嵌入式SOC设计

-SOC功能测试：C语言描述

浙江大学 计算机学院 施青松

武汉

2014年 11月11日

SOC功能测试程序

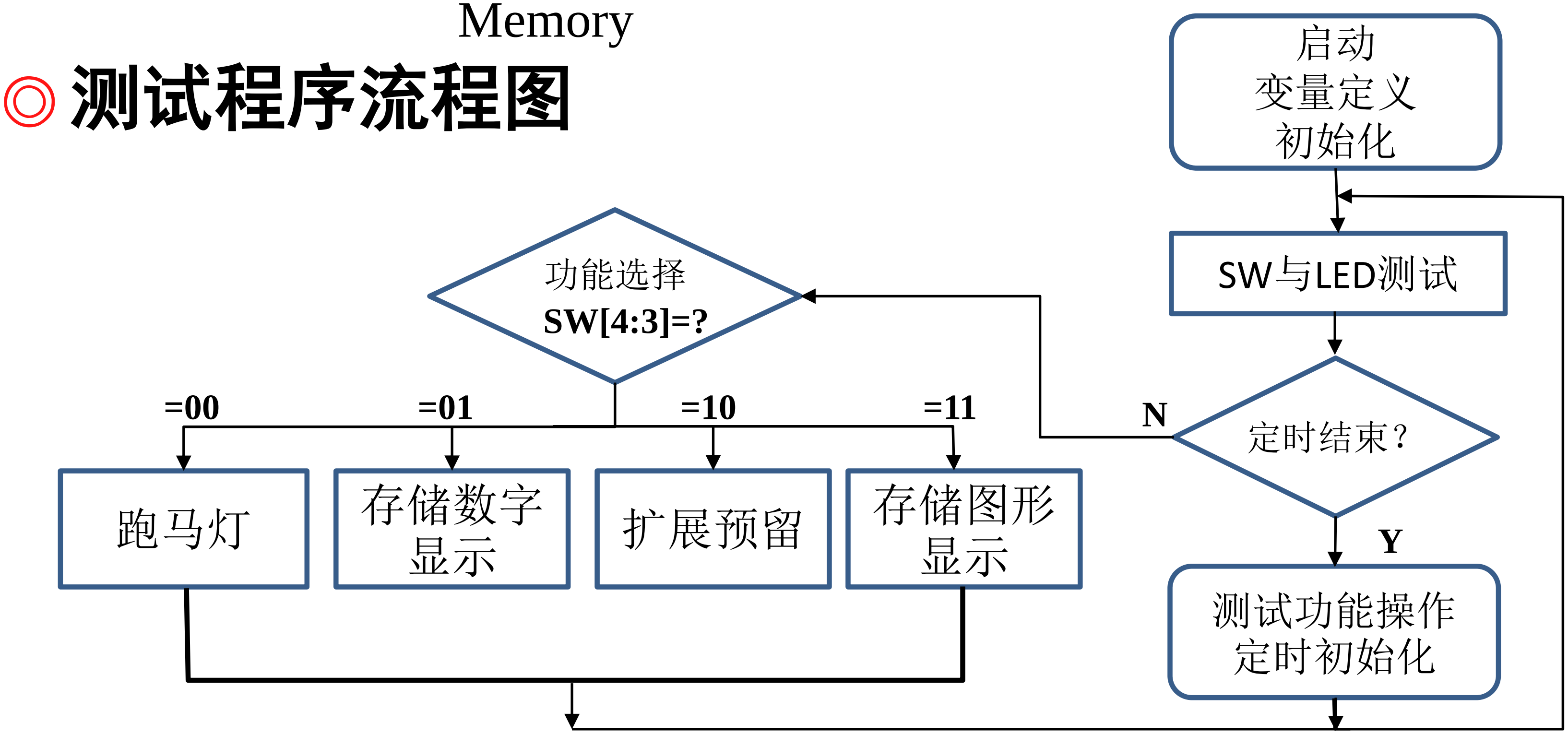
- **测试**
  - 设计测试
    - 功能测试、性能测试
    - 指令测试、通路测试、部件测试、IO测试等
    - 抽样测试、针对性测试
  - 产品测试
    - 完备性测试等
- **功能测试**
  - 选择主要功能设计测试程序
    - 实现验证功能粗调用
  - 选择针对性功能设计测试程序
    - 实现验证细节调试用

