بسمه تعالی

درس طراحی سیستم‌های نهفته مبتنی بر FPGA

آزمایش ۴: طراحی یک شتاب‌دهنده‌ی سخت‌افزاری با رابط Avalon

پردیس دانشکده‌های فنی دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

دکتر بیژن علیزاده

دستیاران آموزشی:

نگار آقاپور n.aghapour.s@gmail.com

علی فلاحتی aflt1998@gmail.com

وصال بخت آزاد bakhtazad.v@gmail.com

نیمسال اول ۱۴۰۲ - ۱۴۰۱

مدت آزمایش: چهار جلسه

اهداف آزمایش:

- ✓ آشنایی با رابط Avalon و مفاهیم پیشرفته در Nios II
- ✓ آشنایی با طراحی یک شتاب‌دهنده‌ی سخت‌افزاری
- ✓ آشنایی با چگونگی اضافه کردن یک جز جدید به سیستم در ابزار Qsys

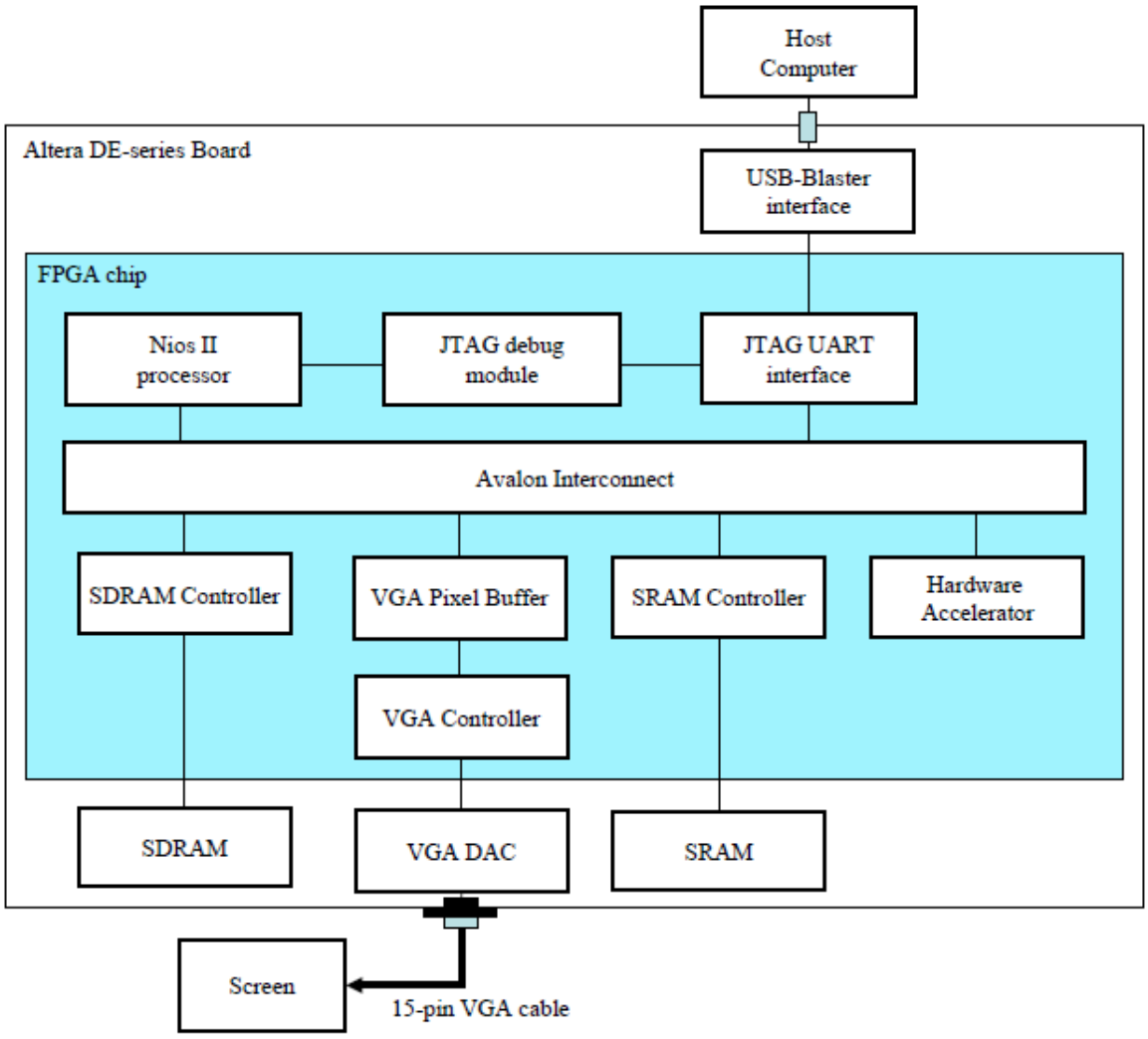
مقدمه

در این آزمایش با مفهوم شتاب‌دهی سخت‌افزاری[1] آشنا خواهید شد. شتاب‌دهنده‌های سخت‌افزاری مدارهایی هستند که به منظور برداشتن وظایف خاصی از دوش پردازنده طراحی شده‌اند. برای مثال انجام بی‌درنگ الگوریتم محاسبه FFT در اندازه‌های بزرگ توسط نرم‌افزار غیر ممکن است. با سپردن این کار به سخت‌افزار، اولاً سخت‌افزار می‌تواند همان کار را در زمان کمتری انجام دهد (مشابه روند مشاهده شده با دستورات اختصاصی) و ثانیاً نرم‌افزار همزمان به انجام سایر وظایف پردازشی خود می‌پردازد (بر خلاف روند دستورات اختصاصی).

[1] Hardware Acceleration

شرح آزمایش

شکل ۱ سیستم نمونه با شتاب‌دهنده سخت‌افزاری را نشان می‌دهد. تنها تفاوت این شکل با سیستم DE2 Media Computer در وجود واحد شتاب‌دهنده سخت‌افزاری است. این شتاب‌دهنده پس از دریافت دستور انجام عملیات مورد نظر، می‌تواند در صورت لزوم مستقیماً از طریق باس Avalon با سایر واحدهای سخت‌افزاری نیز در ارتباط باشد.



