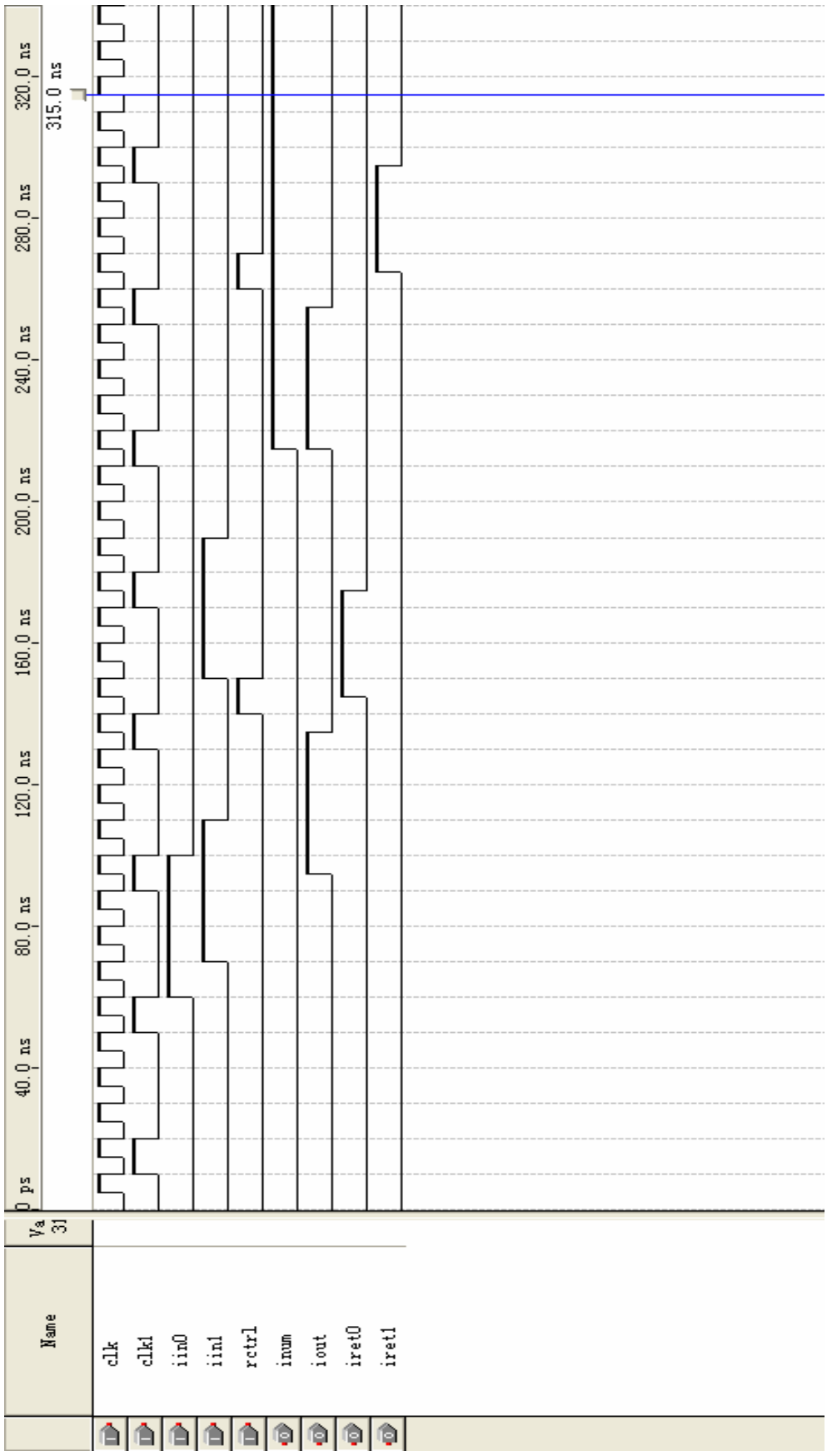
Cs138 的 vwf 图


Name
in
CTCCtrl
KEYCtrl
LCDCtrl
LEDCtrl
PWMCtrl
UARTCtrl
WDTCtrl
0 ps
10.0 ns
20.0 ns
30.0 ns
40.0 ns
50.0 ns
60.0 ns
70.0 ns
80.0 ns
90.0 ns
100.0 ns
110.0 ns
120.0 ns
110.0 ns
000
001
010
011
100
101
110
010

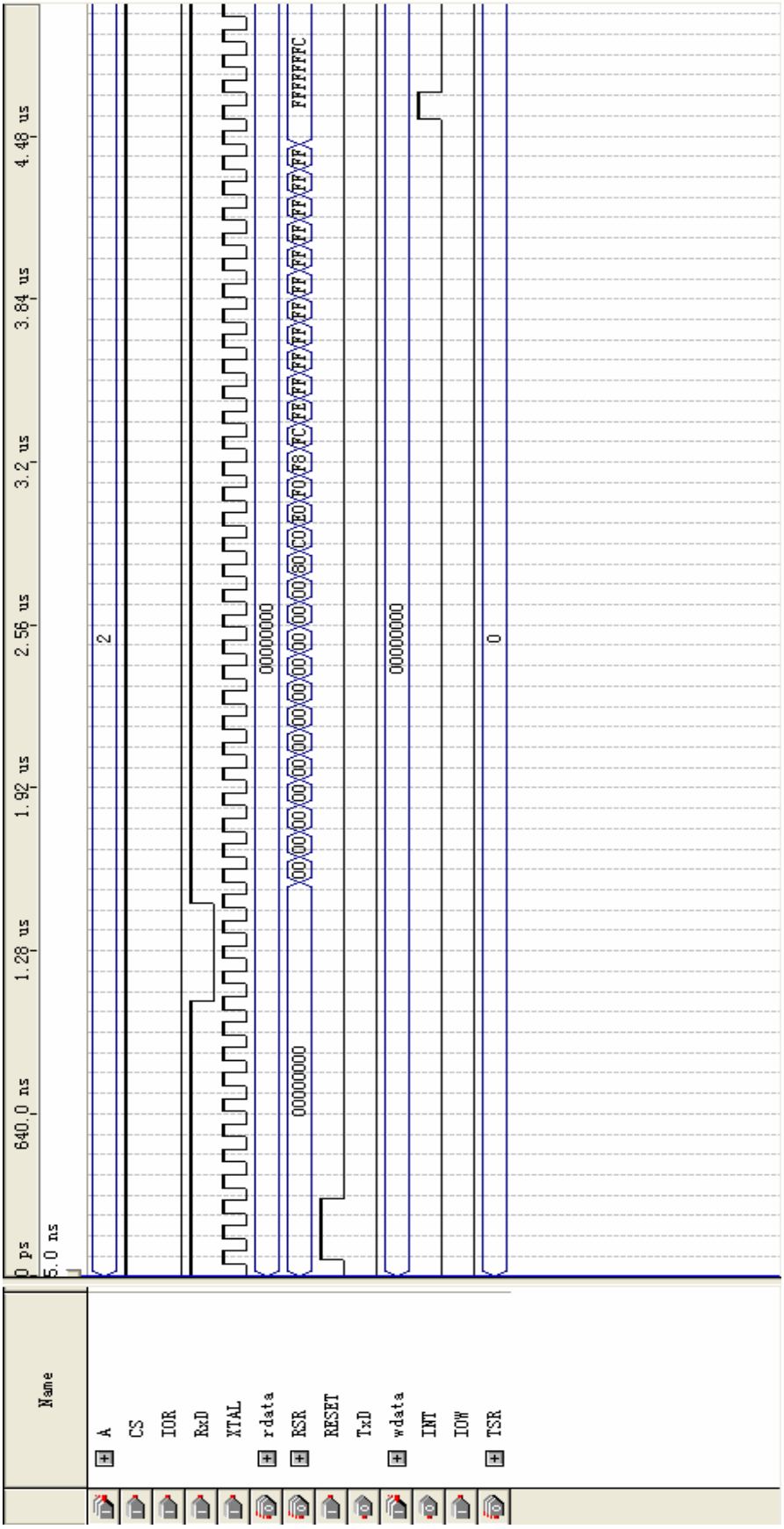
CTC 的 vwf 图


Name
CLK
CS
Reset
wdata
address
COUT0
COUT1
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
00000000
00000
10000
00000