SOC功能粗调测试框图

◎ 功能抽样

- 基本指令：lw、sw、add、sub、and、or、slt、beq、j(jar)
- 基本部件：CPU、MIO_BUS、Display、GPIO-LED、GPIO-SW Memory

◎ 测试程序流程图



SOC功能测试代码：C语言

```
        long int *GPIO_Port, *Display_Port, *Counter_Port;
        long int A[16] =……, B[16] =……;
main(){   long int GPIO, Display, Counter, Ctimer, Memory, Count_Soft;
          long int i=0, N_4, N_left_1, Temp, Dot;
          goto Start;

  Start:  long int *GPIO_Port    =  0xF000000;
          long int *Display_Port =  0xE000000;
          Counter_Port = GPIO_Port + 4;
          Counter = 0x0000003F;
          Ctimer    = 0xF8000000;                    //计数常数
          N_left_1 = 0x80000000;
          N_4       = 0x00000004;                    //字内字节数，字地址+1=字节+4
          Dot       = 0xFFFFFFFE;             //最低位为0的数，在七段码的右上角显示一个点

 loop:     Counter_Port = Ctimer;                    //送硬件计数时间常数
           GPIO         = *GPIO_Port;                //读GPIO状态：SW状态
           GPIO          = GPIO <<2;                 //SW和LED对齐
           GPIO_Port    = GPIO;                      //SW状态送LED显示
           Display_Port = Display + 1;               //显示值输出七段显示器
 ……
```

启动、变量定义与初始化

功能判及测试功能代码

```
          Count_Soft    = A[5];                        //取存储器常量，用于软件计数初值
    loop1: GPIO         = *GPIO_Port;                  //读GPIO状态：SW状态
          GPIO          = GPIO <<2;                    //SW和LED对齐
          *GPIO_Port    = GPIO;                        //SW状态送LED显示
```

SW状态循环显示

```
          GPIO          = *GPIO_Port;                  //再读GPIO状态：状态判断用
          Temp          = GPIO & N_left_1; //与80000000相与,取最高位=out0,屏蔽其余位
         //if (Temp ==N_left_1)  goto C_init; //硬件计数溢出out0=1,计数器通道0溢出,转C_init
          Count_Soft    = Count_Soft + 1;               //程序计数延时(加1)
         if (Count_Soft==0) goto C_init;               //若程序计数溢出转C_init
```

三种定时选择

```
    Next:  GPIO         = *GPIO_Port;     //延时未到继GPIO SW状态：判断显示内容SW[4:3]=?
           Temp         = GPIO  & 0x00000018;
           if(Temp ==0x00000000) goto L20;             //SW[4:3]=00,7段显示“点”左移跑马灯
           if(Temp ==0x00000018) goto L21;             // SW[4:3]=11，七段显示B[i]预存七段图形
           if(Temp ==0x00000008) goto L22;             // SW[4:3]=01，七段显示A[i]预置16进制数
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
            *Display_Port = Display ;                  // SW[4:3]=10，输出显示值+1， SW0=1
           goto loop1;
```

四种显示

```
  L20:     if(Dot ==0xFFFFFFFF) Dot =  0xFFFFFFFE;           // 跑马灯全灭置Dot= 0xFFFFFFFE
            *Display_Port = Dot ;                            // SW[4:3]=00，显示跑马灯
           goto loop1;

 ……
```

测试功能与定时初始化代码：四种显示

```
   L21:  Memory = B[i];
          *Display_Port = Memory;                    //SW[4:3]=11，显示预存七段图形
         goto loop1;

  L22:   Memory = A[i+8];                            //字地址是8，字节地址是0x20
         *Display_Port = Memory;                     //SW[4:3]=01，显示预存16进制数， SW0=1
         goto loop1;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 C_init:                                             //延时结束，修改显示值和定时/延时初始化
         Count_Soft    = A[5]; //取存储器常量，用于软件计数初值
         Dot = Dot  <<1;
         Dot = Dot +1;
         i     =( i +1)&0x0000000f;                  // 数组下标+1，内存地址+4
         Display = Display + 1;                      //当前显示值 +1
         if(Display ==0xFFFFFFFF) Display = 5;       //显示全F,则置5
         GPIO           = GPIO_Port;                 //读GPIO状态：SW状态
         GPIO            = GPIO <<2;                 //SW和LED对齐
         *GPIO_Port    = GPIO;                       //SW状态送LED显示
         *Counter_Port = Ctimer;                     //送硬件计数时间常数
         goto Next;
}
```