شکل ۱ – یک سیستم نهفته با شتاب‌دهنده‌ی سخت‌افزاری.

بخش‌های زیر را به ترتیب انجام دهید.

۱- طراحی سیستم مورد نظر به صورت نرم‌افزاری

در این بخش بر پایه سیستم DE2 Media Computer و نرم‌افزاری که در آزمایش دوم طراحی کرده‌اید، در Nios II Software Build Tools for Eclipse سیستمی پیاده کنید که:

- با کلیک روی دکمه‌ی Record، ۱۰ ثانیه از صدای ورودی ضبط شود. سپس متوسط اندازه دامنه سیگنال در N بازه با زمان ۱۰/N ثانیه، به صورت مستطیل‌های کنار هم رسم شود. برای رسم این نمودار تابع Plot_Audio را بنویسید و آن را پس از اتمام ضبط صدا فراخوانی کنید. کد شما باید با هر N دلخواهی کار کند.
- با کلیک روی دکمه‌ی Play، همراه با پخش صدای ضبط شده، مستطیل‌هایی در زیر مستطیل‌های میانگین نمایش داده شود که نشانگر مکان فعلی پخش صدا باشد (شکل ۲).

شکل ۲ - نمونه‌ای از رابط گرافیکی سیستم مورد نظر.

توجه نمایید که نمونه‌های ۲۴ بیتی صدا در بیت‌های MSB از آرایه‌های ۳۲ بیتی l_buf و r_buf ذخیره شده است. بنابراین برای جمع اندازه نمونه‌ها لازم است از متغیر با عرض بیت بالاتر (۶۴ بیتی در نرم‌افزار) استفاده کنید. زمان اجرای الگوریتم محاسبه دامنه متوسط را برای مقایسه‌های بعدی اندازه بگیرید و در گزارش ذکر کنید. توجه

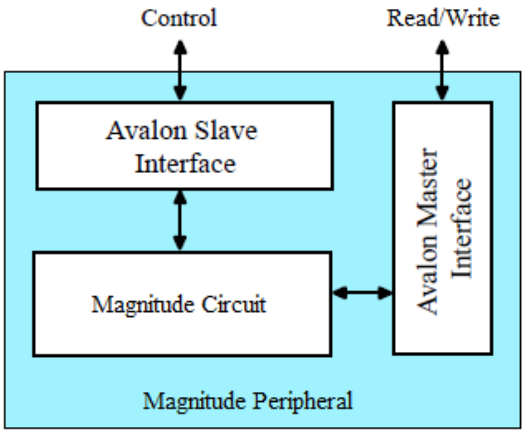
کنید که این کار حتما و حتما با استفاده از کتابخانه‌های مرتبط با زمان انجام شده و روند طی شده برای به دست آوردن زمان به طور کامل گزارش شود.