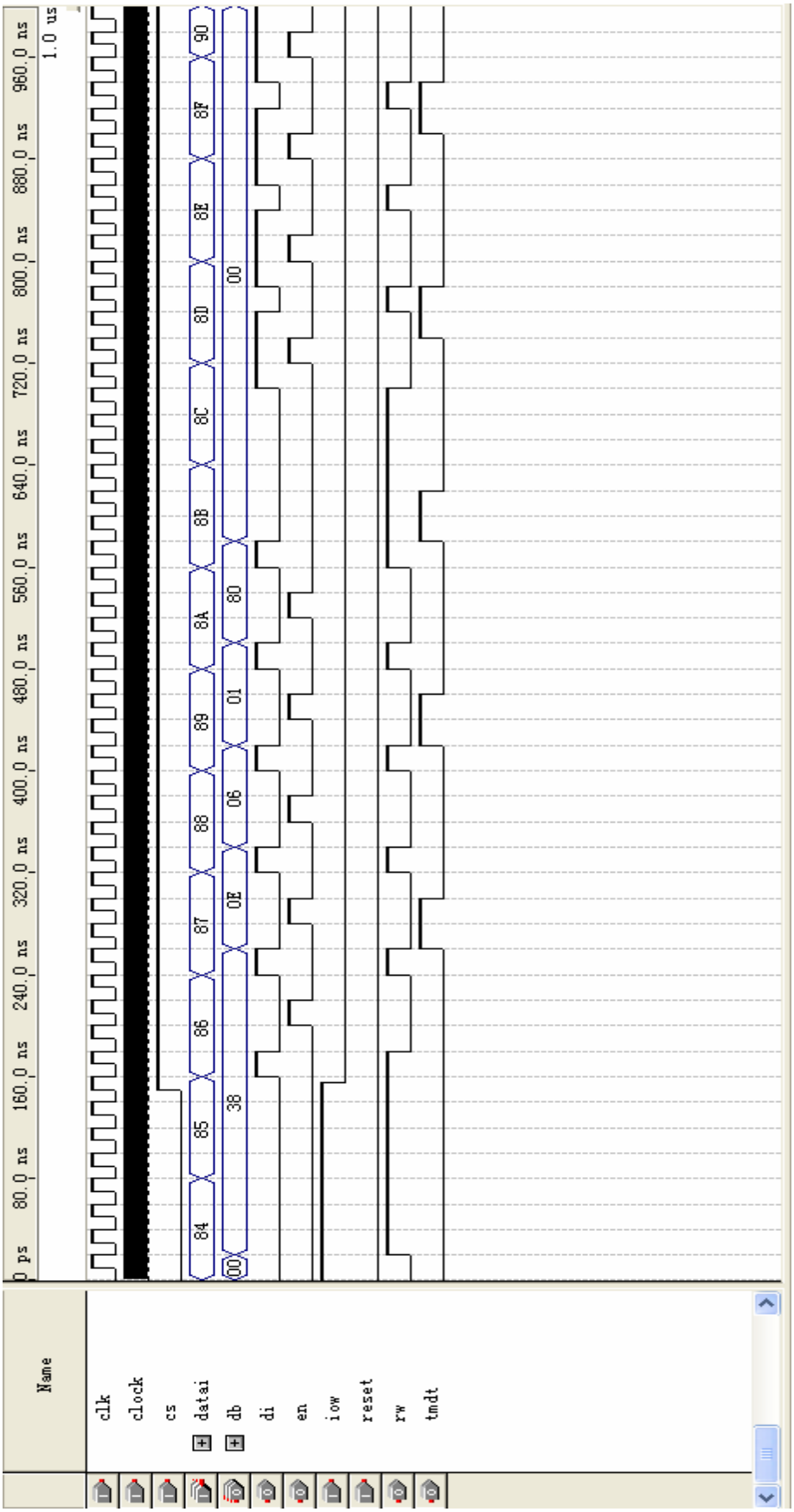
Int32 的 vwf 图


0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
315.0 ns
Name
clk
clk1
iin0
iin1
rctrl
inum
iout
iret0
iret1

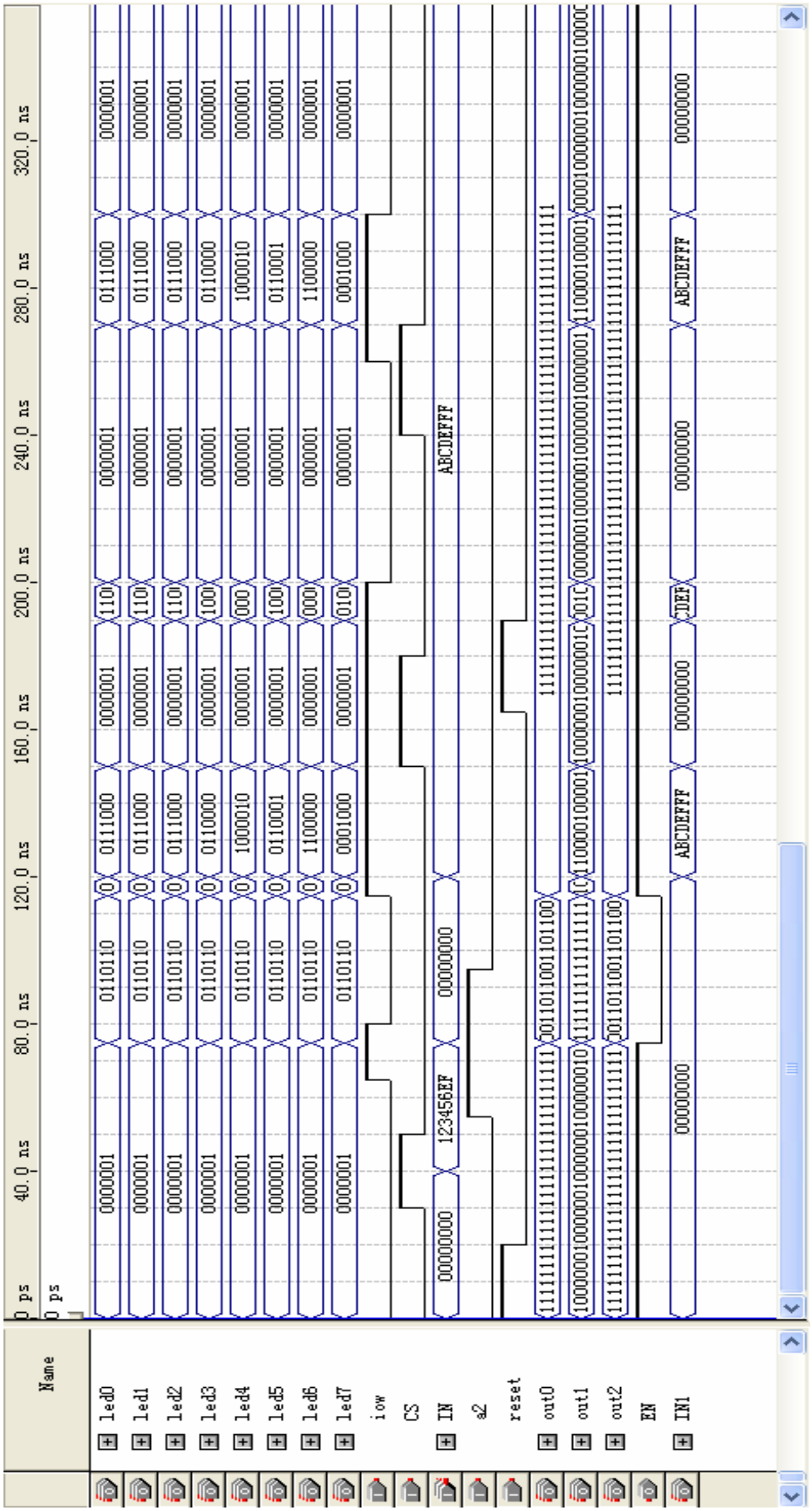
UART 的 vwf 图


Name
A
CS
IOR
RxD
XTAL
rdata
RSR
RESET
TxD
wdata
INT
IOW
TSR
0 ps
640.0 ns
1.28 us
1.92 us
2.56 us
3.2 us
3.84 us
4.48 us
5.0 ns
2
00000000
00000000
00000000
0
FFFFFFFC

