دانشکده مهندسی کامپیوتر

هم طراحی سخت افزار- نرم افزار

استاد بیکی

گزارش کار پروژه

صدیقه مهرعلیزاده – 973613083

مهرداد قصابی – 973613060

تیر 1401

نرم افزار:

جزل با تعداد محدودی پردازنده کار می کند که یکی از آنها 8051 است پس برا ی قسمت نرم افزار در ابت دای کد آنرا includeمی کنیم.

در اینجا برای نرم افزار از 8051 استفاده می‌کنیم و 2 آرایه 8 تایی را به سخت افزار که در جزل پیاده سازی می‌کنیم می‌فرستیم و سخت افزار سورت شده آرایه ها را به نرم افزار می‌فرستد و نرم افزار آن ها را ادغام می‌کند و در خروجی نمایش می‌دهد.

4 پایه P3,P2,P1,P0 داریم حتما با حروف بزرگکه رابط سخت افزار -نرم افزار هستند و در 8051خروجی و در جزل به صورت ورودی تعریف شدند.هر کدام از پایه ها برای ما مثل سیگنالی سیم هستند که مقداری را باید داخل آن بریزیم پس تعریف می کنیم.

تابع send , merge که به صورت نرم افزاری پیاده سازی شدند.

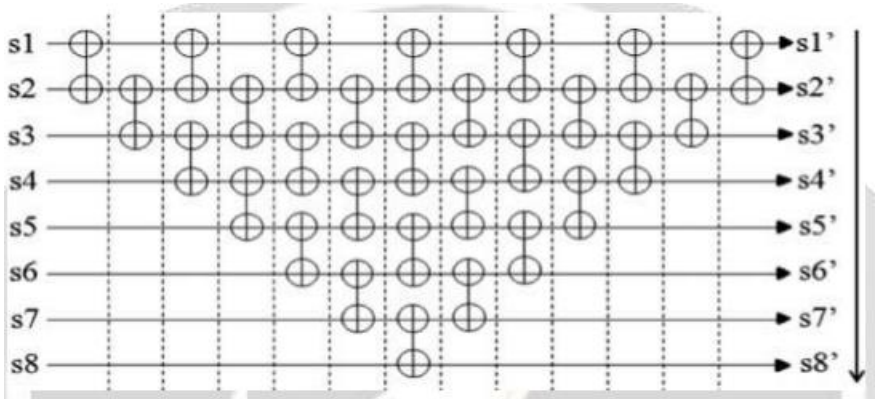
```
#include <8051.h>

void send (char d,char c) {
P0 = d;
P1 = c;
}

void terminate() {P3 = 0x55;}

void main()
{
  send (4,2); // 1
  send (7,9); // 2
  send (8,1); // 3
  send (9,3); // 4
  send (2,6); // 5
  send (1,8); // 6
  send (5,9); // 7
  send (4,4); // 8

  terminate();
}
```

sort.fdl | info.txt | send.c

```c
#include <stdio.h>

int main()
{

    int n1 = 8 ;
    int n2 = 8 ;
    int n3;             //Array Size Declaration
    n3=n1+n2;

    int a[n1],b[n2],c[n3];      //Array Declaration

    c[0]=4;
    c[1]=7;
    c[2]=8;
    c[3]=9;
    c[4]=2;
    c[5]=1;
    c[6]=5;
    c[7]=3;

    int k=n1;


        c[k]=2;
        c[k+1]=9;
        c[k+2]=1;
        c[k+3]=3;
        c[k+4]=6;
        c[k+5]=8;
        c[k+6]=9;
        c[k+7]=4;
```

سخت افزار:

تابع sort را در قسمت سخت افزار پیاده سازی کردیم بدین صورت که از bubble sort استفاده کردیم که موازی جلو می‌رودو در هرکلاک swap را مطابق شکل زیر انجام میدهد.



