بسمه تعالی

درس طراحی در سطح سیستم

تکلیف کامپیوتری ۳ : آشنایی با Chisel

دانشکدگان فنی دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر
دکتر بیژن علیزاده
نیم‌سال دوم سال تحصیلی ۰۴-۱۴۰۳
نگارش: امید قلی‌زاده (omdgzade۱۳۸۰@gmail.com)

مقدمه

کتابخانه Chisel (Constructing Hardware in a Scala Embedded Language) برپایه زبان Scala به صورت متن‌باز می‌باشد که برای توصیف مدارهای دیجیتال در سطح بالا به کار می‌رود. در این تمرین کامپیوتری سعی داریم تا با نحوه ساخت و اجرای پروژه با استفاده از کتابخانه Chisel آشنا شویم.

بخش اول: نصب ابزارهای مورد نیاز

۱. JDK(Java Development Kit) :

JDK مجموعه‌ای از ابزارهای لازم برای توسعه و اجرای برنامه‌های جاوا است که شامل کامپایلر جاوا (javac) ، ماشین مجازی جاوا (JVM)، کتابخانه‌های استاندارد و سایر ابزارهای کمکی می‌شود. نسخه ۱۱ JDK به عنوان یک نسخه پایدار و سازگار با زبان Scala، بستری را فراهم می‌کند تا کدهای Chisel (که به زبان Scala نوشته می‌شوند) کامپایل و اجرا شوند، زیرا Scala روی ماشین مجازی جاوا (JVM) پایه‌ریزی شده و به ابزارهای JDK وابسته است. این نسخه اطمینان عملکرد پایدار را برای ابزارهای مرتبط با Chisel (مانند FIRRTL و Treadle) فراهم می‌کند.

برای دانلود و نصب، می‌توانید از لینک زیر استفاده کنید:

Download JDK۱۱

۲. IntelliJ IDEA :

IntelliJ IDEA یک محیط توسعه یکپارچه (IDE) قدرتمند برای توسعه نرم‌افزارهای جاوا و زبان‌های دیگر (مانند Scala، Kotlin و Groovy) است که توسط شرکت JetBrains توسعه یافته است. برای پروژه‌های Chisel (که به زبان Scala نوشته می‌شوند)، IntelliJ IDEA با نصب پلاگین Scala امکان ویرایش، کامپایل و اشکال‌زدایی کد را فراهم می‌کند، همچنین با یکپارچه‌سازی SBT (ابزار ساخت Scala) و تنظیم صحیح JDK ۱۱، بستر مناسبی برای توسعه و شبیه‌سازی مدارهای سخت‌افزاری با Chisel ایجاد می‌شود.

برای دانلود و نصب می‌توانید این نرم‌افزار را از سایت soft۹۸.ir دانلود کنید و مراحل نصب و راه‌اندازی را مطابق توضیحات داده شده در سایت انجام دهید:

Download IntelliJ IDEA

۳. GTKWave :

GTKWave یک ابزار رایگان و متن‌باز برای تحلیل و نمایش فایل‌های شبیه‌سازی شده موج (Waveform) مانند VCD (Value Change Dump) است که در پروژه‌های طراحی سخت‌افزاری (مانند آن‌هایی که با Chisel نوشته می‌شوند) کاربرد گسترده‌ای دارد. این ابزار به کاربران اجازه می‌دهد تا تغییرات سیگنال‌های دیجیتال را در طول زمان به‌صورت گرافیکی مشاهده کنند، که در عیب‌یابی و تأیید صحت عملکرد مدارهای طراحی‌شده بسیار حیاتی است. در پروژه‌های Chisel ، پس از اجرای شبیه‌سازی، خروجی به صورت فایل VCD تولید می‌شود و GTKWave به عنوان ابزاری کارآمد برای بازکردن و تحلیل این فایل‌ها عمل می‌کند.

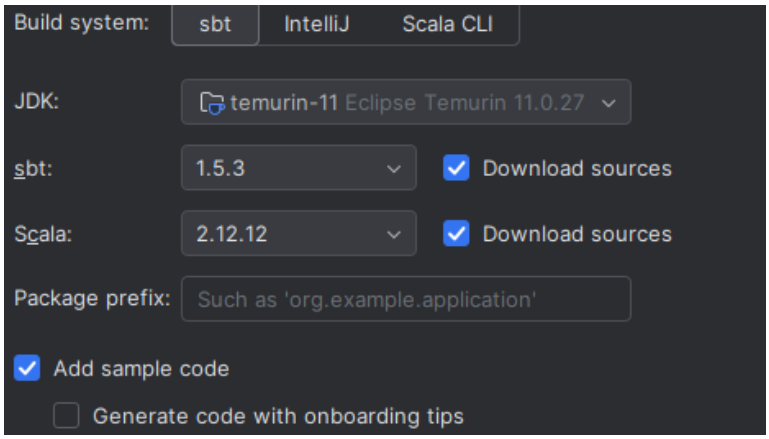
برای نصب و راه‌اندازی این ابزار، می‌توانید از لینک زیر نسخه‌ها و ورژن‌های مختلف از این ابزار را مشاهده کرده و نسخه مناسب سیستم خود را انتخاب کنید:

Download GTKWave

لازم به ذکر است که برای راه‌اندازی هر یک از ابزار فوق و پس از تغییر System Path در Environmental Variable، لازم است که سیستم Reboot شود.