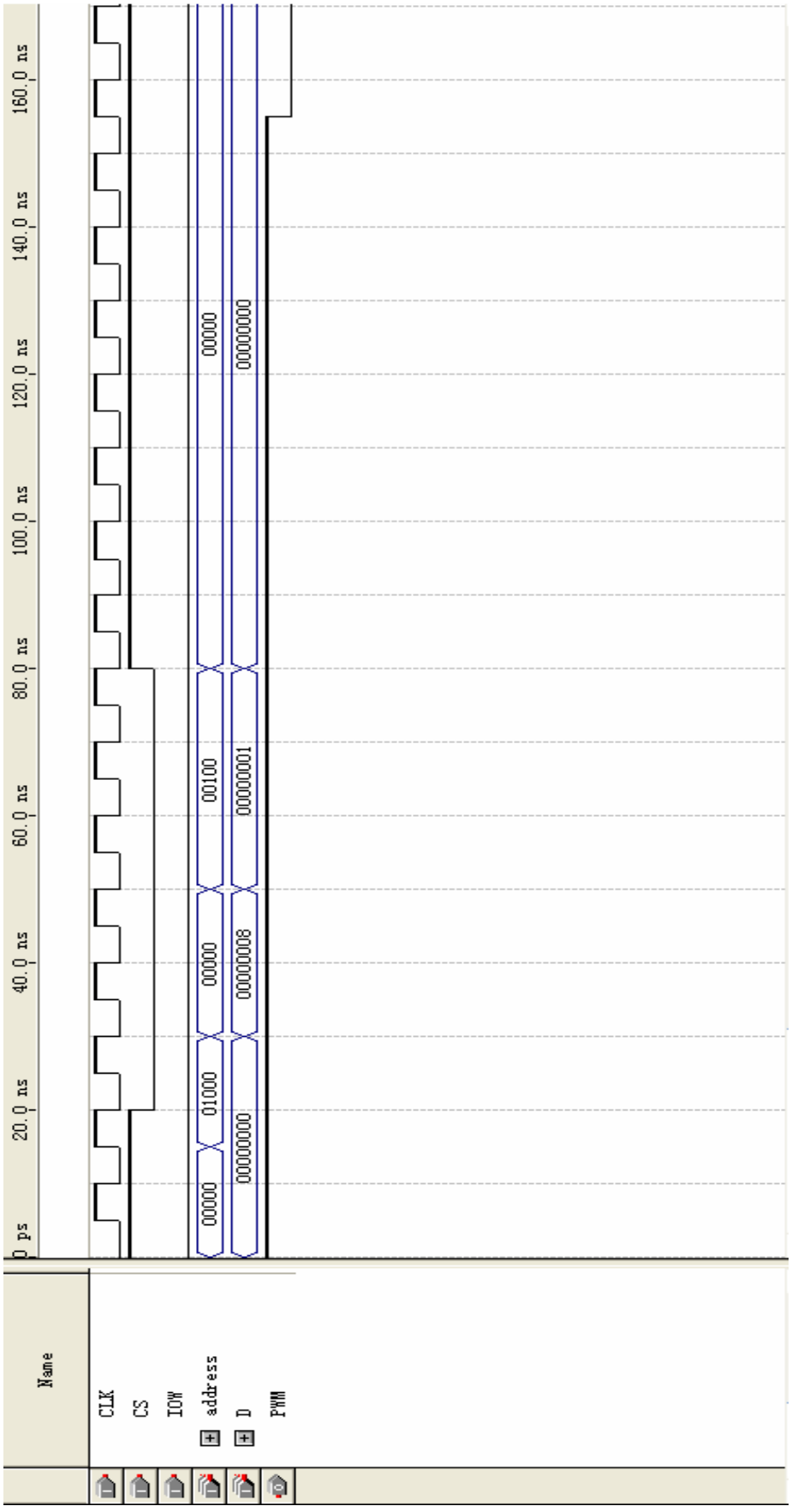
LCD 的 vwf 图


Name
clk
clock
cs
datai
db
di
en
iow
reset
rw
tmdt
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800.0 ns
880.0 ns
960.0 ns
1.0 us
00
84
85
38
86
87
3E
88
06
89
01
8A
80
8B
8C
8D
00
8E
8F
90