نکته: یکی از HALهای استفاده شده برای اندازه‌گیری زمان timestamp_timer می‌باشد. توجه داشته باشید که برای استفاده از این HAL لازم است ابتدا در تنظیمات پروژه BSP در بخش نرم‌افزاری، سخت‌افزاری که میخواهید با این HAL مرتبط شود (برای مثلا Interval Timer) را مشخص کنید. سپس می‌توانید در کد خود با استفاده از توابع کتابخانه alt_timestamp.h زمان را محاسبه کنید.

۲- طراحی شتاب‌دهنده سخت‌افزاری

در این بخش به طراحی سخت‌افزار شتاب‌دهنده و انجام عملیات بخش ۱ با استفاده از این سخت‌افزار خواهیم پرداخت. در شکل ۳ شمای کلی مدار جانبی سخت‌افزار محاسبه متوسط دامنه رسم شده است که می‌بایست به سیستم DE2 Media Computer اضافه شود. این شتاب‌دهنده شامل سه بخش است: مدار محاسبه دامنه، یک رابط Avalon Slave و یک رابط Avalon Master. پردازنده از طریق رابط Master خود، آدرس حافظه شروع بافر صدای راست (.Right Addr) و چپ (.Left Addr)، آدرس مکان ذخیره‌سازی جواب (.Out Addr)، تعداد بازه‌ها (Num) و تعداد نقاط در هر بازه (Size) را در اختیار رابط Slave مدار محاسبه دامنه قرار می‌دهد. با نوشتن مقدار یک در رجیستر Go مدار محاسبه دامنه، کار خود را شروع می‌کند. این مدار از طریق رابط Master خود، داده‌های داخل بافرهای صدا را از حافظه خوانده و پردازش می‌کند. پس از اتمام پردازش، نتیجه را در آدرسی از حافظه که توسط پردازنده مشخص شده است (.Out Addr)، ذخیره کرده و سیگنال Done را فعال می‌کند. پردازنده به صورت polling مقدار این سیگنال را بررسی کرده و به محض یک شدن آن، مستطیل‌های بیانگر دامنه سیگنال را رسم می‌کند. همانطور که مشاهده می‌شود، این روند مشابه عملیات DMA[۲] است با این تفاوت که جهت سهولت کار از وقفه استفاده نشده و به جای آن از polling سیگنال Done برای بررسی زمان اتمام کار شتاب دهنده استفاده شده است. برای درک بهتر ساختار در صورت نیاز به کدهای Verilog پیوست شده مراجعه کنید. توجه کنید که رعایت این ساختار **اجباری** است (رعایت نکردن این ساختار و یکپارچه‌کردن کد باعث کسر نمره خواهد شد.). لذا برای اطمینان از ساختار درست کد خود می‌توانید از کدهای پیوست شده استفاده کنید.

[2] Direct Memory Access



شکل ۳ - شمای کلی مدار جانبی محاسبه‌ی متوسط دامنه.

۲-۱- رابط Avalon Memory-Mapped و رفتار آن

به منظور طراحی و درستی‌سنجی رابط Slave و Master لازم است با پروتکل مورد استفاده در گذرگاه Avalon آشنایی داشته باشید. گذرگاه Avalon یک رابط استاندارد و راحت برای اتصال واحدهای سخت‌افزاری درون FPGA است که انواع مختلفی مانند Streaming و Memory Mapped را شامل می‌شود. در این آزمایش از رابط Avalon Memory-Mapped استفاده خواهیم کرد. انتقال داده بین دو ماژول Master و Slave با استفاده از این رابط با کمک سیگنال‌های address، waitrequest، read، write، readdata و writedata انجام می‌گیرید. از آن‌جایی که آغازگر عملیات خواندن و نوشتن، ماژول Master می‌باشد، از وضعیت ماژول Slave بی‌اطلاع بوده که ممکن است قادر به پاسخگویی نباشد. در این صورت سیگنال waitrequest که از سمت Slave صادر می شود، هماهنگ‌کننده این ارتباط است. نحوه استفاده از این سیگنال‌ها در آزمایش چهارم، در ادامه توضیح داده خواهند شد.

