بسمه تعالی

HW4 codesign

سید محمدرضا حسینی 97243129

سید عباس میرقاسمی 97243068

با توجه به اینکه درصورت پروژه ذکر شده بود باید از شیوه mapped-memory برای برقراری ارتباط بین سخت افزار و نرم افزار استفاده کنیم؛ ما به همین شکل عمل کردیم و طبق مثال کتاب جلو رفتیم و بر همین اساس آدرسهایی از حافظه به صورت قراردادی انتخاب کردیم تا پل ارتباطی میان سخت افزار و نرم افزار باشد. ملاحظاتی در رابطه با شیوه تعریف هر پورت ارتباطی باید لحاظ میشد که شامل armsystemsource و armsystemsink و myarm=core میشد؛ همچنین توجه به نوع پورت که ورودی است یا خروجی هم بسیار حائز اهمیت بود. در گام بعد هم، متغیرهایی volatile در سمت نرم افزار تعریف کردیم که با استفاده از اشارهگر به خانه های حافظه متناظر با هر بخش اشاره میکردند و فقط طراحی و پیادهسازی یک پروتکل ارتباطی لازم بود تا ارتباط سخت افزار و نرم افزار بطور کامل برقرار شود. در طراحی این پروتکل ارتباطی که بخشی از ماشین حالت اصلی سخت افزار بود باید به سخت افزار اطلاع میدادیم که چه موقع پارامترهای ورودی سیستم که در صورت تمرین مشخص شده بود آماده استفاده است و به همین خاطر یک پورت با نام REQ تعریف کردیم تا این وظیفه را برعهده بگیرد. به صورت کلی 8 پورت برای input ، 8 پورت برای خروجی ، 1 پورت برای ACK و یک پورت برای REQ در نظر گرفتیم . پروتوکل ارتباطی بین سخت افزار و نرم افزار بدین صورت است که نرم افزار با معکوس کردن مقدار REQ ، به سخت افزار اعلام میکند که میتواند کار را انجام دهد و سپس سخت افزار نیز باید یک ACK برای نرم افزار بفرستد تا هردو به کار خود ادامه دهند . ACK ها به صورت افزایشی میباشند و در کل از 11 ، Ack استفاده کردیم .

در ابتدا برای اینکه بتوانیم اعداد اعشاری را به سخت افزار بدهیم باید آن را به صورت fixed point و int تبدیل کنیم . به همین علت از 8 بیت در نظر گرفته شده 3 بیت برای قسمت صحیح عدد و 5 بیت برای قسمت اعشاری عدد در نظر گرفتیم . در ادامه برای تبدیل ابتدا عدد را در 64 ضرب میکنیم که این کار به مثابه یک شیفت چپ به اندازه 6 بیت میباشد . سپس این اعداد را به سخت افزار داده و با انجام دادن عملیات ها درون سخت افزار مقدار را دریافت کرده و از حالت مکمل 2 به حالت ابتدایی تبدیل میکنیم .

با توجه به کد ابتدایی داده شده ، در ابتدا با مشخص کردن تعداد اجرا های حلقه در کد ، اجرای حلقه ها را به صورت پشت سر هم درآورده و حلقه ها را از برنامه حذف کردیم .

```
l = 0-----------------------------
j = 0-------------
i = 0
i = 2
i = 4
i = 6
u1 = -1.000000 ---   u2 = 0.000000---
c1 = 0.000000 ---   c2 = -1.000000---
l = 1-----------------------------
j = 0-------------
i = 0
i = 4
u1 = 0.000000 ---   u2 = -1.000000---
j = 1-------------
i = 1
i = 5
u1 = -1.000000 ---   u2 = -0.000000---
c1 = 0.707107 ---   c2 = -0.707107---
l = 2-----------------------------
j = 0-------------
i = 0
u1 = 0.707107 ---   u2 = -0.707107---
j = 1-------------
i = 1
u1 = 0.000000 ---   u2 = -1.000000---
j = 2-------------
i = 2
u1 = -0.707107 ---   u2 = -0.707107---
j = 3-------------
i = 3
u1 = -1.000000 ---   u2 = 0.000000---
c1 = 0.923880 ---   c2 = -0.382683---
```

```
root@6ac88873b8ec:/opt/src# arm-linux-gcc -static main.c -o main -lm
root@6ac88873b8ec:/opt/src# gplatform model.fdl
core myarm
armsystem: loading executable [main]
armsystemsink: set address 2147483652
armsystemsink: set address 2147483688
armsystemsink: set address 2147483692
armsystemsink: set address 2147483696
armsystemsink: set address 2147483700
armsystemsink: set address 2147483704
armsystemsink: set address 2147483708
armsystemsink: set address 2147483712
armsystemsink: set address 2147483716
X readed completely
x0=0/40 x1=0/40 x2=0/0x3=0/0 x4=0/40 x5=0/40x6=0/0 x7=0/0
Y readed completely
y0=0/0 y1=0/0 y2=0/0y3=0/0 y4=0/0 y5=0/0y6=0/0 y7=0/0
sort done completely
x0=40/40 x1=40/40 x2=0/0x3=0/0 x4=40/40 x5=40/40x6=0/0 x7=0/0
y0=0/0 y1=0/0 y2=0/0y3=0/0 y4=0/0 y5=0/0y6=0/0 y7=0/0
X OUTPUT:
o0=0/20 o1=0/0 o2=0/10 o3=0/0 o4=0/0 o5=0/0 o6=0/10 o7=0/0
Y OUTPUT:
o0=20/0 o1=0/0 o2=10/fffffff0 o3=0/0 o4=0/0 o5=0/0 o6=10/10 o7=0/0
0
0
Total Cycles: 175816
root@6ac88873b8ec:/opt/src#
```

نتیجه اجرای کد با ورودی های مشابه در کد داده شده برابر است و سیستم ساخته شده به درستی کار میکند.