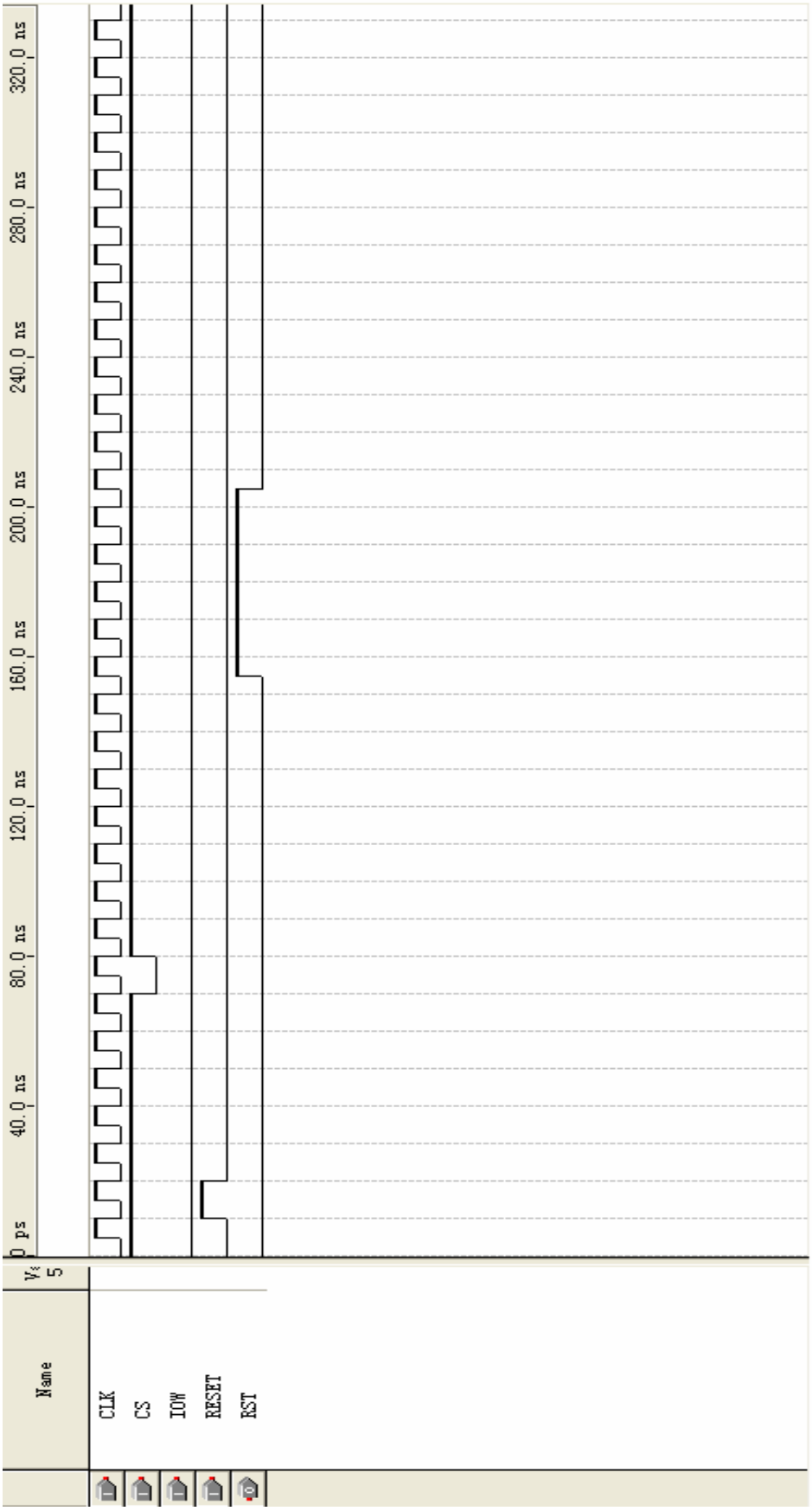
LED 的 vwf 图


Name
led0
led1
led2
led3
led4
led5
led6
led7
iow
CS
IN
a2
reset
out0
out1
out2
EN
IN1
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
123456EF
ABCDEFFF

