به نام خداوند جان و خرد

پروژه‌ی پایانی

گزارش کار

بازی حدس اعداد

درس:

مدارهای منطقی

استاد:

استاد حاتم عبدلی

کمک‌استاد:

خانم بهار حمیدی‌محب

نویسنده:

محمد امین چیتگرها

(۹۸۱۲۳۵۸۰۱۵)

زمستان ۱۳۹۹

اسفند ماه

ساختار فایل‌ها

در پروژه‌ی کنونی، که نام آن حدس‌آلود است، برای دسته‌بندی بهتر کدها، پوشه‌های گوناگونی دیده می‌شوند. کدهای منبع، شامل عنصرهای (entity) اصلی برنامه (مانند مقایسه‌کننده یا شمارنده) و درون پوشه‌ی src هستند. فایل‌های تست‌بنچ نیز، در پوشه‌ی tests/unit قرار دارند. تلاش شده است برای تقریبا همه‌ی عنصرها، تست‌بنچ جداگانه و مستقل نوشته شود. هم‌چنین، فایل‌های VHDL درون این دو پوشه، خود در پوشه‌هایی خوشه‌بندی شده‌اند. هم‌چنین، پوشه‌ی scripts، شامل اسکریپت‌هایی به زبان PHP است که کار آزمون و ساخت فایل‌ها را ساده می‌سازد (و در نتیجه، جزو اصل پروژه نیست).

نام همه‌ی عنصرها، از قاعده‌ی `snake_case` پیروی می‌کنند. درون هر فایل، دقیقا یک عنصر موجود است و نام‌های فایل‌ها، به تناسب عنصر درون‌شان گزیده شده‌اند. همه‌ی عنصرهای تست‌بنچ، با پیشوند `test_` آغاز می‌گردند.

پیاده‌سازی

در تعریف عنصرها، کلی‌نگری در نظر گرفته شده است و سپس به تناسب نیاز، از آن عنصر، نسخه‌ی ویژه‌ی مورد نیاز ساخته شده است. برای نمونه، در این پروژه تنها نیاز به شمارنده‌ی ۳بیتی است؛ اما نسخه‌ی کلی nبیتی آن پیاده‌سازی شده است.

عنصرهای زیر در پروژه قابل‌مشاهده‌اند:

- **لتچ‌ها و فلیپ‌فلاپ‌ها:** این عنصرها، برای استفاده در عنصرهای دیگر پیاده‌سازی شده‌اند (که البته برخی بی‌استفاده مانده‌اند). مهم‌ترین آن‌ها، فلیپ‌فلاپ تی ساعت‌بالارونده است که برای آن یک بیت فعال‌سازی (enable) و یک بیت پاک‌سازی (clear) ناهمگام (asynchronous) در نظر گرفته شده است.
- **مقایسه‌کننده‌ی nبیتی:** این مقایسه‌کننده، به‌شکل ساختاری و به‌کمک n مقایسه‌کننده‌ی تک‌بیتی ایجاد شده است. ورودی آن، دو عدد است و سه خروجی دارد که به ترتیب نشان‌دهنده‌ی بزرگ‌تر بودن، برابری و کوچک‌تر بودن عملوند سمت چپ از عملوند سمت راست است. برای ساده‌سازی پیاده‌سازی، این خروجی‌ها از یک‌دیگر استفاده می‌کنند؛ برای نمونه، کوچک‌تر بودن همسان با بزرگ‌تر نبودن و نابرابری است.
- **شمارنده‌ی nبیتی:** این عنصر به‌کمک مجموعه‌ای از فلیپ‌فلاپ‌های تی ساعت‌بالارونده و به‌شکل ساختاری پیاده شده است. به کمک بیت‌های پاک‌سازی ناهمگام در این فلیپ‌فلاپ‌ها، شمارنده نیز دارای بیت پاک‌سازی ناهمگام با ساعت می‌باشد. هم‌چنین، این عنصر دارای ورودی فعال‌سازی نیز هست.