数组A、B初始化值

A[ ] =

```
{0xf0000000, 0x000002AB, 0x80000000, 0x0000003F, 0x00000001,
0xFFF70000, 0x0000FFFF, 0x80000000, 0x00000000, 0x11111111,
0x22222222, 0x33333333, 0x44444444, 0x55555555, 0x66666666,
0x77777777, 0x88888888, 0x99999999, 0xAAAAAAAA, 0xBBBBBBBB,
0xCCCCCCCC, 0xDDDDDDDD, 0xEEEEEEEE, 0xFFFFFFFF, 0x00000000};
```

B[ ] =

```
{0x557EF7E0, 0xD7BDFBD9, 0xD7DBFDB9, 0xDFCFFCFB, 0xDFCFBFFF,
0xF7F3DFFF, 0xFFFFDF3D, 0xFFFF9DB9, 0xFFFFBCFB, 0xDFCFFCFB,
0xDFCFBFFF, 0xD7DB9FFF, 0xD7DBFDB9, 0xD7BDFBD9, 0xFFFF07E0,
0x007E0FFF, 0x03bdf020, 0x03def820, 0x08002300, 0x00000000};
```

嵌入式SOC设计

-SOC功能测试：RISC-V程序

浙江大学 计算机学院 施青松
紫金港

2021年 03月22日

SOC功能测试RISC-V代码

启动与常数子代码

```
.text: 0000
      j start                   # 0
     add zero, zero, zero       # 4
     add zero, zero, zero       # 8
     add zero, zero, zero       # C
     add zero, zero, zero       # 10
     add zero, zero, zero       # 14
     add zero, zero, zero       # 18
     add zero, zero, zero       # 1C
start:
     lw t0, 14(zero)      # t0 =非0。 需要一个非零常数
     slt t1, zero, t0     # t1 =0000_0001H
     add t2, t1, t1       # t2 =0000_0002H
     add t3, t2, t1       # t3 =0000_0003H
     add a4, t2, t2       # a4 =0000_0004H： 常数4
     add t0, t3, t3       # t0 =0000_0006H
     add t0, t0, t0       # t0 =0000_000CH
     add t4, t0, t3       # t4 =0000_000FH： F
     add t5, t4, t4       # t5 =0000_001EH
     add t5, t5, t5       # t5 =0000_003CH
     add s0, t5, t3       # s0 =0000_003FH： 常数3F
     add t5, t5, t5       # t5 =0000_0078H
     add t5, t5, t5       # t5 =0000_00F0H
     add t6, t5, t4       # t6 =0000_00FFH： FF
     add t5, t5, t5       # t5 =0000_01E0H
     add t5, t5, t5       # t5 =0000_03C0H
```

```
add t5, t5, t5       # t5 =0000_0780H
add t5, t5, t5       # t5 =0000_0F00H
add t5, t5, t5       # t5 =0000_1E00H
add t5, t5, t5       # t5 =0000_3C00H
add t5, t5, t5       # t5 =0000_7800H
add t5, t5, t5       # t5 =0000_F000H
add t5, t5, t5       # t5 =0001_7000H
add t5, t5, t5       # t5 =0003_C000H
add t5, t5, t5       # t5 =0007_8000H
add t5, t5, t5       # t5 =000F_0000H
add t5, t5, t5       # t5 =001E_0000H
add t5, t5, t5       # t5 =003C_0000H
add t5, t5, t5       # t5 =0078_0000H
add t5, t5, t5       # t5 =00F0_0000H
add t5, t5, t5       # t6 =01E0_0000H
add t5, t5, t5       # t6 =03C0_0000H
add t5, t5, t5       # t6 =0780_0000H
add t5, t5, t5       # t6 =0F00_0000H
add t5, t5, t5       # t6 =1E00_0000H
add t5, t5, t5       # t6 =3C00_0000H
add t5, t5, t5       # t6 =7800_0000H
add s1, t5, t5       # S1 =F000_0000H： GPIO地址
or    a2, s1, t5     # a2 =F8000_0000H计数器时常数
add s2, s1, s1       # S2 =E000_0000H： 七段显示地
add t0, s2, s2       # t0 =C000_0000H
add t0, t0, t0       # t0 =8000_0000H： 最高有效位
```