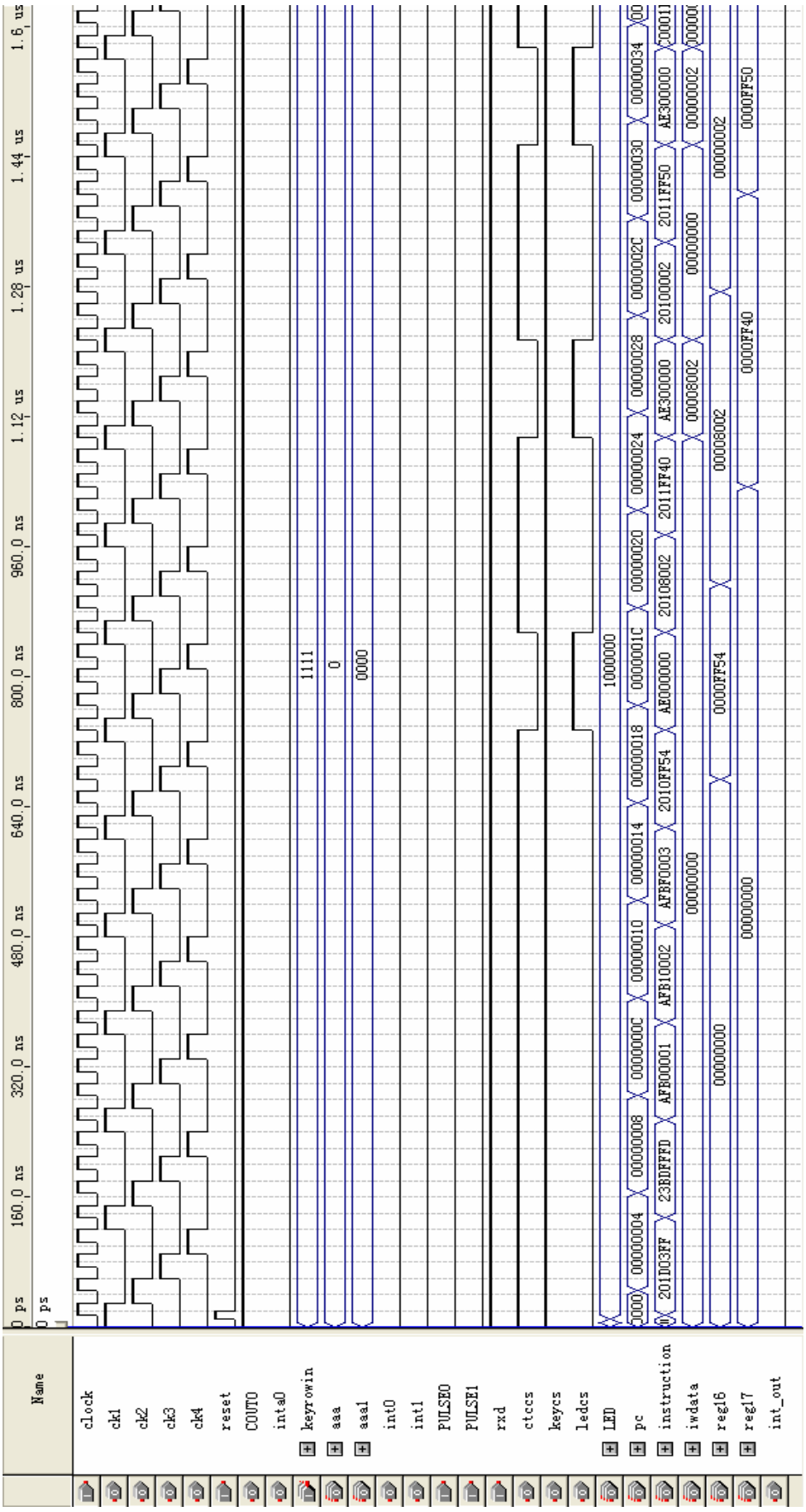
PWM 的 vwf 图

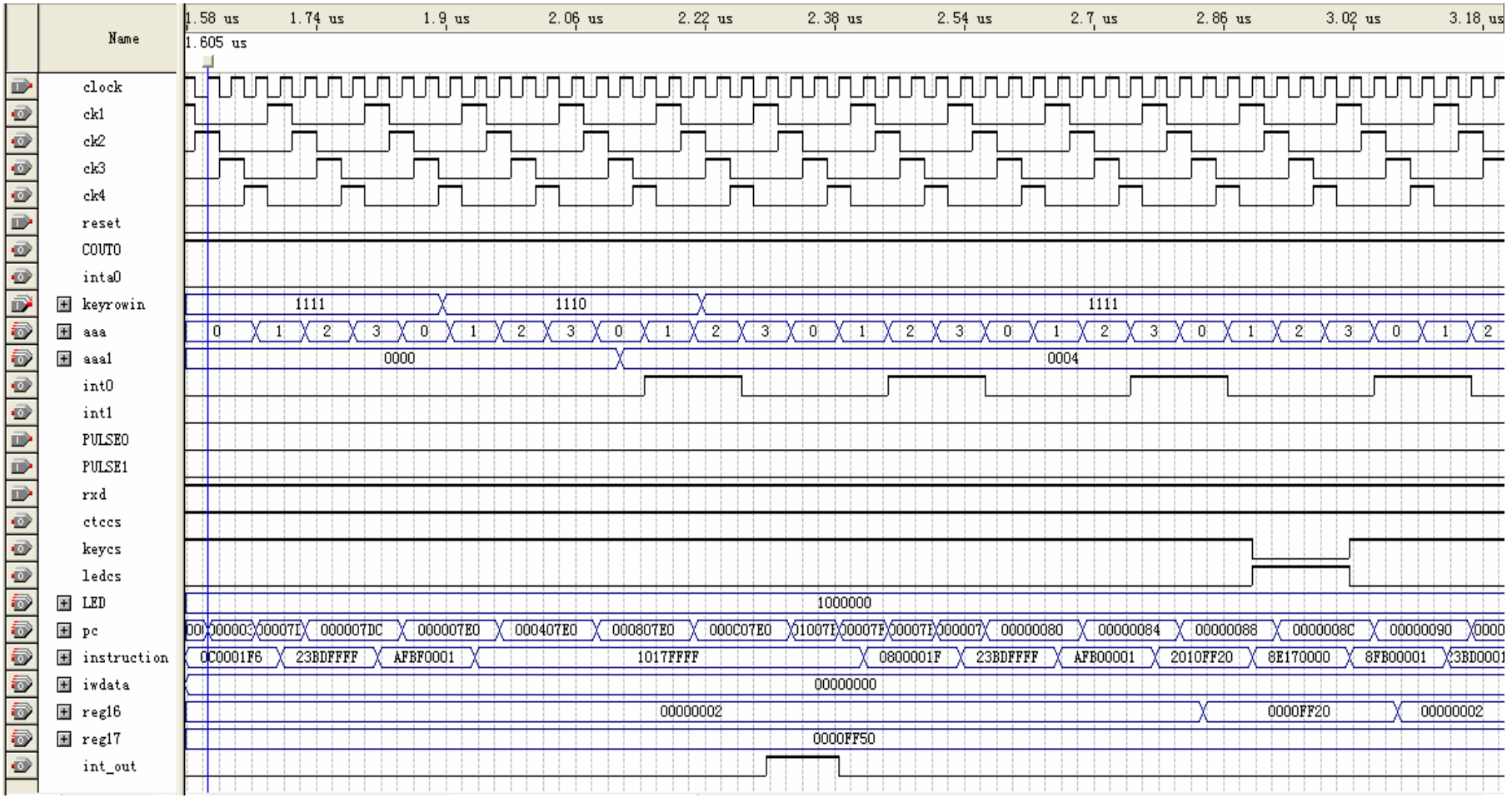

Name
CLK
CS
IOW
address
D
PWM
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0 ns
100.0 ns
120.0 ns
140.0 ns
160.0 ns
00000
01000
00000
00100
00000
00000000
00000008
00000001
00000000

