به نام خدا

پروژه‌ی نهایی درس طراحی سیسیتم های دیجیتال

هومن هاشمی 92100415  
شراره میرزرگر 92106283  
مهدی بهروزی‌خواه 92100386

**شرح مختصر:**

در این پروژه هدف پیاده‌سازی یک شبیه‌ساز برای شبکه های روی چیپ است. این شبیه ساز ساختار شبکه و ترافیک آن را از روی فایل دریافت می کند و سپس تعداد سیکل هایی که برای انتقال این ترافیک بر روی این شبکه لازم است را محاسبه می‌کند.

**محیط پیاده‌سازی:**

برای پیاده‌سازی این شبیه‌ساز از زبان وریلاگ و نرم افزار سنتز ISE استفاده شده است. ساختار ماژولار و ماژول های استفاده شده در این شبیه ساز در ادامه توضیح داده خواهند شد. نحوه‌ی خواندن ورودی ها از فایل های hex تولید شده از روی کانفیگ است که درون ROM ذخیره می‌شوند و نحوه‌ی اعلام پایان شبیه سازی و نوشتن خروجی ها به وسیله‌ی پایه های خروجی ماژول اصلی است.

**ماژول Router:**

**خلاصه**:

این ماژول همانند فایل cpp که در رفرنس وجود داشت (router.cpp) عمل می‌کند و قرار است مانند یک روتر عمل کند. از جمله کار‌هایی که می‌کند، routing table را پیاده‌سازی ‌می‌کند و با توجه به آن مسیر یابی می‌کند و از روتر های دیگر با flit هایی می‌گیرد و در پورت‌های خروجیش flit های مسیر یابی شده را می‌دهد.

**جزئییات**:

در ابتدا این ماژول تعداد پورت‌های ورودی و خروجیش را از ورودی می‌گیرد و همچنین تأخیر credit و تعداد کانال‌های مجازی ای که از فایل می‌خوانیم را به این ماژول می‌دهیم. سپس جدول روتینگ را در هر کلاک به این ماژول می‌دهیم تا تمام جدول را بگیرد و پس از آن ماژول آماده استفاده است.

پس از این ماژول سه حالت اصلی دارد که در آن‌ها می‌چرخیم. قسمت اولی هنگامی است که ورودی های هر پورت را از پردازنده یا روتر دیگری میگیریم و در پورت ورودی این ماژول قرار می‌دهیم. قسمت بعدی روتر ورودی های گرفته شده را در بافر vc مربوطه قرار می‌دهد و در قسمت آخر مسیر یابی با توجه به جدولی که ابتدا گرفته بود انجام می‌شود. همچنین برای اینکه بدانیم کدام vc خالی شده، هر روتری در جهت عکس، یک credit می‌فرستد و در فاز آخر خروجی و این کردیت آماده هستند.

برای روتر هر کدام از این کار‌ها را انجام دهد، باید چیزی آن را از بیرون کنترل کند که در کد ما main این کار را انجام می‌دهد و دستورات مختلفی را به این ماژول می‌دهد. دستورات مختلفی که انجام می‌دهد به شرح زیر است:

* **Init**: در این وضعیت ورودی های اولیه را مانند تعداد پورت ها و تعداد کانال های مجازی را می‌گیرد و آماده می‌شود.
* **LoadStaging:** در این وضعیت بسته ای را که در ورودی اش قرار دارد می‌خواند و در داخل ماژول نگه داری می‌کند.
* **Phase0**: در این وضعیت بسته ای ماژول از ورودی خوانده بود را در بافر مربوطه پورتش که به ازای هر پورت به تعداد vc ها بافر داریم، می‌ریزد.
* **Phase1:** در فاز عملیات روتینگ انجام می‌شود و با توجه به روتر مقصد، پورت خروجی را پیدا پیدا می‌شود و بسته را در پورت خروجی مربوطه قرار می‌گیرد.

**ماژول traffic:**

**خلاصه**:

این ماژول همانند فایل cpp که در رفرنس وجود داشت عمل می‌کند و در‌واقع قرار است نقش پردازنده را ایفا کند. این ماژول مانند یک صف عمل می‌کند و به ترتیب فلیت هایی که از فایل خوانده ایم را در خروجی می‌دهد.

**جزئییات**:

همانطور که گفته شد این ماژول مانند یک صف عمل می‌کند. پس ابتدا باید پاکت هایی که از فایل خوانده‌ایم را در ورودی به این ماژول می‌دهیم، به طوری که هر clock یک پکت و سپس در ادامه از خروجی آن برای ورودی دادن به پورت ۰ روتر مربوطه‌اش استفاده می‌کنیم. هرگاه میتوانستیم این انتقال را انجام دهیم، اول صف را به بیرون می‌ریزیم و خروجی به بعدی اشاره می‌کند. همچنین این ماژول به صورت ضمنی پکت ها را به فلیت تبدیل می‌کند و موقع خروجی دادن بجای اینکه به پکت بعدی برود، به فلیت بعدی می‌رود.