- **رجیستر nبیتی:** دارای بیت‌های فعال‌سازی و پاک‌سازی ناهمگام. این سبب می‌شود بتوان در هر زمان محتویات این عنصر را پاک نمود. این عنصر رفتاری پیاده‌سازی شده است.
- **تبدیل‌کننده‌ی باینری به سون‌سگمنت:** این تبدیل‌کننده، بر خلاف موردهای پیشین، تنها برای ورودی ۳بیتی کار می‌کند.
- **مدیریت‌کننده‌ی ال‌ای‌دی‌ها:** این عنصر، همه‌ی ال‌ای‌دی‌های خروجی را مدیریت می‌نماید؛ شامل نمایش‌دهنده‌های وضعیت حدس کاربر نسبت به عدد تصادفی تولیدی (سه بیت که در هر زمان دقیقا یکی از این‌ها روشن خواهد بود)، و این‌که آیا به شمار حدس‌های مجاز رسیده شده است یا خیر. با یک ورودی فعال‌سازی، این عنصر می‌تواند در هر زمانی همه‌ی چراغ‌های خود را خاموش یا روشن نماید. مزیت این کار، برای نمونه، خاموش کردن ال‌ای‌دی‌ها پیش از رسیدن مدار به حالت پایدار است.
- **تولیدکننده‌ی عدد تصادفی nبیتی:** این تولیدکننده‌ی همگام با ساعت، برای تصادفی بودن هر چه بیش‌تر خروجی آن، به‌شکل پیچیده‌ای پیاده‌سازی شده است. خلاصه‌وار، این عنصر یک بذر (seed) به‌اندازه‌ی دلخواه را گرفته، و سپس بذر بعدی را از روی این بذر، با توجه به الگوریتم ویژه‌ای، می‌سازد؛ که البته از ساعت گرفته‌شده نیز برای مقدار بعدی استفاده می‌شود.

  این تولیدکننده، در هر مرحله، دارای یک وضعیت ویژه است، که بسیار به بذر داده‌شده بستگی دارد. در هر وضعیت، مجموعه‌ای از انتخاب‌کننده‌ها وجود دارد که برای بهم‌ریختن و جابجایی بذر کنونی برای ساختن وضعیت و بذر بعدی، به کار می‌رود. مقدار بعدی این انتخاب‌کننده‌ها، مانند سلسله اعداد فیبوناچی پیش می‌رود (عدد جدید = جمع همه‌ی عددهای پیشین).

  این الگوریتم، به‌ازای هر بذر یکتا، توزیع یکتا و گوناگونی را می‌سازد. کلید این وابستگی به بذر، این است که انتخاب‌کننده‌ها پس از مدت زمانی، با توجه به بذر اولیه، از نظر اندازه‌ای و نیز مقداری تغییر می‌کنند.

  این عنصر، در ابتدا، به شکل یک تولیدکننده‌ی عدد تصادفی تک‌بیتی بود که با توزیع نسبتا خوب و قابل‌قبولی میان صفر و یک جابجا می‌شد (و می‌شود؛ بخش تصویرهای نمونه از شبیه‌سازی‌ها را ببینید). اما، از آن‌جایی که ساختن یک تولیدکننده‌ی به همین مقدار تصادفی nبیتی از روی آن پیچیده و زمانبر بود، از یک‌بار شیفت دادن خروجی به چپ و جای‌گذاری بیت تصادفی جدید در کم‌ارزش‌ترین بیت، استفاده شده است. با (و حتی بدون) درنظرگیری یک‌دسته بیت مرتب کنار هم در دوره‌های زمانی ثابت، توزیع این تولیدکننده نیز اگرچه عالی نیست، ولی قابل‌قبول و حتی به‌ازای بذرهای مناسب، از کیفیت مناسبی برخوردار است.

  لازم به ذکر است که تصادفی‌شدن عددهای تولیدشده با این الگوریتم نیازمند یک کمینه زمان است. به همین دلیل، بیت `done` تعریف شده است که به محض پایداری خروجی تولیدشده، این بیت نیز فعال خواهد شد.

- **تولیدکننده‌های سیگنال‌های تک‌بیتی:** دو تولیدکننده‌ی سیگنال برای استفاده در تست‌بنچ‌ها ساخته شده است. اولی برای تولید ساعت (clock)، به دو شکل محدود و نامحدود است. دومی نیز، برای جابجایی میان مقدارهای صفر و یکِ یک تک‌بیت، در زمان‌های گوناگون است. این دو عنصر، به همراه یک بسته‌ی (package) دیگر که یک نوع ویژه (`time_array`) را، برای استفاده در عنصر اشاره‌شده‌ی پایانی، تعریف می‌کند، در خوشه‌ی utils تعریف شده‌اند.
- **عنصر اصلی (`main`):ـ** این عنصر، پیونددهنده‌ی (تقریبا) همه‌ی عنصرهای پیشین برای ساخت یک بازی حدس عدد است. دو بیت، به عنوان دو دکمه‌ی فیزیکی مجازی، برای ثبت عدد واردشده‌ی کاربر (`enter_button`) و نیز تنظیم‌مجدد وضعیت بازی (`reset_button`)، به عنوان ورودی پویای این عنصر تعریف شده است. هم‌چنین، خروجی‌های این برنامه نیز همان ال‌ای‌دی‌ها و نیز خروجی سون‌سگمنت است.