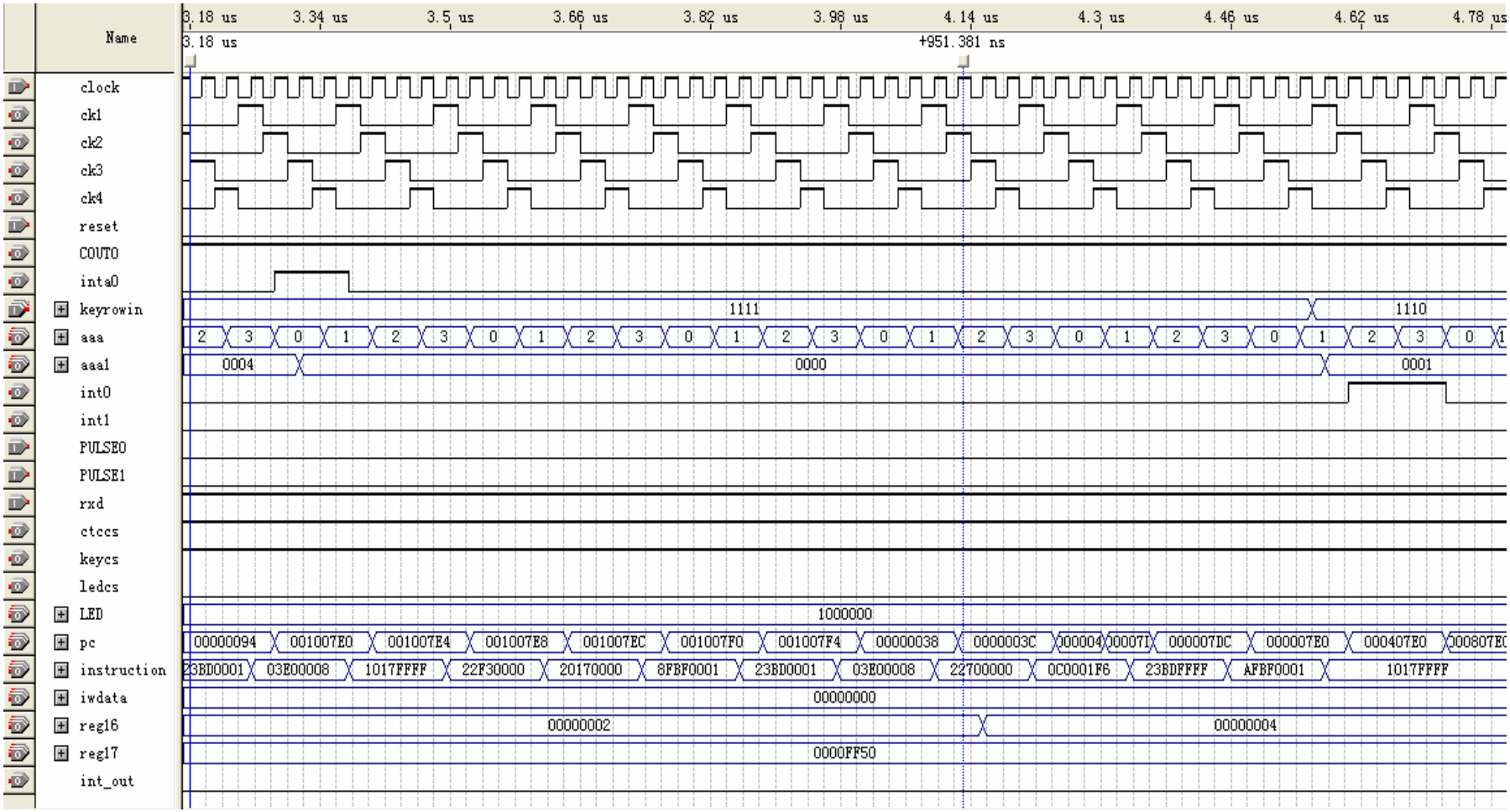
WDT 的 vwf 图（此处为了演示方便，暂时把初值置为 8）

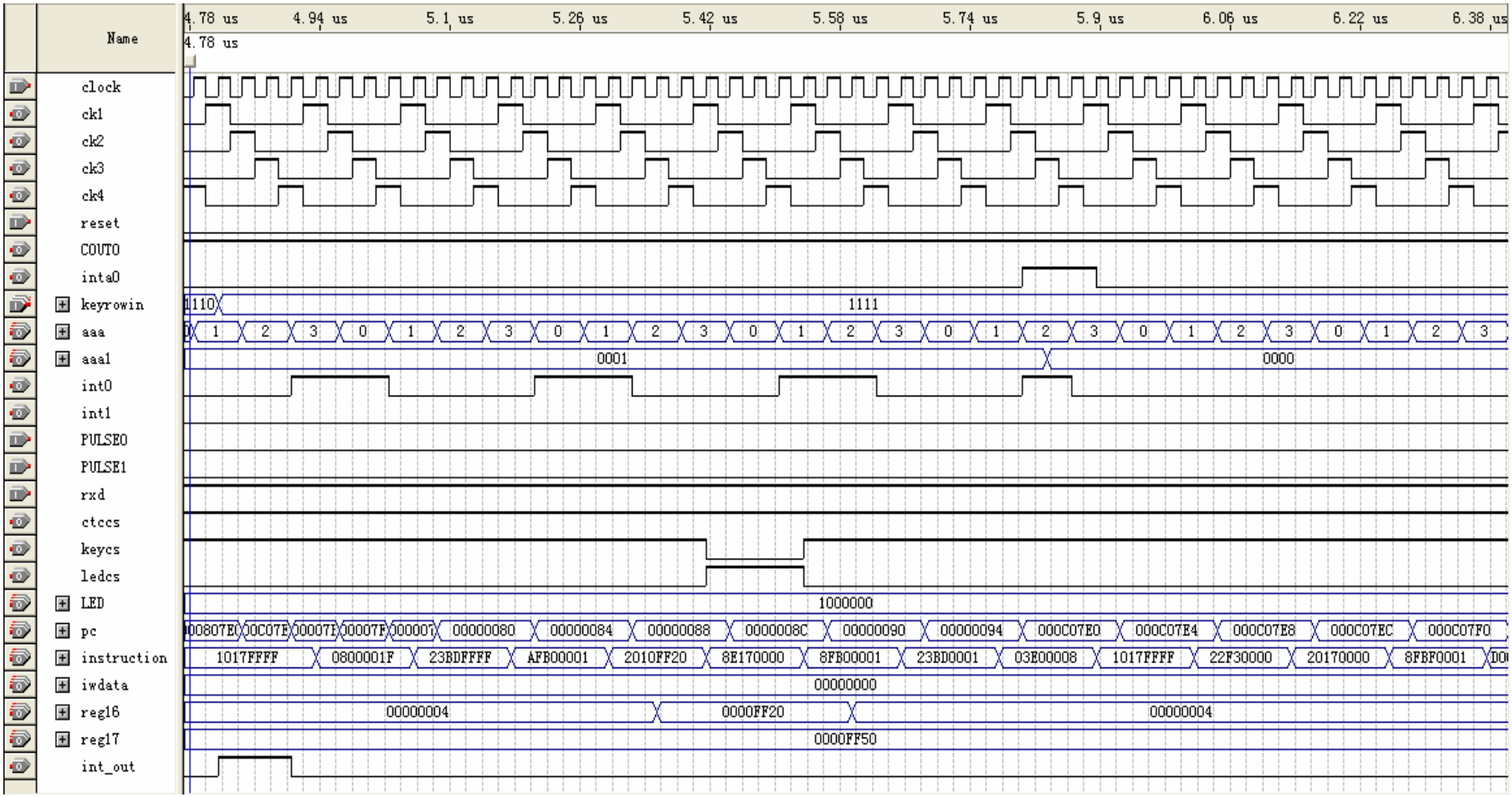


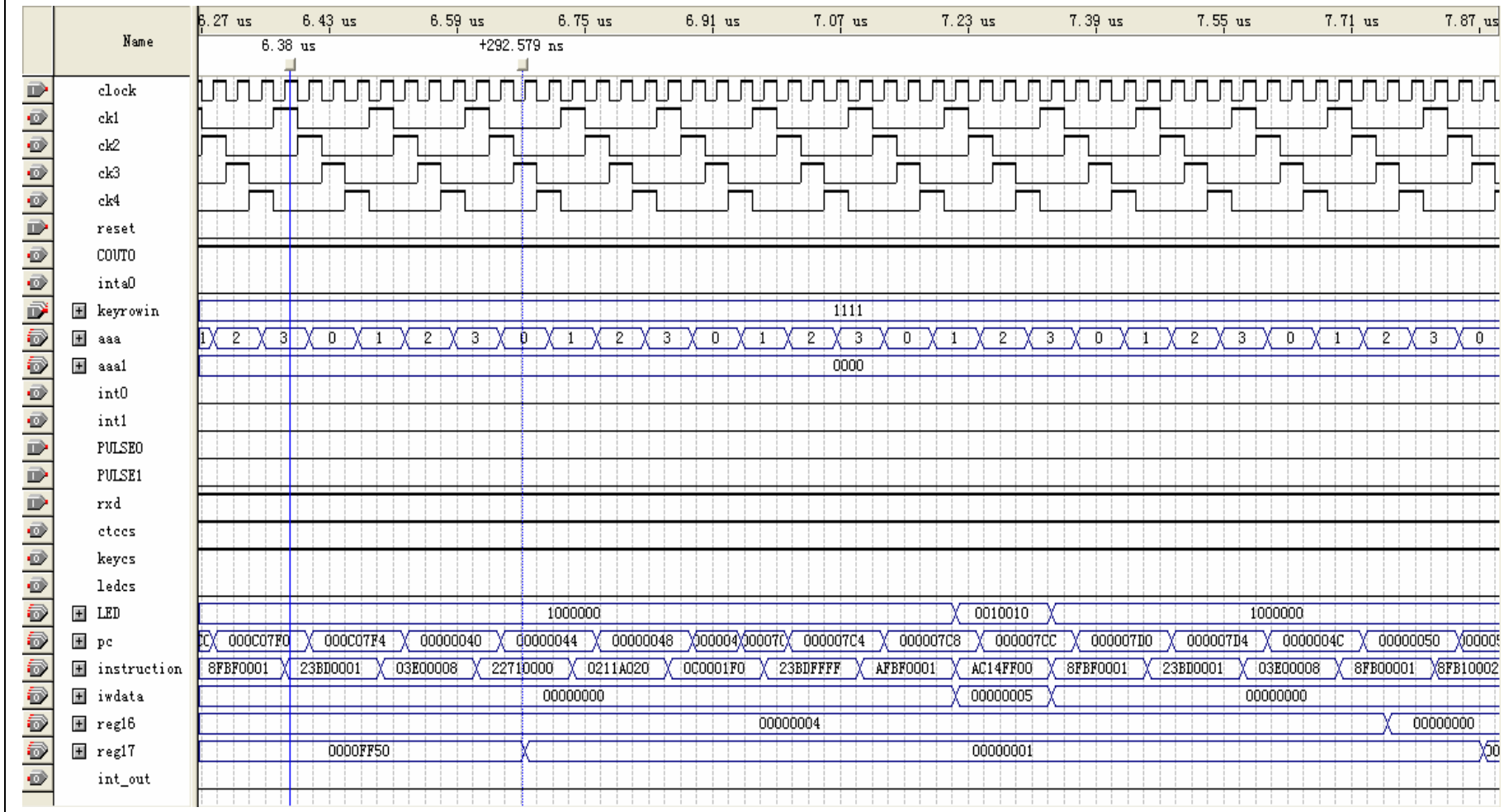
SOC 的 vwf 图

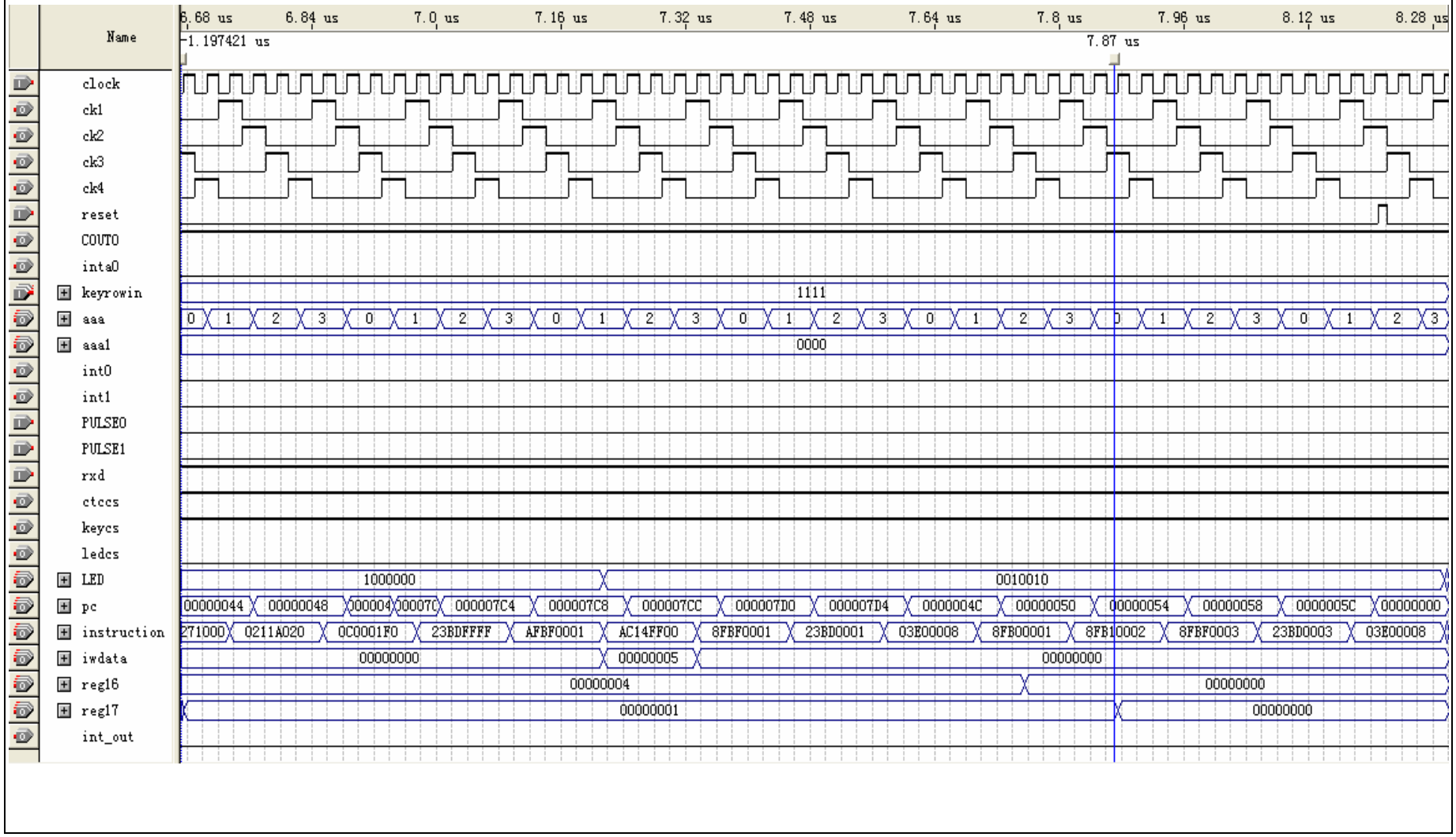

Name
clock
ck1
ck2
ck3
ck4
reset
COUT0
intaO
keyrowin
aaa
aaa1
intO
int1
PULSE0
PULSE1
rxd
ctccs
keycs
ledcs
LED
pc
instruction
iwdata
reg16
reg17
int_out
0 ps
160.0 ns
320.0 ns
480.0 ns
640.0 ns
800.0 ns
960.0 ns
1.12 us
1.28 us
1.44 us
1.6 us
1111
0
0000
1000000


Name
1.58 us
1.74 us
1.9 us
2.06 us
2.22 us
2.38 us
2.54 us
2.7 us
2.86 us
3.02 us
3.18 us
1.605 us
clock
ck1
ck2
ck3
ck4
reset
COUT0
inta0
keyrowin
aaa
aaa1
int0
int1
PULSE0
PULSE1
rxd
ctccs
keycs
ledcs
LED
pc
instruction
iwdata
reg16
reg17
int_out
1111
1110
1111
0000
0004
1000000
00000000
00000002
0000FF20
00000002
0000FF50


Name
3.18 us
3.34 us
3.5 us
3.66 us
3.82 us
3.98 us
4.14 us
4.3 us
4.46 us
4.62 us
4.78 us
3.18 us
+951.381 ns
clock
ck1
ck2
ck3
ck4
reset
COUT0
inta0
keyrowin