初次运行设置代码

```
loop:
        sub a3, zero, t1            # x13 =FFFFFFFF(MIPS： nor $t2, zero, zero)
        sw a2,  0x4(s1)             # 计数器端口: F0000004，送计数常数x12 =F8000000
        lw a1,  0x0(s1)             # 读GPIO端口F0000000状态:x11=
                                            ={out0， out1， out2， 9’h00， BTN3-BTN0， SW15-SW0}
        add a1, a1, a1              # 左移
        add a1, a1, a1              # 左移2位将SW与LED对齐， 同时D1D0置00， 选择计数器通道0
        sw a1,  0x0(s1)             # x11输出到GPIO端口F0000000， 设置计数器通道 counter_set =
                                    00端口、LED=SW： {GPIOf0[15:2], LED, GPIOf0[1:0]/counter_set}
        add s5, s5, t1              # x21=x21+1
        sw s5, 0x0(s2)              # x21送s2=E0000000七段码端口
        lw s6, 0x14(zero)           # 取存储器20单元预存数据至x22, 程序计数延时常数loop:
        sub a3, zero, t1            # x13 =FFFFFFFF(MIPS： nor $t2, zero, zero， RISC V无not)
        sw a2, 0x4(s1)              # 计数器端口: F0000004，送计数常数x12 =F8000000
        lw a1, 0x0(s1)              # 读GPIO端口F0000000状态:x11=
                                            ={out0， out1， out2， 9’h00， BTN3-BTN0， SW15-SW0}
        add a1, a1, a1              # 左移
        add a1, a1, a1              # 左移2位将SW与LED对齐， 同时D1D0置00， 选择计数器通道0
        sw a1, 0x0(s1)              # x11输出到GPIO端口F0000000， 设置计数器通道counter_set=
                                    00端口、LED=SW： {GPIOf0[15:2], LED, GPIOf0[1:0]/counter_set}
        add s5, s5, t1              # x21=x21+1
        sw s5, 0x0(s2)              # x21送s2=E0000000七段码端口
        lw s6, 0x14(zero)           # 取存储器20单元预存数据至x22, 程序计数延时常数
```

功能判断子代码

```
loop1:
     lw a1, 0x0(s1)              #读GPIO端口F0000000状态，同前。
     add a1, a1, a1;
     add a1, a1, a1;             #左移2位将SW与LED对齐
     sw a1, 0x0(s1);             #再将新$a1:r5写到GPIO端口F0000000,写到LED
     lw a1, 0x0(s1);             #再读GPIO端口F0000000状态
     and s8, a1, t0;             #与80000000相与，即：取最高位=out0,屏蔽其余位
     add s6, s6, t1;             #程序计数延时(加1)
    #beq   s8, t0, C_init;       #硬件计数。out0=1,Counter通道0溢出,转C_init
    beq s6, zero, C_init;        #若程序计数$t5:r13=0,转C_init
l_next:                          # 延时未到，继续：判断7段码显示模式：SW[4:3]
     lw a1, 0x0(s1) ;            #再读GPIO端口F0000000开关SW
     add s7, a4, a4;             #因x14=4, 故s7：x23=00000008
     add s9, s7, s7;             #s9：x25=00000010
     add s7, s7, s9;             #s7：x23=00000018(00011000)：11对应SWO[4:3]
     and s8, a1, s7;             #取SW[4:3]：屏蔽其余位送x24
     beq s8, zero, L00;          #SW[4:3]=00, L00：7段显示"点"循环移位，SW0=0
     beq s8, s7,    L11;         #SW[4:3]=11，L11：显示七段图形，SW0=0
     add s7, a4, a4 ;            #$s2:r18=8
     beq s8, s7,    L01 ;        #SW[4:3]=01, L01：显示内存预置16进制值
L10：  ......                    #SW[4:3]=10，L10显示x21(即时值+1)，SW0=1(用户扩展：)
```

SWO状态
循环显示

硬件/软件/中断
三种定时选择

功能判断

测试功能子代码：四种显示

```
 L10:          sw  s5, 0x0(s2);           #SW[4:3]=10，输出x21显示“当前数+1” ，SW0=1
                j loop2;

L00:
               beq a5, a3, L4;            ## x15=0xFFFFFFFF,当前显示全黑，转移L4
               j L3;
L4:
               add a5, a3, a3;            # x15=0xFFFFFFFE， a3=0xFFFFFFFFH
L3:
               sw a5, 0x0(s2) ;            #SW[4:3]=00,七段显示点左移后显示，SW0=0
               j loop2;
L11:
               lw s5, 0x60(s3);           #SW[4:3]=11，读取内存读取预存七段图形，SW0=0
               sw s5, 0x0(s2) ;           #SW[4:3]=11，输出E0000000显示端口显示七段码图形
               j loop2;
L01:
               lw s5, 0x20(s3) ;          #SW[4:3]=01，读取内存预存16进制数字
               sw s5, 0x0(s2);            #SW[4:3]=01，输出E0000000端口显示16进制数，SW0=1
                j loop2;
```