  برنامه تا زمانی که کاربر عدد وارده‌ی خود را ثبت کند، کاری انجام نمی‌دهد و کلیت برنامه در حالت غیرفعال قرار دارد. به محض تایید کاربر، عدد کنونیِ تولیدشده از تولیدکننده‌ی تصادفی، با توجه به زمان کنونی، گرفته شده و در رجیستر مربوطه ذخیره می‌گردد؛ تا زمانی که کاربر درخواست تنظیم‌مجدد بازی را بدهد.

  محدودیت ۷ حدس کاربر و خروجی سون‌سگمنت، به‌کمک شمارنده‌ی موجود در این عنصر، مدیریت می‌شود. اگر این شمارنده به محدودیت خود برسد، مدار غیرفعال شده و ورودی جدیدی نمی‌پذیرد. فعالیت درونی مدار، به کمک سیگنال `enable_global` مدیریت می‌گردد؛ که این سیگنال با ساعت مدار همگام می‌باشد.

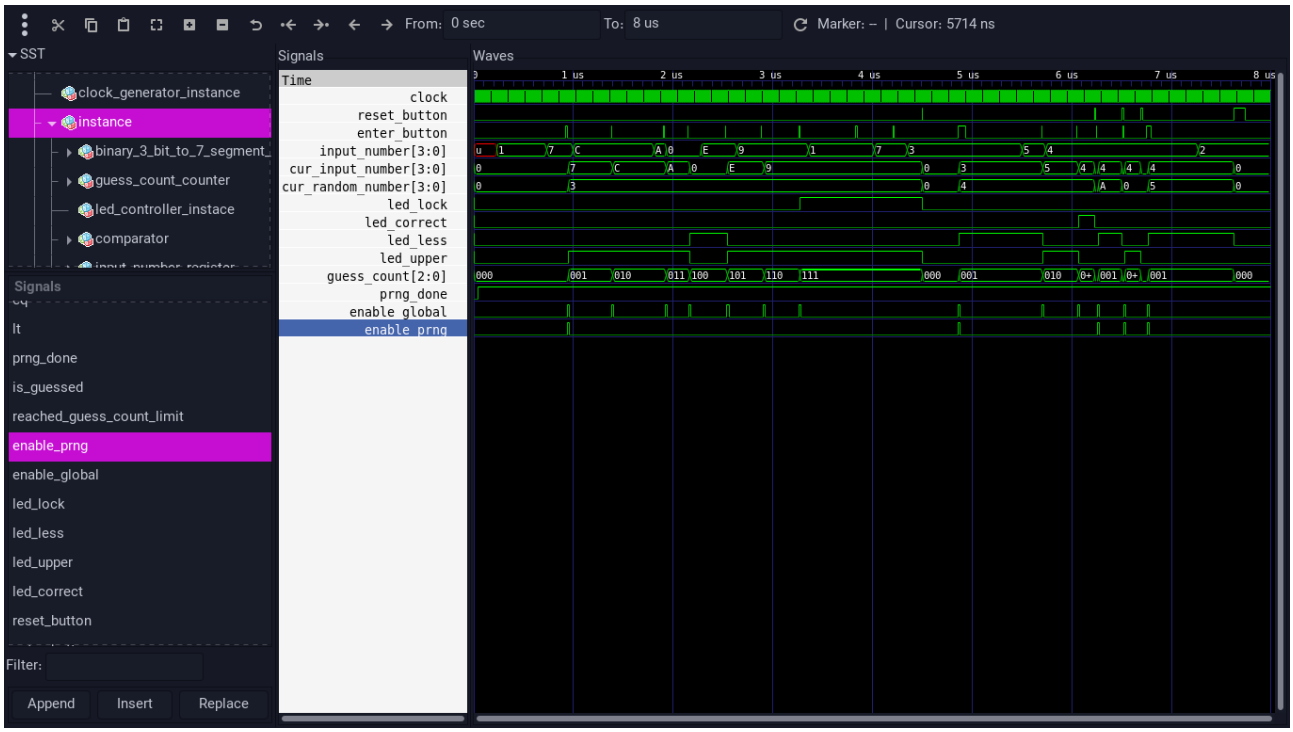
  هم‌چنین، در هر وضعیتی، با فعال‌سازی بیت تنظیم‌مجدد، می‌توان مدار را تقریبا به حالت اولیه بازگرداند؛ چون این بیت با ساعت مدار ناهمگام است و به همه‌ی عنصرهای درونی با قابلیت پاک‌سازی ناهمگام، متصل شده است. تنها چیزی که با این بیت یا بیت فعال‌سازی تغییر نمی‌کند، وضعیت تولید عددهای تصادفی است. به محض شروع کار مدار، تولید عددهای تصادفی شروع شده و تا پایان کار مدار، این عملیات ادامه خواهد یافت.

