دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
پروژه ساختار و زبان کامپیوتر

عنوان:

پروژه اول: شبیه ساز مدارات منطقی با قابلیت طراحی، انتقال و اجرای زنده روی برد آردوینو

Arduino Logic Circuit Simulator

نگارش

محمدپارسا جعفرنژادی، پوریا رحمانی، محمدمهدی عابدینی، نیما قدیرنیا

بهمن ۱۴۰۳

۱ مقدمه

منطق قابل پیکربندی مجدد یا Reconfigurable Logic ، از موضوعات میان‌رشته‌ای جذاب میان کامپیوتر و برق است که به ما قابلیت پیاده‌سازی سخت‌افزارهای دیجیتال و ساختارهای منطقی را به صورت انعطاف‌پذیر می‌دهد و میتوانیم منطق های مورد نظر را بسته به نیازهایمان تغییر دهیم.
هدف این پروژه طراحی و توسعه سیستمی است که امکان شبیه‌سازی و پیاده‌سازی مدارهای منطقی را بر روی سخت‌افزار به صورت پویا فراهم کند .
در این برنامه به صورت مجازی از یک برد آردوئینو استفاده شده و کاربر میتواند از طریق یک رابط گرافیکی که با javaFX نوشته شده است ، جداول صحت متعددی طراحی کند که ورودی های این جدول ها ، میتوانند خروجی های جدول‌های دیگر باشند که یعنی دست کاربر را برای طراحی مدار ، آزادتر میگذارد. در نهایت با انتقال منطق طراحی شده توسط کاربر باتوجه به جدول‌ها به برد آردوئینو، مدار به درستی و با عملکرد مطلوب به برد انتقال میابد .

۲ آماده‌سازی بُرد آردوینو

۲-۱ راه‌اندازی شبیه‌ساز

برای شبیه‌سازی مدار آردوینو در این پروژه از شبیه‌ساز Wokwi استفاده شده است. با استفاده از این نرم‌افزار می‌توانیم برد آردوینو با قطعات و اتصالات دلخواه طراحی کنیم و عملکرد آن را شبیه‌سازی کنیم. اجزای تعریف شده عبارتند از

- برد Arduino Uno به همراه یک breadboard برای اتصالات.
- یک عدد DIP Switch دارای ۸ سوییچ جهت ورودی گرفتن.
- ۴ عدد ال‌ای‌دی برای نمایش خروجی.
- دو عدد 7-Segment Display به همراه دو عدد چیپ 74HC595 به منظور نمایش خروجی به صورت دهدهی.

۲-۲ نرم‌افزار بُرد

حال باید نرم‌افزاری بنویسیم که خروجی‌ها را با توجه به ورودی‌ها و مدار تعریف شده، مقداردهی کند. برای این کار از تابع‌های تعریف شده برای آردوینو در زبان C++ استفاده می‌کنیم. برای کامپایل کردن این کد در این پروژه از PlatformIO استفاده شده است.
ساختار کلی کد به این صورت است که پس از مشخص کردن انواع پورت‌های ورودی و خروجی در تابع `()setup`، در هر دور چرخش در تابع `()loop`، برنامه ورودی‌ها را می‌خواند و طبق آن‌ها و داده‌های مدار، ولتاژ ال‌ای‌دی‌ها را تعیین می‌کند.

۲-۲-۱ نحوه ذخیره‌سازی مدار

از آنجایی که ۸ ورودی داریم، ۲۵۶ حالت ورودی وجود دارد که درون ثابت STATE_COUNT قرار دارد.
اطلاعات مدار در آرایه `circuit[STATE_COUNT]` ذخیره می‌شود. به این شکل که هر اندیس این آرایه نشان‌دهنده یک حالت ورودی است. در حقیقت برای دسترسی به خروجی به ازای یک ورودی خاص باید ورودی‌ها را به صورت دودویی به عدد تبدیل کنیم و آن اندیس را از این آرایه بخوانیم. محتوای این آرایه هم از جنس byte می‌باشد که یعنی دارای ۸ بیت است. هر بیت نشان‌دهنده یک وضعیت خروجی می‌باشد. از آنجایی که فقط ۴ خروجی داریم کافی است ۴ بیت کم‌ارزش بایت را بخوانیم و با توجه به هر کدام، ولتاژ ال‌ای‌دی‌ها را مشخص کنیم.

