

به نام خدا



پروژه مدار منطقی پیشرفته

استاد اجلالی

مبدل RTL به Verilog

شبیم شیخها

۹۵۱۰۵۶۷۹

الف) ویژگی‌های برنامه

مبدل به زبان جاوا نوشته شده است. فرض شده است که کد RTL ورودی این شرایط را دارد:

۱. در خط اول همه‌ی رجیسترها و فلیپ‌فلاپ‌ها مقداردهی اولیه بشوند. فلیپ‌فلاپ‌های واحد کنترل هستند مقدار صفر، رجیسترهای مسیر داده مقدار ورودی و یا مقدار صفر و ... را بگیرند. مانند زیر:

$R1 \leftarrow in1, R2 \leftarrow in2, QR \leftarrow 0, V \leftarrow 0, Err \leftarrow 0, R \leftarrow 0, Q \leftarrow 0, F1 \leftarrow 1$

۲. در خط آخر خروجی‌ها مقداردهی شوند. مانند زیر:

$V \leftarrow 1, Q \leftarrow QR, R \leftarrow R1, F1 \leftarrow 0, Err \leftarrow 0$

در اینجا رجیسترهای Q و R خروجی هستند و تنها یکبار در خط آخر مقداردهی می‌شوند.

به عنوان نمونه، کد rtl.txt قرار داده شده است.

لازم به ذکر است که صرفاً کافی است فرایض بالا رعایت شوند. نیاز به رعایت هیچ‌گونه فرض دیگری مانند رعایت ترتیب در سایر خطوط برنامه و ... نیست. در صورت رعایت این فرایض مبدل ویژگی‌های زیر را خواهد داشت:

۱. رجیسترهای ورودی، رجیسترهای خروجی، رجیسترهای مسیره‌داده، فلیپ‌فلاپ‌های مسیر داده (Flagها) و فلیپ‌فلاپ‌های واحد کنترل را تشخیص می‌دهد.

۲. به صورت عادی، سائز همه‌ی رجیسترها ۱۶ بیت است. ولی کاربر میتواند سائز هر یک را به دلخواه خود تغییر دهد.

۳. هیچ‌گونه محدودیتی در استفاده از عملگرها ندارد.

۴. اولویت عملگرها را، چه با پرانتزگذاری و چه بدون پرانتزگذاری تشخیص می‌دهد.

۵. عملیات زیر بین رجیسترها را در مسیره داده پشتیبانی میکند:

- $OR(reg_name)$:

عملیات از این نوع را Reduction Or تلقی میکند و به $lreg_name$ تبدیل می‌کند.

- $L(R1, R2)$

سایر عملیات مقایسه مانند $G(R1, R2)$ و $E(R1, R2)$ را نیز پشتیبانی می‌کند.

- $reg_name[i]$


دسترسی به بیت‌های مختلف یک رجیستر را نیز پشتیبانی می‌کند.

۵. نسبت به white space های زائد حساس نیست.


لازم به ذکر است که در هیچ یک از قسمت‌های این مبدل، از کد و یا توابع آماده، از پیش ساخته شده و یا موجود در اینترنت استفاده نشده است.

ب) نحوه‌ی استفاده از رابط گرافیکی:


همانطور که گفته شد برنامه به زبان جاوا نوشته شده است. بعد از اجرای برنامه ابتدا صفحه‌ی فهرست شامل سه دکمه‌ی Start، Help و Exit باز می‌شود. با انتخاب دکمه‌ی Start وارد صفحه‌ی جدیدی می‌شوید. در اینجا دو گزینه پیش رو دارید:


۱. با انتخاب دکمه‌ی  فایل txt. شامل کد RTL را انتخاب کنید.

۲. در قسمت خاکستری رنگ، کد خود را دستی وارد کنید.


بعد از وارد کردن کد به هر کدام از روش‌های گفته شده، دکمه‌ی  را فشار دهید.

پس از فشار دادن دکمه فیلدهای جدیدی را مشاهده خواهید کرد. اینها برای وارد کردن سائز دلخواه رجیسترهای مسیر داده است. سائز دیفالت هر رجیستر ۱۶ بیت است.

در صورت تغییر سائز، پس از وارد کردن سائز موردنظر دکمه‌ی  را انتخاب کنید.

در انتها پس از تعیین سائز رجیسترها دکمه‌ی  را انتخاب کنید.

در قسمت خاکستری رنگ ایجاد شده، کد Verilog موردنظر قرار دارد.

در صورتی که می‌خواهید کد را ذخیره کنید، ابتدا دکمه‌ی  را فشار دهید. سپس اسم و پسوند دلخواه خود را بنویسید.

ج) توضیح قسمت‌های مختلف برنامه:

در این قسمت، توابع مختلف برنامه را شرح خواهم داد.

- assign()

در این تابع ورودی‌ها، خروجی‌ها، رجیسترهای مسیر داده، فلیپ‌فلاپ‌های مسیر داده و فلیپ‌فلاپ‌های واحد کنترل تشخیص داده می‌شوند.
این تابع از توابع زیر کمک می‌گیرد:

- assign_DP_parts();
- assign_DP_reg_FF();
- assign_CU_Flipflops();
- assign_User_inputs();
- assign_outputs();
- assign_widths();

- convert_to_verilog()

در این تابع، پس از مشخص شدن رجیسترهای مسیر داده و ... با استفاده از تابع assign()، تبدیل کد RTL به Verilog صورت می‌گیرد.
این تابع از توابع زیر کمک می‌گیرد:

- module_deceleration();
- variable_assignment();

لازم به ذکر است که در این قسمت، رجیسترهای از یک نوع (مثلاً همگی ورودی باشند) با ساینز یکسان در یک خط تعریف می‌شوند. یعنی کد زیر:

```
reg [15:0] r1;  
reg [15:0] r2;
```

دیده نمی‌شود. بلکه:

```
reg [15:0] r1, r2;
```

دیده می‌شود.

- `initial_block();`
- `always_block();`
- `createMainStageContent() / createMenuContent()`

صرفا جهت ایجاد رابط گرافیکی استفاده شده‌اند.