ابزارهای استفاده‌شده

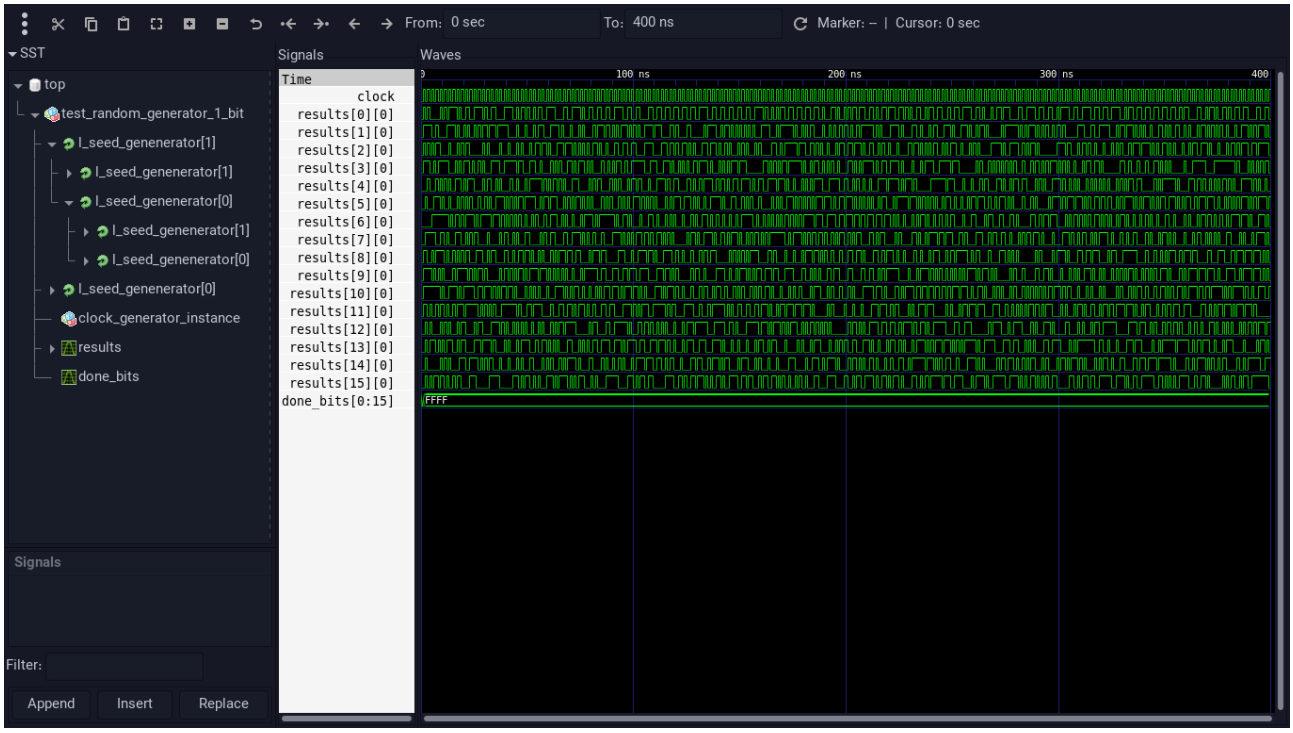
از گیت برای توسعه‌ی این پروژه استفاده شده است؛ شما می‌توانید تاریخچه‌ی همه‌ی فعالیت‌ها را از ابتدا تا به کنون ببینید. برای شبیه‌سازی، از نرم‌افزارهای GHDL و GtkWave بهره برده شده است.

همان‌گونه که پیش از این نیز گفته شد، دو اسکریپت پرکاربرد برای خودکارسازی مرحله‌ی توسعه و نیز آزمایش عنصرها به زبان PHP نوشته شده است (که می‌توانید آن‌ها را نیز امتحان کنید). یکی از اسکریپت‌ها، فایل‌های VHDL را می‌سازد؛ و دیگری که بسیار پرکاربردتر است، با آنالیز خودکار فایل‌های مورد نیاز، با توجه به استانداردهای موجود در پروژه، یک تست‌بنچ را شبیه‌سازی می‌کند.