aaa
aaa1
int0
int1
PULSE0
PULSE1
rxd
ctccs
keycs
ledcs
LED
pc
instruction
iwdata
reg16
reg17
int_out
1111
1110
0004
0000
0001
1000000
00000094
001007E0
001007E4
001007E8
001007EC
001007F0
001007F4
00000038
0000003C
000007DC
000007E0
000407E0
23BD0001
03E00008
1017FFFF
22F30000
20170000
8FBF0001
23BD0001
03E00008
22700000
0C0001F6
23BDFFFF
AFBF0001
1017FFFF
00000000
00000002
00000004
0000FF50

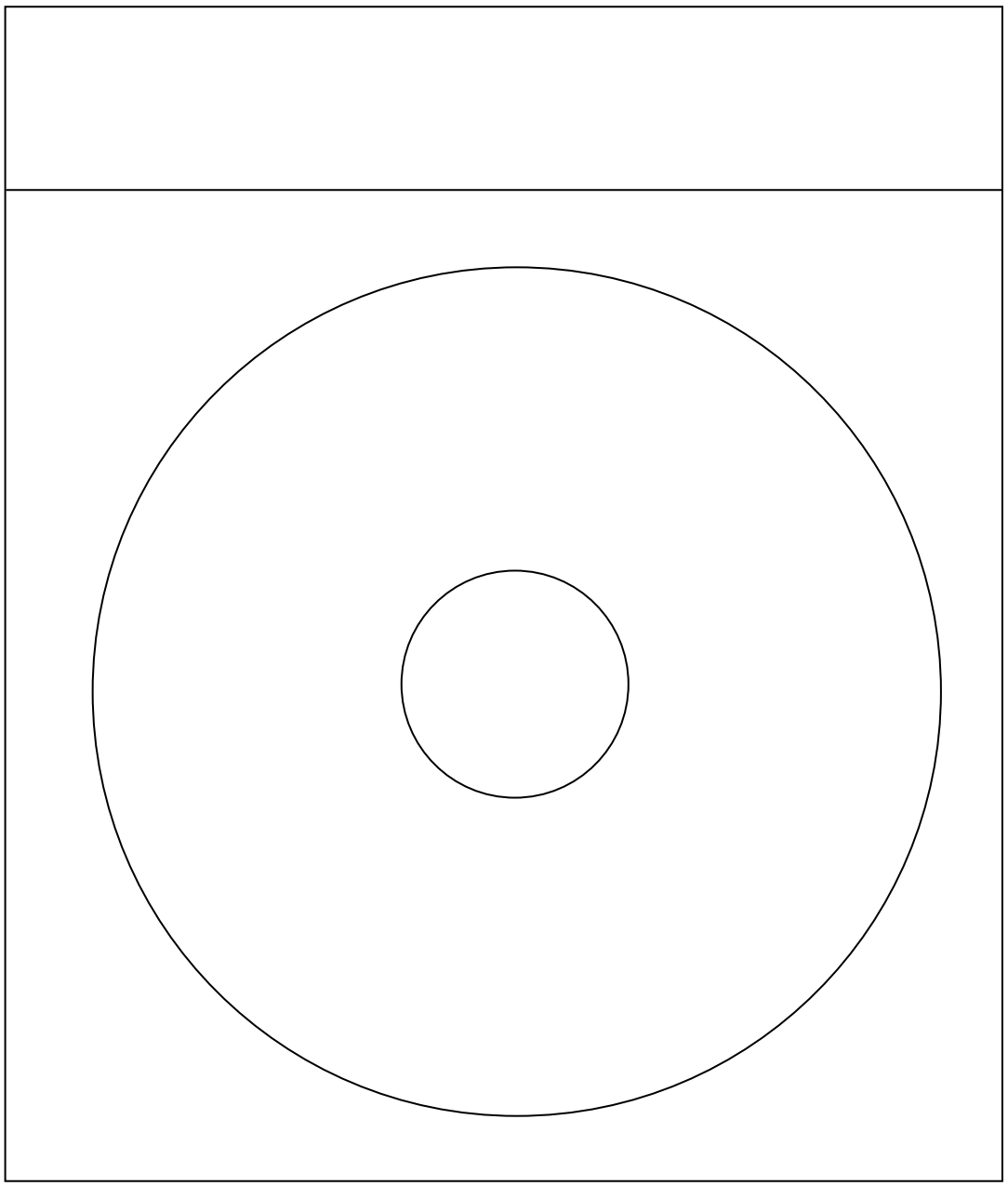

Name
4.78 us
4.94 us
5.1 us
5.26 us
5.42 us
5.58 us
5.74 us
5.9 us
6.06 us
6.22 us
6.38 us
clock
ck1
ck2
ck3
ck4
reset
COUT0
inta0
keyrowin
aaa
aaa1
int0
int1
PULSE0
PULSE1
rxd
ctccs
keycs
ledcs
LED
pc
instruction
iwdata
reg16
reg17
int_out
1110
1111
0001
0000
1000000
00000080
00000084
00000088
0000008C
00000090
00000094
000C07E0
000C07E4
000C07E8
000C07EC
000C07F0
1017FFFF
0800001F
23BDFFFF
AFB00001
2010FF20
8E170000
8FB00001
23BD0001
03E00008
1017FFFF
22F30000
20170000
8FBF0001
00000000
00000004
0000FF20
00000004
0000FF50