بخش دوم: نحوه ساخت پروژه

پس از نصب و راه‌اندازی ابزار های فوق، اکنون می‌توان یک پروژه جدید به زبان Scala ساخت. نرم‌افزار IntelliJ IDEA را اجرا کرده و پس از انتخاب گزینه New Project، زبان Scala را از tab سمت چپ انتخاب کنید. پارامترهای مربوط به ساخت پروژه را همانند شکل ۱ تنظیم کنید و سپس بر روی گزینه Create کلیک کنید.



شکل ۱- تنظیم پارامترهای پروژه

- توجه کنید که اگر در قسمت انتخاب نسخه JDK گزینه‌ای برایتان فعال نبود، از نصب JDK ۱۱ و از اضافه‌شدن آن به System Path سیستم، اطمینان حاصل کنید. همچنین توجه کنید که پس از زدن گزینه Create، نرم‌افزار سعی دارد تا نسخه‌های تعیین شده برای Scala و sbt را برای پروژه مورد نظر دانلود و نصب کند. بنابراین مطمئن شوید که در این مرحله به اینترنت متصل هستید.
- لازم به ذکر است که برای ساخت و اجرای پروژه به زبان Scala، باید Scala Plugin را در نرم‌افزار دانلود و نصب داشته باشید. برای این امر می‌توانید با رفتن به قسمت File->Settings و انتخاب تب Plugins ، پلاگین مورد نظر را دانلود و نصب کنید.

SBT (Scala Build Tool) ابزار استاندارد مدیریت پروژه‌های Scala (و Chisel) است که فرآیندهایی مانند کامپایل، مدیریت وابستگی‌ها، تست و ساخت پروژه را اتوماتیک می‌کند. تنظیمات اصلی پروژه در فایل build.sbt نوشته می‌شوند، جایی که نام پروژه، نسخه، وابستگی‌های لازم ، گزینه‌های کامپایل و پلاگین‌های مورد نیاز تعریف می‌شوند. SBT با خواندن این فایل، محیط توسعه را برای کار با کد Scala و ابزارهای مرتبط (مانند ChiselTest برای تست شبیه‌سازی) پیکربندی می‌کند و دستوراتی مانند sbt compile یا sbt test را اجرا می‌کند. این سیستم انعطاف‌پذیر و ساده، توسعه پروژه‌های سخت‌افزاری با Chisel را کارآمد می‌کند.

پس از راه‌اندازی یک پروژه Scala (به‌ویژه پروژه‌های Chisel) با استفاده از ابزارهایی مانند SBT، ساختار دایرکتوری‌های استاندارد زیر ایجاد می‌شود تا کد و منابع پروژه به‌صورت سازمان‌یافته مدیریت شوند. این ساختار به IntelliJ IDEA و فرآیندهای کامپایل/تست کمک می‌کند تا فایل‌ها را به درستی شناسایی کنند.

دایرکتوری src :

این دایرکتوری شامل تمام کدهای منبع پروژه است و دو زیردایرکتوری اصلی دارد:

۱. src/main/scala:

- کد تولیدی (Production Code) :

تمام کلاس‌ها، ماژول‌ها و کامپوننت‌های Chisel (مانند Module, Bundle) در این مسیر قرار می‌گیرند.

- ساختار بسته‌بندی (Package Structure)

۲. src/test/scala:

- کد تست (Test Code) :

تست‌های واحد (Unit Tests) و تست‌های شبیه‌سازی (Simulation Tests) با استفاده از کتابخانه‌هایی مانند ChiselTest و یا iotesters در این مسیر نوشته می‌شوند.

دلیل اهمیت این ساختار منظم دایرکتوری‌ها، سازگاری با SBT و IntelliJ IDEA می‌باشد.

```
my-chisel-project/
├── build.sbt
├── src/
│   ├── main/
│   │   └── scala/
│   │       └── mypackage/
│   │           ├── MyModule.scala
│   │           └── MyALU.scala
│   └── test/
│       └── scala/
│           └── mypackage/
│               ├── MyModule.scala
│               └── MyALU.scala
└── project/
    └── build.propertieslass
```

شکل۲- ترتیب دایرکتوری‌ها در پروژه

حال باید فایل build.sbt را اجرا کنیم تا کتابخانه‌های مورد نیاز دانلود شوند. برای فایل Build.sbt، از قطعه کد شکل ۳ استفاده کنید:

```
name := "PROJECT_NAME"

version := "0.1"

scalaVersion := "2.12.12"

scalacOptions := Seq("-deprecation", "-Xsource:2.11")

resolvers ++= Seq(
  Resolver.sonatypeRepo("snapshots"),
  Resolver.sonatypeRepo("releases")
)
libraryDependencies += "edu.berkeley.cs" %% "chisel3" % "3.4.2"
libraryDependencies += "edu.berkeley.cs" %% "chisel-iotesters" % "1.5.1"
libraryDependencies += "edu.berkeley.cs" %% "chiseltest" % "0.3.1"
```

شکل۳- کد Build.sbt

در این کد، نسخه‌ها و مخازن مرتبط به این نسخه‌ها به SBT معرفی شده تا بتواند نسخه‌های مدنظر را دانلود کند.