۲-۲-۲ نحوه پیاده‌سازی خروجی دهدهی

این ویژگی اضافه بر سازمان تعریف شده و برای مدارهایی که عملیات جبری مانند جمع یا تفریق انجام می‌دهند می‌تواند کاربردی باشد. برای این کار از آنجایی که پورت‌های دیجیتال در حال استفاده توسط ورودی‌ها و ال‌ای‌دی‌ها هستند، باید از پورت‌های آنالوگ برای مشخص کردن segmentهای صفحه استفاده کنیم. به این صورت که با یک سیگنال clock، یک سیگنال latch و یک سیگنال data می‌توانیم به یک شیفت رجیستر داده‌ها را بدهیم و در هر لحظه خروجی موازی شیفت رجیسترها را به عنوان ورودی 7-Segment Display استفاده کنیم. قطعه 74HC595 یک شیفت رجیستر ۸ بیتی از نوع SIPO است و مناسب این کار می‌باشد. برای ارسال داده اول سیگنال latch را پایین می‌بریم تا رجیسترها ورودی بگیرند. سپس با استفاده از تابع `shiftOut` می‌توانیم داده خود را ارسال کنیم. این داده هم به ازای هر رقم در یک آرایه تعریف شده است.

۲-۲-۳ نحوه دریافت اطلاعات

برای ارتباط برقرار کردن بین بُرد و دستگاه، نیازمند درگاه Serial هستیم. برنامه روی بُرد می‌تواند با فعال‌سازی آن در ستاپ، در هر دور حلقه داده‌های از طرف دستگاه را بخواند. قراردادی که برای ارسال داده‌های مدار داریم این است که یک رشته ۲۵۶ کاراکتری به فرمت Hexadecimal دریافت می‌شود. هر کاراکتر آن از آنجایی که در مبنای ۱۶ است، معادل ۴ بیت است. این ۴ بیت را معادل خروجی ال‌ای‌دی‌ها قرار می‌دهیم. در حقیقت کافی است که هر کاراکتر ارسال شده را به مقدار مبنای ۱۶ تبدیل کرده و آن را در اندیس متناظر خود در آرایه `circuit` ذخیره کنیم.

۲-۲-۴ ارسال داده از سمت دستگاه

در شبیه‌ساز Wokwi درگاه Serial بر روی یک پورت شبکه تحت پروتکل RFC 2217 شبیه‌سازی می‌شود و کتابخانه pySerial در پایتون به ما این قابلیت را می‌دهد که از این پروتکل استفاده کنیم. اسکریپت `serial_write.py` این کار را می‌کند. به طوری که در هر اجرا داده‌های درون فایل `data.bin` را خوانده و از طریق پروتکل، به پورت شبیه‌ساز ارسا ل می‌کند. نوشتن درون فایل داده به عهده رابط کاربری طراحی مدار می‌باشد.

۳ منطق تأثیر ورودی کاربر بر مدار روی برد

۳-۱ یادآوری شیوه‌ی تغییر کد روی برد

همانطور که پیشتر اشاره شد ، اجرای کد پایتون `serial_write.py` مدار روی برد را باتوجه به محتوای فایل باینری `data.bin` تغییر می‌دهد.
این فایل باینری شامل یک رشته ۲۵۶ رقمی Hex است که هر رقم آن نشان‌دهنده‌ی وضعیت چهار LED به ازای یک ورودی خاص است.
به‌طور مثال، با فرض این که *شماره‌گذاری بیت‌ها را از سمت چپ به راست بگیریم، یعنی بیت پرارزش شماره ۱ را داشته باشد*، اگر رقم سوم این رشته برابر ۵ باشد ،پس از اجرا شدن `serial_write.py` اگر همه ورودی‌ها صفر باشند به جز ورودی شماره ۷، لامپ‌های دوم و چهارم روشن می‌شوند و دو لامپ دیگر خاموش می‌مانند.
بنابراین ، فایل باینری به‌طور کامل مدار را مشخص می‌کند و درنتیجه هدف ما در پیاده‌سازی که با

زبان جاوا انجام شده ، دارای سه بخش است:

- تعیین خروجی ال‌ای‌دی ها به ازای تمام حالات در مدار تعریف شده توسط کاربر
- تشکیل رشته‌ی Hex متناظر خروجی‌ها که در بخش پیش به‌دست آوردیم و نوشتن آن در فایل باینری
- اجرا کردن کد serial_write.py از طریق کد جاوا