定时子代码

```
C_init:                                # 延时结束，修改显示值和定时/延时初始化
        lw s6, 0x14(zero) ;            # 取程序计数延时初始化常数
        add a5, a5, a5;                # a5左移，x15=xxxxxxx0，七段图形点左移
        or a5, a5, t1;                 # a5:x15末位置1，消除七段显示器右上角点，不显示
        add s3, s3, a4;                # x14=00000004，LED图形访存地址+4
        and s3, s3, s0;                # 和3F相与，x19=000000xx，屏蔽高位，简单截取低位地址(6位)
        add s5, s5, t1 ;               # x21+1
        beq s5, a3, L6 ;               #若x21=ffffffff，重置x21=5
        j L7;
L6:
        add s5, zero, a4 ;             #x21=4
        add s5, s5, t1 ;               #重置x21=5
L7:
        lw a1, 0x0(s1) ;               #读GPIO端口F0000000状态
        add s8, a1, a1;
        add s8, s8, s8 ;               # 左移2位将SW与LED对齐，同时D1D0置00，选择计数器通道0
        sw s8, 0x0(s1);                # x24输出到GPIO端口F0000000，计数器通道counter_set=00端口
                                       不变、LED=SW：  {GPIOf0[15:2], LED, GPIOf0[1:0]/counter_set}
        sw a2, 0x4(s1)                 # 计数器端口:F0000004，送计数常数x12=F8000000
         j  l_next;                    #本处直接跳转，若中断或子程序 调用则返回
```

注：硬件计数与计数中断时要判断硬件计数溢出已经消除，否则会造成多次进入。

ROM初始数据-.coe

RISCV-DEMO9.coe初始数据

```
memory_initialization_radix=16;
memory_initialization_vector=
0200006F，00000033，00000033，00000033，00000033，00000033，00000033，00000033，
00C02283，00502333，006303B3，00638E33，00738733，01CE02B3，005282B3，01C28EB3，
01DE8F33，01EF0F33，01CF0433，01EF0F33，01EF0F33，01DF0FB3，01EF0F33，01EF0F33，
01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，
01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，
01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF0F33，01EF04B3，01E4E633，00948933，
012902B3，005282B3，406006B3，00C4A223，0004A583，00B585B3，00B585B3，00B4A023，
006A8AB3，01592023，01402B03，0004A583，00B585B3，00B585B3，00B4A023，0004A583，
0055FC33，006B0B33，040B0E63，0004A583，00E70BB3，017B8CB3，019B8BB3，0175FC33，
000C0C63，037C0463，00E70BB3，037C0663，01592023，FB9FF06F，00D78463，0080006F，
00D687B3，00F92023，FA5FF06F，0609AA83，01592023，F99FF06F，0209AA83，01592023，
F8DFF06F，01402B03，00F787B3，0067E7B3，00E989B3，0089F9B3，006A8AB3，00DA8463，
00C0006F，00E00AB3，006A8AB3，0004A583，00B58C33，018C0C33，0184A023，00C4A223，F6DFF06F；
```

RAM初始数据-.coe

- D_mem.coe初始数据

```
memory_initialization_radix=16;
memory_initialization_vector=
F0000000, 000002AB, 80000000, 0000003F, 00000001, FFF70000,
0000FFFF, 80000000, 00000000, 11111111, 22222222, 33333333,
44444444, 55555555, 66666666, 77777777, 88888888, 99999999,
aaaaaaaa, bbbbbbbb, cccccccc, dddddddd, eeeeeeee, FFFFFFFF,
557EF7E0, D7BDFBD9, D7DBFDB9, DFCFFCFB, DFCFBFFF, F7F3DFFF,
FFFFDF3D, FFFF9DB9, FFFFBCFB, DFCFFCFB, DFCFBFFF, D7DB9FFF,
D7DBFDB9, D7BDFBD9, FFFF07E0, 007E0FFF, 03bdf020, 03def820,
08002300;
```

- 下载和仿真均可用

红色数据是LED图形

Thank you!