پس از کپی کردن کد در Build.sbt، گزینه Sync SBT Changes را انتخاب کرده و منتظر بمانید تا پیام Success در ترمینال مربوط به Build چاپ شود. اگر عملیات موفقیت‌آمیز باشد، می‌توانید کتابخانه‌های دانلود شده را در بخش External Libraries مشاهده کنید.

بخش سوم: پیاده‌سازی یک ALU

حال می‌خواهیم یک ماژول ALU (Arithmetic Logic Unit) بسازیم. این ماژول عملیات جمع، تفریق، شیفت به راست و منطقی را انجام خواهد داد. همچنین این ماژول باید پارامتری باشد و تعداد بیت‌های ورودی و خروجی به صورت پارامتری نوشته شود.

جدول ۱- ریزدستورهای واحد ALU

Operation	Function	Opcode
NOP	Res = A	۰۰۰
Add	Res = A + B	۰۰۱
Sub	Res = A - B	۰۱۰
And	Res = A and B	۰۱۱
Or	Res = A or B	۱۰۰
Xor	Res = A xor B	۱۰۱
Ld	Res = B	۱۱۰
Shr	Res = A>>۱	۱۱۱

پس از تعریف ماژول ALU و تعریف عملکرد آن می‌توانید با استفاده از کد قابل اجرای شکل۴، خروجی Verilog ماژول تعریف شده را دریافت کنید:

```
object ALUMaker extends App {
  println ("(Generating Verilog file for ALU)")
  (new chisel3.stage.ChiselStage).emitVerilog(new Your_ALU_Module(n), Array("--target-dir",
"Verilog"))
}
```

شکل۴-کد ساخت خروجی Verilog

توجه داشته باشید که بجای Your_ALU_Module ، نام ماژولی که تعریف کرده‌اید را قرار داده و بجای n، مقدار ۱۶ را قرار دهید تا خروجی Verilog حاصل یک ۱۶-bit ALU باشد.

برای این ماژول در دایرکتوری مربوطه(src/test/scala)یک تست‌بنچ با استفاده از کتابخانه‌های iotesters و یا ChiselTest، بنویسید و با استفاده از قطعه کد شکل ۵، علاوه بر بررسی نتایج مورد انتظار، فایل vcd. مربوط به این تست را تولید کنید:

```
object ALUTester extends App{
  println("(Running TestBench for ALU)")
  chisel3.iotesters.Driver(() => new Your_ALU_Module(n)) {c =>
    new Your_ALU_TB(c)
  }
  chisel3.iotesters.Driver.execute(Array("--generate-vcd-output", "on"), () => new
Your_ALU_Module(n)) {
    c => new Your_ALU_TB(c)
  }
}
```

شکل۵-کد ساخت خروجی VCD و تست کد تست‌بنچ

❖ توجه کنید که تغییرات لازم در کد همانند کد قبل اعمال شود.

همچنین یک تست‌بنچ به زبان Verilog معادل تست‌بنچ Chisel خود بنویسید و تشابه‌ها و تفاوت‌های موجود در دو شبیه‌سازی را با نمایش WaveForm های مربوط به آن گزارش کنید.

نکات پایانی:

۱. پوشه src و فایل pdf گزارش‌کار خود را در یک فایل زیپ‌شده به نام ESL_CA۳_<StudentId>.zip ارسال نمایید که در آن <StudentId> همان شماره دانشجویی شماست.
۲. در فایل Build.sbt، متغیر name را به صورت CA۳_<StudentId> مقدار داده که که در آن <StudentId> همان شماره دانشجویی شماست.
۳. تمامی کدها باید خوانا بوده و کامنت‌گذاری مناسب داشته باشد.
٤. گزارش‌کار باید یک توضیح جامع از کل پروژه و حتما شامل توضیح بخش های رنگ شده باشد.
٥. در صورت داشتن هرگونه ابهام یا سوال، می‌توانید از طریق ایمیل (omdgzade۱۳۸۰@gmail.com) یا تلگرام (@OMD_Gzade) با بنده در ارتباط باشید.

بارم‌بندی سوالات

- بخش ۱ : ۱۰ نمره
- بخش ۲ : ۲۰ نمره
- بخش ۳ : ۷۰ نمره
- تمیزی گزارش و مرتب بودن فایل‌های آپلود شده نمره امتیازی دارد.

موفق و سلامت باشید.