با اجرا کردن این سه مرحله، مدار آن‌گونه که کاربر می‌خواهد، تغییر می‌کند.
پیش از آن‌که به این سه بخش بپردازیم ، کلیتی از نحوه‌ی شبیه‌سازی جدول‌ها توسط کد را بیان می‌کنیم:

۳-۲ اجزای اصلی منطق پیاده‌سازی

بخش model در کد ، از سه کلاس تشکیل شده :

- کلاس VarLed

اعضای این کلاس، متغیرهای میانی و LED های ما هستند (یک متغیر boolean این دو نوع را از هم جدا می‌کند). هر object از این کلاس ، یک آرایه status دارد که در آن، وضعیت این متغیر به ازای ورودی‌های مختلف نگه‌داری می‌شود.
همچنین یک متغیر boolean به نام isEvaluated در هر object مشخص می‌کند که آیا این آرایه به‌ازای جداول کنونی ، تا کنون به‌درستی محاسبه شده‌است یا خیر.
متغیر مهم دیگری که نگهداری می‌شود ، عدد صحیح outputIn است که تعیین می‌کند VarLed مدنظر، در جدول چندم به عنوان خروجی آمده‌است(اگر هیچ‌گاه به عنوان خروجی نیامده‌بود، این عدد برابر منفی یک خواهد بود)

- کلاس Circ

این کلاس شامل اطلاعات و متغیرهای static مربوط به کل مدار می‌شود، مثل تعداد سوئیچ‌ها(بیت‌های ورودی)، تعداد متغیرها، مقدار default برای LED هایی که به عنوان خروجی نیامده‌اند و مقدار default برای variable هایی که به عنوان خروجی نیامده‌اند.
همچنین arraylist کل variable ها و کل led ها ، در این کلاس آمده‌است.

- کلاس Table

در این کلاس، ساختار جدول‌ها تعیین می‌شوند. شماره‌های سوئیچ‌هایی که به عنوان وروردی جدول می‌آیند در یک arraylist و شماره‌های variable هایی که به عنوان ورودی می‌آیند، در یک arraylist جداگانه و همچنین یک arraylist از VarLed های خروجی نگه‌داری می‌شود. بعلاوه، یک HashMap از VarLed به آرایه‌ها از اعداد صحیح نگه داری می‌شود که وظیفه‌ی آن نگه‌داری درایه‌های جدول است. بدین‌صورت که به هر VarLed در خروجی یک آرایه نسبت می‌دهد که وضعیت این متغیر را به ازای حالات مختلف ورودی که در جدول آمده‌است نگه‌داری می‌کند. (پس از ساخت هر جدول، انتخاب دوباره‌ی خروجی‌های آن به عنوان خروجی، به لحاظ گرافیکی غیر ممکن می‌شود تا هیچ متغیری در بیش از یک جدول به عنوان خروجی نیاید)

۳-۳ تعیین خروجی های مدار به ازای ورودی‌های مختلف

این کار در تابع evaluateVarLed در کلاس CircController انجام می‌شود. این تابع ، یک VarLed را به عنوان ورودی می‌گیرد و سپس از طریق متغیر outputIn بررسی می‌کند که آیا در هیچ جدولی خروجی بوده‌است یا خیر. اگر نبود ، مقدار default آن که در کلاس Circ ذخیره شده است را به ازای همه‌ی ۲۵۶ حالت به آن نسبت می‌دهیم، اما اگر متغیر، خروجی جدول شماره‌ی i بود، از جدول i ام ، به آرایه وضعیت VarLed مذکور که در HashMap ذخیره شده ، دسترسی پیدا می‌کنیم. تنها کاری که باقی می‌ماند این است که تشخیص دهیم هر ۸ تایی از ورودی‌ها به کدام خانه از آرایه وضعیت VarLed اشاره می‌کند که برای این موضوع صرفا کافی است این تابع را به صورت بازگشتی روی همه‌ی ورودی های از جنس variable جدول صدا بزنیم تا ببینیم هر کدام از variable های ورودی به ازای هر ۸ تایی از ورودی‌ها، چه مقداری می‌پذیرند. (این الگوریتم بازگشتی در صورت loop خوردن خروجی‌ها پایان نمی‌پذیرد و درنتیجه، اگر تعداد باری که تابع صدا زده شد، از حد خاصی بیشتر شد ، متوجه وجود loop می‌شویم و یک پیام خطا به کاربر نشان می‌دهیم)