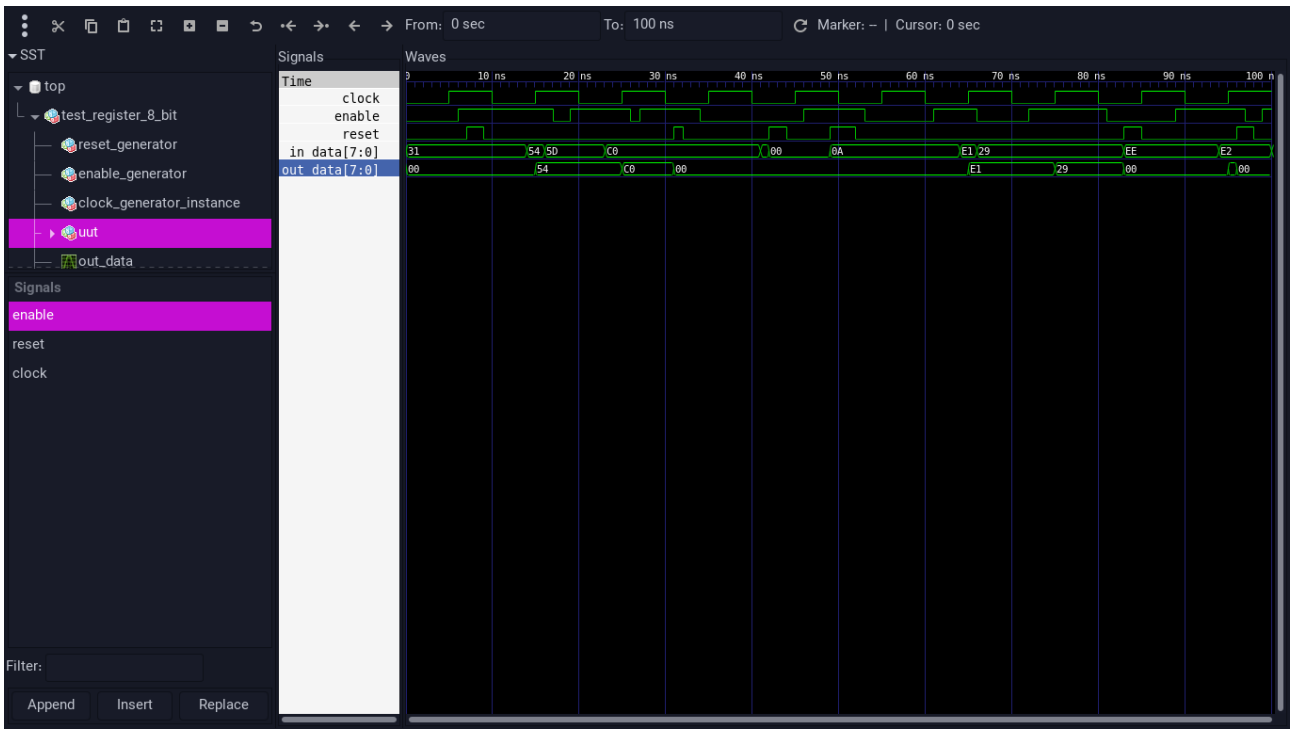
تصویرهای نمونه از شبیه‌سازی‌ها



شبیه‌سازی `main`: کاربری فرضی در حال تعامل با بازی و وارد کردن اطلاعات. فرض شده است کاربر در جاهایی اشتباه می‌کند؛ مثل این‌که برای یک ورودی، دو بار عدد ورودی خود را ثبت می‌کند. هم‌چنین، ساعت به دلیل بسامد بالا، یکنواخت دیده می‌شود.



شبیه‌سازی یک تولیدکننده‌ی عدد تصادفی تک‌بیتی. اندازه‌ی بذر، ۴ در نظر گرفته شده است و همه‌ی بذرهای ممکن ($۲^۴$ = ۱۶ عدد) تولید و آزمایش شده‌اند. می‌توانید کیفیت گوناگونی و یکتایی مقدارهای تولیدشده را ببینید؛ یافتن شباهت میان‌شان دشوار است و این نشان از کیفیت خوب خروجی دارد.



شبیه‌سازی یک رجیستر ۸بیتی (n = ۸). در این‌جا، بیت‌های enable و reset به‌کمک عنصر تولیدکننده‌ی بیت جابجاشونده (switching_signal_generator) تولید شده‌اند.

دیگر موارد...

پروژه‌ی حدس‌آلود، تحت گواهی‌نامه‌ی GPL، نسخه‌ی ۳ (یا بالاتر) انتشار یافته است که نسخه‌ای از آن در فایل‌های پروژه موجود می‌باشد.

موفق باشید.