Name
6.27 us
6.43 us
6.59 us
6.75 us
6.91 us
7.07 us
7.23 us
7.39 us
7.55 us
7.71 us
7.87 us
6.38 us
+292.579 ns
clock
ck1
ck2
ck3
ck4
reset
COUT0
inta0
keyrowin
1111
aaa
aaa1
0000
int0
int1
PULSE0
PULSE1
rxd
ctccs
keycs
ledcs
LED
1000000
0010010
1000000
pc
000C07F0
000C07F4
00000040
00000044
00000048
000007C4
000007C8
000007CC
000007D0
000007D4
0000004C
00000050
instruction
8FBF0001
23BD0001
03E00008
22710000
0211A020
0C0001F0
23BDFFFF
AFBF0001
AC14FF00
8FBF0001
23BD0001
03E00008
8FB00001
iwdata
00000000
00000005
00000000
reg16
00000004
00000000
reg17
0000FF50
00000001
int_out


Name
6.68 us
6.84 us
7.0 us
7.16 us
7.32 us
7.48 us
7.64 us
7.8 us
7.96 us
8.12 us
8.28 us
-1.197421 us
7.87 us
clock
ck1
ck2
ck3
ck4
reset
COUT0
inta0
keyrowin
1111
aaa
0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3
aaa1
0000
int0
int1
PULSE0
PULSE1
rxd
ctccs
keycs
ledcs
LED
1000000
0010010
pc
00000044 00000048 00000004 00007C 000007C4 000007C8 000007CC 000007D0 000007D4 0000004C 00000050 00000054 00000058 0000005C 00000000
instruction
271000 0211A020 0C0001F0 23BDFFFF AFBF0001 AC14FF00 8FBF0001 23BD0001 03E00008 8FB00001 8FB10002 8FBF0003 23BD0003 03E00008
iwdata
00000000
00000005
00000000
reg16
00000004
00000000
reg17
00000001
00000000
int_out

本组光盘粘贴处

请将光盘装入纸袋中粘贴于此。



本组设计主要测试结果与性能分析（.rpt 中的资源使用情况）

```
+------------------------------------------------------------+
; Assembler Summary                                          ;
+---------------------+--------------------------------------+
; Assembler Status    ; Successful - Tue Sep 19 21:42:50 2006 ;
; Revision Name       ; SOC                                  ;
; Top-level Entity Name ; SOC                                ;
; Family              ; Cyclone                              ;
; Device              ; EP1C6F256C6                          ;
+---------------------+--------------------------------------+



+------------------------------------------------------------------+
; Fitter Summary                                                   ;
+---------------------+--------------------------------------------+
; Fitter Status       ; Successful - Tue Sep 19 21:42:44 2006      ;
; Quartus II Version  ; 5.0 Build 148 04/26/2005 SJ Full Version ;
; Revision Name       ; SOC                                        ;
; Top-level Entity Name ; SOC                                      ;
; Family              ; Cyclone                                    ;
; Device              ; EP1C6F256C6                                ;
; Timing Models       ; Final                                      ;
; Total logic elements ; 4,632 /5,980 (77 %)                       ;
; Total pins          ; 85 /185 (45 %)                             ;
; Total virtual pins  ; 0                                          ;
; Total memory bits   ; 50,176 /92,160 (54 %)                      ;
; Total PLLs          ; 0 / 2 ( 0 % )                              ;
+---------------------+--------------------------------------------+



+------------------------------------------------------------------+
; Flow Summary                                                     ;
+-----------------------+------------------------------------------+
; Flow Status           ; Successful - Tue Sep 19 21:43:35 2006     ;
; Quartus II Version    ; 5.0 Build 148 04/26/2005 SJ Full Version ;
; Revision Name         ; SOC                                      ;
; Top-level Entity Name ; SOC                                      ;
; Family                ; Cyclone                                  ;
; Met timing requirements ; No                                     ;
; Total logic elements  ; 4,632 /5,980 (77 %)                      ;
```

; Total pins ; 85 /185 (45 %)
; Total virtual pins ; 0
; Total memory bits ; 50,176 / 92,160 (54 %)
; Total PLLs ; 0 / 2 (0 %)
; Device ; EP1C6F256C6 ;
; Timing Models ; Final ;
+----------+----------+

+----------+
; Analysis&Synthesis Summary
+----------+----------+
; Analysis & Synthesis Status ; Successful - Tue Sep 19 21:37:38 2006;
; Quartus II Version ; 5.0 Build 148 04/26/2005 SJ Full Version ;
;RevisionName ;SOC ;
; Top-level Entity Name ; SOC ;
;Family ;Cyclone ;
; Total logic elements ; 5,606 ;
; Total pins ;85 ;
; Total virtual pins ; 0 ;
; Total memory bits ; 50,176 ;
;Total PLLs ; 0 ;
+----------+----------+

+----------+
;Power Play Power Analyzer Summary ;
+----------+----------+
; PowerPlay Power Analyzer Status;
Successful - Tue Sep 1920:46:01 2006 ;
; Quartus II Version ; 5.0 Build 148 04/26/2005 SJ Full Version;
; Revision Name ; SOC ;
; Top-level Entity Name ; mips32 ;
; Family ; Cyclone ;
; Device ; EP1C20F400C6 ;
; Total Thermal Power Dissipation ; 120.11 mW ;
; Dynamic Thermal Power Dissipation ; 0.00 mW ;
; Static Thermal Power Dissipation ; 120.11 mW ;
; Power Estimation Confidence ; Low: user provided insufficient toggle rate data ;
+----------+----------+

```
+-------------------------------------------+
; Simulator Summary                          ;
+---------------------------+--------------+
; Type                      ; Value          ;
+---------------------------+--------------+
; Simulation Start Time     ; 0  ps          ;
; Simulation End Time       ; 8.3  us        ;
; Simulation Netlist Size   ; 21811 nodes    ;
; Simulation Coverage       ; 21.84 % ;
; Total  Number  ofTransitions  ; 156001     ;
+---------------------------+--------------+
```

测试 SOC 主频：16.67MHz
测试 SOC 功耗：120.22mW

课程设计总结（包括设计的总结和还需改进的内容）

本次设计，通过大家的不懈努力和团队合作，基本实现了预定目标。

本次设计所用到的知识涉及微机原理与接口技术、计算机组成原理、编译原理、Verilog HDL 等课程，内容覆盖范围广，综合难度大。

在已有知识的基础上，仍要不断学习新知识。如 SOC、IP 核、哈弗结构、MIPS 指令集等。由于相关的书籍少之又少，所以经常利用搜索引擎，感受到网络的优越性。

很多知识相互交叉、渗透，思维转变快，综合运用知识能力强，这给我们的相互协作提出了更高的要求。

从闻所未闻，到逐步了解，认识熟悉这整个过程，既是本课程设计的一个缩影，更是我们温故而知新的探索新知识的历程，它记录了我们学术的进步，也记录下辛辛苦苦的日日夜夜。

通过近两个月的工作，我们有了丰硕的成果:

1、熟悉并掌握了 CPU 及外围接口电路的工作流程及设计;

2、能够熟练掌握 Verilog HDL 进行代码编写;

3、熟悉了 ARM 和 MIPS 指令集;

4、软硬结合，初步有了整机的蓝图;

5、了解了如 SOC、IP 核、哈弗结构等基本概念;

6、对硬件器件有所了解;

7、翻阅了很多相关的资料，丰富了自己的知识;

8、培养了良好的团队合作精神;

教 师 评 语

签名：_____