Data Path

```
reg i : tc(10);

reg hold0: ns(8);
reg hold1: ns(8);
reg hold2: ns(8);
reg hold3: ns(8);
reg hold4: ns(8);
reg hold5: ns(8);
reg hold6: ns(8);
reg hold7: ns(8);

reg r0: ns(8);
reg r1: ns(8);
reg r2: ns(8);
reg r3: ns(8);
reg r4: ns(8);
reg r5: ns(8);
reg r6: ns(8);
reg r7: ns(8);

reg huld0: ns(8);
reg huld1: ns(8);
reg huld2: ns(8);
reg huld3: ns(8);
reg huld4: ns(8);
reg huld5: ns(8);
reg huld6: ns(8);
reg huld7: ns(8);

reg a0: ns(8);
reg a1: ns(8);
reg a2: ns(8);
reg a3: ns(8);
reg a4: ns(8);
```

در این قسمت ابتدا پایه های P را که قبال توضیح دادیم را با تعداد بیت و بدون عالمت بودن تعریف می کنیم .و رجیستری را برای نگهداری حاصل عملوند تعریف می کنیم .بعد از آن به قسمت sfg می رسیم که مثل قسمتی که در int & void ،vhdl &..راتعریف می کند است. پس تابع ها را تعریف می کنیم در این قسمت و در نهایت تابع emptyرا داریم که برای موقعی که می خواهیم در fsm تابع ها را با مراحل و نحوه ی اجرا و ترتیب را برای تابع هابنویسیم باید از else,if استفاده کنیم باید حتما بعد از else تابعی باشد پسemptyرا یاوریم که هیچ کاری هم انجام نمی دهد.

Fsm

این قسمت سورت را برای ما نشان می‌هد.

مقایسه و سورت برای هرآرایه 8 تایی جداگانه انجام میشود.

```
fsm sort(sys)
{
        initial s0;

        state  s1, s2, s3, s4, s5, s6, s7, s8, s9,s10,
              s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,
              s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,
              s31,s32,s33,s34,s35,s36,s37,s38,s39,s40,
              s41,s42,s43,s44,s45,s46,s47,s48,s49,s50,
              s51,s52,s53,s54,s55,s56,s57,s58,s59,s60,
              s61,s62,s63,s64,s65,s66,s67,s68,s69,s70,
              s71,s72,s73,s74,s75,s76,s77,s78,s79,s80,
              s81,s82,s83,s84,s85,s86,s87,s88,s89,s90,
              s91,s92,s93,s94,s95,s96,s97,s98,s99,s100,
              s101,s102,s103,s104,s105,s106,s107,s108,s109,s110,
              s111,s112,s113,s114,s115,s116,s117,s118,s119,s120,
              s121,s122,s123,s124,s125,s126,s127,s128,s129,s130,
              s131,s132,s133,s134,s135,s136,s137,s138,s139,s140,
              s141,s142,s143,s144,s145,s146,s147;


        @s0 init -> s1;

        @s1 if (r0>r1) then noth ->s2;
        else swap01 ->s9;
        //
        @s9 set1 ->s2 ;


        @s2 if (r1>r2) then noth ->s3;
        else swap12 ->s10;
        //
```

```
      @s3 if((r0>r1) & (r2>r3)) then noth ->s4;
      else if((r1>r0) & (r2>r3)) then swap01 ->s11;
      else if((r1>r0) & (r2>r3)) then swap23 -> s12;
      else (swap01,swap23) -> s13 ;
      //
      @s11 set1 ->s4 ;
      @s12 set3 ->s4 ;
      @s13 (set1,set3) ->s4 ;


      @s4 if((r1>r2) & (r3>r4)) then noth ->s5;
      else if((r2>r1) & (r3>r4)) then swap12 ->s14;
      else if((r1>r2) & (r4>r3)) then swap34 -> s15;
      else (swap12,swap34) -> s16 ;
      //
      @s14 set2 ->s5 ;
      @s15 set4 ->s5 ;
      @s16 (set2,set4) ->s5 ;


      @s5 if((r0>r1) & (r2>r3) & (r4>r5)) then noth ->s6;
      else if((r0>r1) & (r2>r3) & (r5>r4)) then swap45 ->s17;
      else if((r0>r1) & (r3>r2) & (r4>r5)) then swap23 -> s18;
      else if((r0>r1) & (r3>r2) & (r5>r4)) then (swap45,swap23) -> s19;
      else if((r1>r0) & (r2>r3) & (r4>r5)) then swap01 -> s20;
      else if((r1>r0) & (r2>r3) & (r5>r4)) then (swap45,swap01) -> s21;
      else if((r1>r0) & (r3>r2) & (r4>r5)) then (swap01,swap23) -> s22;
      else (swap01,swap23,swap45) -> s23 ;
      //
      @s17 set5 ->s6 ;
      @s18 set3 ->s6 ;
      @s19 (set5,set3) ->s6 ;
      @s20 set1 ->s6 ;
      @s21 (set5,set1) ->s6 ;
```