۴-۳ بدست آوردن رشته‌ی متناظر و نوشتن آن در فایل باینری

در این مرحله، به شیوه‌ای که در بخش پیش توضیح دادیم ، مقدار همه‌ی LED ها که در کلاس Circ ذخیره شده‌اند را با تابع evaluateVarLed ، به ازای همه‌ی حالات ورودی بدست می‌آوریم

و در آرایه status مربوط به object خودشان ذخیره می‌کنیم. (این کار در متد prepareLEDs انجام می‌شود). حال ، برای تعیین رقم i ام از رشته‌ی Hex که قرار است در data.bin نوشته شود ، درایه‌های i ام آرایه‌های status از LED ها را کنار هم می‌چینیم تا یک عدد چهار بیتی(معادلا یک رقم در مبنای ۱۶) تشکیل شود و با توجه به آن یک کاراکتر معادل آن رقم Hex به StringBuilder ای که رشته‌ی نهایی را در خود ذخیره خواهد کرد، append می‌شود. در نهایت stringBuilder را به String تبدیل کرده و string را به آرایه‌ای از بایت‌ها تبدیل کرده و با کمک یک outputStream درون فایل می‌نویسیم.

۳-۵ اجرای کد پایتون

مرحله‌ی آخر است که نیاز به توضیح چندانی ندارد، از ساختار processBuilder های جاوا استفاده می‌کنیم.

۴ رابط گرافیکی برنامه

رابط گرافیکی برنامه از چند منو تشکیل شده است.

۴-۱ منوی اصلی

این منو در واقع منوی آغازین برنامه است. در این منو چند گزینه وجود دارد که به این صورت هستند:

- Add Table

  این المان در واقع یک Button است که با کلیک بر روی آن به منوی Add Table منتقل می‌شویم. در این منو باید جدول‌های مورد نیاز را اضافه کنیم.

- Existing Tables

  این المان نیز یک Button است که با کلیک بر روی آن به منوی Existing Tables منتقل می‌شویم. در این منو لیستی از جدول‌های موجود را مشاهده می‌کنیم.

- Export

  این المان نیز Button است و با کلیک بر روی آن رشته hex مربوط به مدار تولید شده و در فایل data.bin نوشته می‌شود و کد پایتون ران می‌شود. در اینجا دو المان دیگر به نام‌های ledDef و varDef نیز داریم که در واقع ۰ یا ۱ بودن هر کدام از متغیرها و LEDها را در حالتی که توسط مدار مقدار مشخصی برایشان تعیین نشده باشد به طور پیش‌فرض تعیین می‌کند.

۴-۲ منوی اضافه کردن جدول

در این منو ابتدا ورودی‌ها و خروجی‌ها را انتخاب کرده و با کلیک روی دکمه Add Table جدول با ورودی و خروجی‌های مورد نظر ساخته می‌شود. از دکمه‌های Reset و Back نیز به ترتیب می‌توان برای ریست کردن همه انتخاب‌های انجام شده و بازگشت به منوی اصلی استفاده کرد.

۴-۳ منوی لیست جدول‌های ایجاد شده

در این منو صرفا لیستی از جدول‌ها داریم و برای هر جدول دو گزینه Show و Remove وجود دارد. با کلیک روی Remove جدول مورد نظر حذف می‌شود. با کلیک روی Show نیز به منوی نشان دادن

جدول موردنظر منتقل می‌شویم. المان Back نیز وجود دارد که با کلیک بر روی آن به منوی اصلی باز می‌گردیم.

۴-۴ منوی مشاهده یک جدول انتخاب شده

در این منو جدول موردنظر نشان داده می‌شود. ستون مربوط به هر خروجی در ابتدای ایجاد جدول برابر صفر است. برای تغییر هر درایه و ساخت جدولی دلخواه با خروجی مدنظر کافیست روی آن درایه کلیک کنیم تا مقدار آن از صفر به یک و یا از یک به صفر تغییر کند. یک دکمه Back نیز وجود دارد که با کلیک بر روی آن به منوی لیست جدول‌ها باز می‌گردیم. همه متغیرهای ورودی با رنگ آبی و همه متغیرهای خروجی نیز با قرمز نشان داده شده‌اند.