برای کنترل این وضعیت ها، این ماژول مانند یک ماشین وضعیت است و از آن جایی که این صف باید از بیرون کنترل شود، یک وضعیت از بیرون می‌گیرد و با توجه به آن وضعیت طبق برنامه اش عمل می‌کند. انواع وضعیت‌هایی که می‌تواند قرار بگیرد در زیر آمده است:

* **Init**: در این وضعیت تعداد پاکت‌هایی که در آینده قرار است در این صف قرار بگیرد را می‌گیرد.
* **Fill:** هر بار که در این وضعیت قرار می‌گیرد، ورودی را که یک پکت است، می‌خواند و در صف خودش ذخیره می‌کند.
* **PreDeque**: پس از اینکه روتر پر شد، باید خروجی آن را به اولین عنصر صف اشاره دهیم، برای اینکار لازم است یک بار روتر در این وضعیت قرار گیرد تا خروجیش درست شود.
* **Dequeue:** این وضعیت به این معنی است که باید به flit بعدی اشاره کنیم، برای این منظور چک می‌کنیم که اگر همین پکتی که در حال فرستادن آن بودیم، flit دیگری داشت، آن را می‌فرستیم، در غیر این صورت، به پکت بعدی می‌رویم و اولین flit آن را می‌فرستیم.

**ماژول main:**

**خلاصه**:

این ماژول در‌واقع نقش ارتباط بین روتر ها و پردازنده هر روتر با روتر مربطه اش را انجام می‌دهد. در‌واقع ورودی ها و خروجی های هر پورت روترها را از روتر یا پردازه ای که باید می‌گیرد و در روتر مقصد می‌گذارد. همچنین تمام instance های روترها و پردازنده‌ها درون این ماژول قرار دارند و در نتیجه فایل‌های ورودی را نیز این ماژول می‌خواند و پردازش می‌کند.

**جزئییات**:

این ماژول ابتدا ورودی‌ها را از فایل می‌خواند. برای این قسمت سعی کردیم از دستور $readmemh استفاده کنیم تا قالبیت سنتز داشته باشد. سپس اطلاعات خوانده شده را در آرایه هایی دسته بندی می‌کنیم. پس از این به اندازه حداکثر روتر هایی که در فایل پارامتر مشخص شده با استفاده از generate block ، از router و traffic نمونه می‌گیریم. هر کدام از این نمونه‌ها با یک state machine کار میکنند و state آن‌ها از طریق این ماژول بیرونی main به وسیله ورودی که از پورت op اشان می‌گیرند، کنترل می‌شود.

پس از این فرآیند‌ها این ماژول مانند یک state machine عمل می‌کند که سر هر لبه clock یا reset تغییر وضعیت می‌دهد. حالت‌های آن و فعالیت‌های مربوط به هر بخش این‌گونه است:

* **InitState**: این حالت اولیه ایست که هر بار اگر reset ۱ شود به این حالت باز‌می‌گردیم و تمام کار‌های اولیه را انجام می‌دهد مانند ورودی خواندن از فایل.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت nop می‌بریم.
  + **وضعیت پردازنده‌ها:**  تمام پردازنده‌ها را به حالت nop می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت InitTraffic می‌رویم.
* **InitTraffic**: در این حالت به تمام پردازنده‌ها تعداد ترافیک دریافتیشان را می‌دهیم و آن هارا آماده اجرای دستورات بعدی می‌کنیم.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت nop می‌بریم.
  + **وضعیت پردازنده‌ها:** تمام پردازنده‌ها را به حالت Init می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت FillTraffic می‌رویم.
* **FillTraffic**: در این حالت به تمام پردازنده‌ها ترافیک هایی را که از فایل خوانده بودیم می‌دهیم. در هر کلاک یک ترافیک و اگر ترافیکمان تمام شد آن پردازنده را بی‌کار می‌کنیم.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت nop می‌بریم.
  + **وضعیت پردازنده‌ها:** پردازنده هایی که هنوز تمام پکت هایشان را نگرفته اند، به حالت fill و سایر پردازنده ها به وضعیت nop در می‌آیند.
  + **وضعیت بعدی خود ماژول:** به حالت FillTraffic می‌رویم اگر پردازنده ای باشد که در این وضیعت fill بوده. در غیر اینصورت یعنی تمام پردازنده ها ترافیک مربوط به خودشان را گرفته‌اند و به وضعیت PreDequeTraffic می‌رویم.
* **PreDequeTraffic**: در این حالت تمام پردازنده‌ها صف مربوطه اشان را آماده می‌کنند و خروجیشان را برابر اولین ترافیکی که در قسمت قبل به آن‌ها داده شده بود می‌کنند تا در ادامه بتوانیم از خروجی آن‌ها استفاده کنیم.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت nop می‌بریم.
  + **وضعیت پردازنده‌ها:** تمام پردازنده‌ها را به حالت PreDeque می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت InitRouter می‌رویم.
* **InitRouter**: در این حالت تمام روتر‌ها را آماده سازی ‌می‌کنیم و اطلاعاتی که از ورودی خوانده‌ایم مانند تعداد پورت‌های ورودی و خروجی هر روتر را به آن می‌دهیم.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت Init می‌بریم.
  + **وضعیت پردازنده‌ها:**  تمام پردازنده‌ها را به حالت nop می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت LoadRtRouter می‌رویم.
* **LoadRtRouter**: در این حالت routing table ای که از ورودی خوانده‌ایم را به روتر‌ها می‌دهیم و هر مرحله یک عضو routing table را به روتر مربوطه‌اش می‌دهیم.
  + **وضعیت روتر‌ها:** تمام روتر هایی که routing table اشان را هنوز کامل دریافت نکرده اند را به وضعیت LoadRt می‌بریم و بقیه را به حالت nop.
  + **وضعیت پردازنده‌ها:** تمام پردازنده‌ها را به حالت nop می‌بریم.
  + و**ضعیت بعدی خود ماژول:** به حالت LoadRtRouter می‌رویم اگر روتری باشد که ورودی اش را کامل نخوانده، در غیر اینصورت به حالت LoadStagingRouter می‌رویم.
* **LoadStagingRouter**: در این حالت خروجی هر پردازنده را نگاه می‌کنیم و در صورتی که کانال مجازی متناظرش در روتر خالی باشد، این خروجی را در ورودی پورت ۰ روتر می‌گذاریم تا در نیم کلاک بعد، آن را بخواند (زیرا روتر ها با negedge clock کار می‌کنند). همچنین با توجه به اتصالاتی که از ورودی خوانده‌ایم، خروجی هر پورت هر روتر را در ورودی پورت روتر مقصد می‌گذاریم. از طرفی هر روتری برای اینکه اعلام کند vc اش خالی شده باید در جهت عکس به روتر دیگر پیام بفرستد بنابر‌این پیامی در جهت عکس اتصالات نیز از روتر مقصد به‌ مبدأ می‌فرستیم.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت LoadStaging می‌بریم.
  + **وضعیت پردازنده‌ها:** پردازنده هایی که خروجی آن‌ها وارد روتر می‌شود را به حالت Deque می‌بریم زیرا باید خروجیشان را تا clock بعدی دور بریزند و flit بعدی را بدهند. سایر پردازنده ها را به حالت nop می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت Phase0Router می‌رویم.
* **Phase0Router**: در این حالت تمام روتر‌ها فعالیت مرحله صفرشان را روی ورودی های گرفته شده انجام می‌دهند.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت Phase0 می‌بریم.
  + **وضعیت پردازنده‌ها:** تمام پردازنده‌ها را به حالت nop می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت Phas1Router می‌رویم.
* **Phase1Router**: در این حالت تمام روتر‌ها فعالیت مرحله یکشان را روی ورودی های گرفته شده انجام می‌دهند.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت Phase1 می‌بریم.
  + **وضعیت پردازنده‌ها:** تمام پردازنده‌ها را به حالت nop می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت CheckEnd می‌رویم.
* **CheckEnd**: در این حالت هر روتر را چک ‌می‌کنیم که آیا کارش تمام شده یا خیر و در صورتی که تمام روتر‌ها کارشان تمام شده بود خروجی is\_end را یک می‌کنیم و کار تمام است در غیر این‌صورت باید باز هم پکت ها را جا به جا کنیم و دوباره به حالت‌های قبلی رمی‌گردیم. همچنین در این وضعیت cycle هارا یک واحد زیاد می‌کنیم تا تعداد مراتبی که عملیات جابه‌جایی را انجام داده‌ایم بشماریم.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت nop می‌بریم.
  + و**ضعیت پردازنده‌ها:** تمام پردازنده‌ها را به حالت nop می‌بریم.
  + **وضعیت بعدی خود ماژول:** در صورتی که همه روتر ها کارشان تمام شده باشد به حالت EndState می‌رویم و در غیر اینصورت به حالت LoadStaging برمی‌گردیم.
* **EndState**: حالت پایانی است و به حالت بی‌کار در می‌آید.
  + **وضعیت روتر‌ها:** تمام روتر ها را به وضعیت nop می‌بریم.
  + **وضعیت پردازنده‌ها:** تمام پردازنده‌ها را به حالت nop می‌بریم.
  + **وضعیت بعدی خود ماژول:** به حالت EnState می‌رویم.