و ادامه که پیوست شده است.

در این قسمت همانطور که تقریبا گفته شد مراحل وترتیب اجرای تابع ها را با استفاده از ماشین حالت وبا اسنفاده از else,ifمی نویسیم .

Core & port

در این قسمت core که همان 8051 است و portها را مشخص می کنیم و هم چنین source بودن و خروجی بودن آن ها را

Sys

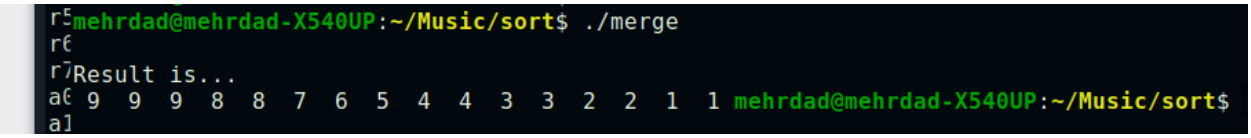
در این قس مت نیز ماژول ها را Instantiate می کنیم Use.هم در اینجا مثل map port در vhdl عمل میکند و اخرین use هم برای تابع dp می نویسیم که از ماژول استفاده شود.(نه که صرفا تعریف شود).

و درنهایت با استفاده از دستورات گفته شده کد را اجرا می‌کنیم.(فایل توضیحات و اجرای کد پیوست شده است.)

```
sedii@ubuntu:~/Downloads/sort$ export PATH=$PATH:/opt/gezel/bin/
sedii@ubuntu:~/Downloads/sort$ sdcc send.c
sedii@ubuntu:~/Downloads/sort$ gplatform sort.fdl
```

```
i8051system: loading executable [sort.ihx]
============swap01============
r0 = 4/7
r1 = 7/7
r2 = 8/8
r3 = 9/9
r4 = 2/2
r5 = 1/1
r6 = 5/5
r7 = 3/3
============set1============
r0 = 7/7
r1 = 7/4
r2 = 8/8
r3 = 9/9
r4 = 2/2
r5 = 1/1
r6 = 5/5
r7 = 3/3
============swap12============
r0 = 7/7
r1 = 4/8
r2 = 8/8
r3 = 9/9
r4 = 2/2
r5 = 1/1
r6 = 5/5
r7 = 3/3
```

و ادامه خروجی که در ویدیو قابل مشاهده است.



لینک مربوط به ویدیو از اجرا و توضیحات

https://drive.google.com/file/d/1bW_r3sVe9rwhZbOu8iU5WW7YyV1fsdV-/view?usp=sharing

مراجع

لینک مقاله مربوط به سورت استفاده شده در قسمت سخت افزار

http://ijariie.com/AdminUploadPdf/hardware_implementatioon_of__sorting__algorithm_using_FPGA_ijariie7623.pdf

https://github.com/Mehrdadghassabi/gezel_intro/tree/main/example/merge_sort