با توجه به توضیحات داده شده مراحل زیر را به ترتیب انجام دهید:

۲-۲ طراحی رابط Avalon Memory-Mapped Slave

همانطور که در شکل ۴ نشان داده شده است، در سیستم مبتنی بر Nios II، برای ارتباط میان پردازنده و سایر اجزا از گذرگاه Avalon استفاده می‌شود که در آن پردازنده‌ی Nios II نقش Master (پایه) را دارد. برای آنکه دستورات لازم توسط پردازنده به شتاب‌دهنده سخت‌افزاری ارسال شود، لازم است شتاب‌دهنده شامل یک رابط Slave (پیرو) باشد. در این بخش لازم است یک رابط Avalon Memory-Mapped Slave طراحی کنید که

قابلیت خواندن از رجیسترهای شکل ۴ و نوشتن در آن‌ها را داشته باشد. این رجیسترها را در بخش بعد استفاده خواهیم کرد. این رابط و فایل testbench مربوطه را به کمک Modelsim (یا هر محیط شبیه سازی سخت‌افزار دیگر) طراحی و درستی‌سنجی کنید.

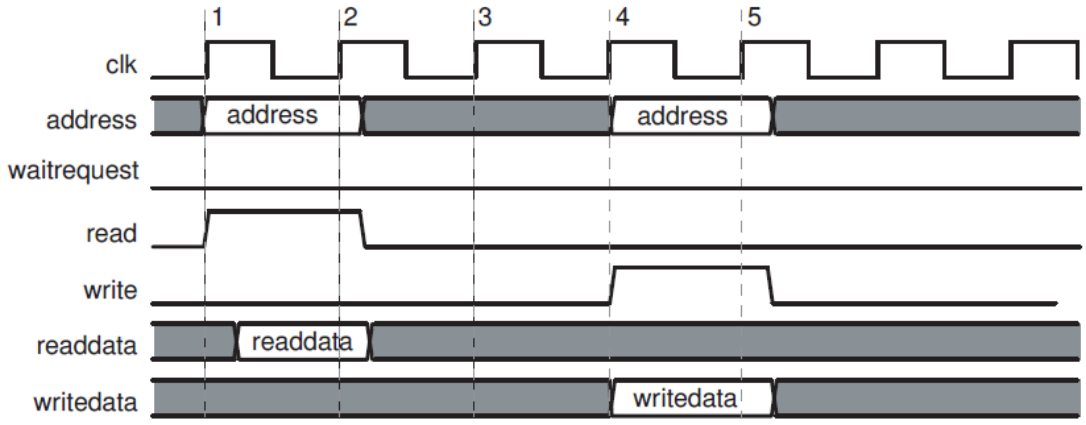
Slave Address	**31**	**30..12**	**11..1**	**0**	
00	Done	Size	Num	Go	**Config. Reg.**
01					**Right Addr.**
10					**Left Addr.**
11					**Out Addr.**

شکل ۴ - رجیسترهای مورد استفاده در رابط Slave.

نکته مهم: هنگام پیاده‌سازی رابط Slave جهت سهولت تشخیص واسط‌ها توسط Qsys، اسامی تمامی سیگنال‌های آن را با پیشوند _avs_avalonslave مانند avs_avalonslave_read انتخاب کنید (لیست کامل در اسلایدهای درس موجود است).

در این قسمت پروژه، رابط Slave شما نیازی به انتشار waitrequest ندارد، زیرا فرض بر این است که می‌تواند در یک کلاک یا تعداد کلاک ثابتی پاسخ دهد (و باید این کار را انجام دهد). شکل ۵ زمان‌بندی مورد نیاز در گذرگاه Avalon را (با عملکرد بدون waitrequest و با Wait Cycle=0) نشان می‌دهد. دقت کنید سیگنال chipselect که در طول انتقال خواندن/نوشتن فعال می‌شود، در این شکل نشان داده نشده است. همچنین بر خلاف این شکل، طرح شما می‌تواند Wait Cycle غیر صفر برای خواندن یا نوشتن داشته باشد. روند شکل ۵ بدین صورت است:

۱) Master سیگنال‌های read و address را روی لبه بالارونده کلاک منتشر می‌کند. در همان سیکل، Slave سیگنال‌های Master را decode می‌کند و readdata را ارائه می‌کند.
۲) Master، سیگنال readdata را روی لبه بالارونده کلاک می‌خواند و آدرس و سیگنال‌های کنترلی را غیر فعال می‌کند. در این لحظه انتقال پایان می‌یابد.
۳) هیچ سیگنال کنترلی منتشر نمی‌شود.
۴) Master، سیگنال‌های address، write و writedata را روی لبه بالارونده کلاک منتشر می‌کند. سیگنال‌های Master ثابت باقی مانده و Slave آن‌ها را decode می‌کند.
۵) Slave، سیگنال writedata را روی لبه بالارونده کلاک می‌گیرد. Master سیگنال‌های writedata، address و write را غیر فعال می‌کند و انتقال پایان می‌یابد.



شکل ۵ - زمانبندی Avalon برای انتقال خواندن و نوشتن بدون سیگنال waitrequest.

۱-۲-۲ درستی‌سنجی سیستم به کمک سوئیچ‌ها و LEDهای روی برد (اختیاری و در صورت نیاز)

قسمت زیر خواسته اصلی آزمایش نیست و صرفا جهت درستی‌سنجی قرار گرفته است. لذا در صورتی که در انجام آزمایش به مشکلی خوردید که نتوانستید آن را رفع کنید به سراغ این قسمت بروید.

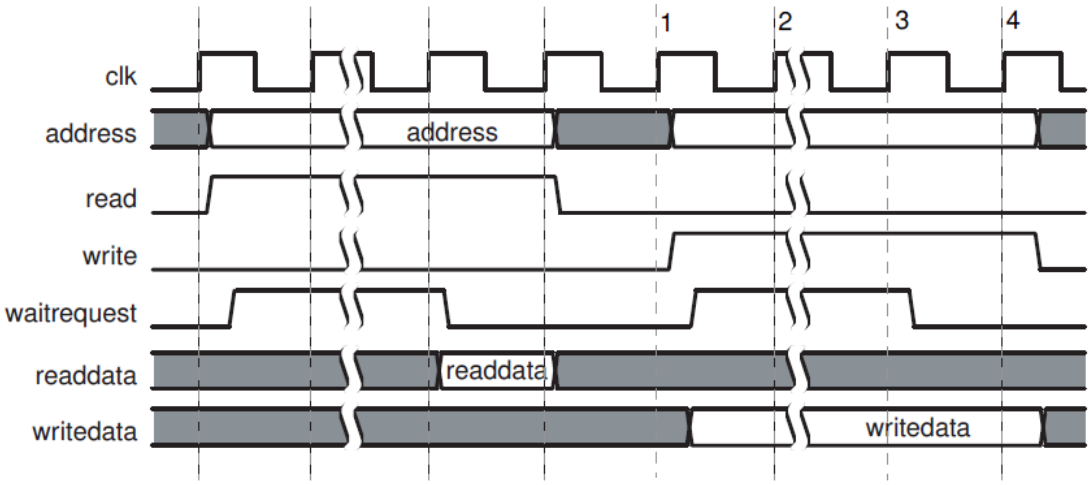
برای آنکه رابط Slave خود را بر روی FPGA تست کنید، خروجی LEDR[17:0] و ورودی SW[1:0] را به رابط Avalon Slave خود اضافه کنید به طوری که بیت‌های ۰ تا ۱۷ رجیسترها به LEDR وصل شوند (با استفاده از یک مالتی‌پلکسر که با سوئیچ‌های SW[1:0] کنترل می‌شود). این سخت‌افزار را در محیط Qsys با کلیک بر روی New Component... (از منوی فایل یا برگه Component Library) به سیستم DE2 Media Computer اضافه کنید (این روند کاملاً مشابه روند اضافه کردن Custom Instruction است با این تفاوت که اینترفیس Avalon Slave استفاده می‌شود و SW و LEDR به صورت Conduit تعریف می‌شوند). اینترفیس Avalon Slave سخت‌افزار خود را به باس data از Nios II متصل کنید. توجه کنید که در Qsys باید PIOهای مربوط به LEDهای قرمز و سوئیچ‌ها را غیر فعال کنید (تیک کنار آن‌ها را بردارید) و در ماژول خود پورت‌های مربوطه را export کنید. فایل DE2_Media_Computer.v را در پروژه Quartus متناظر با تغییرات ایجاد شده در خروجی Qsys (پورت‌های فایل nios_system.v تغییر یافته است) تغییر دهید. پروژه را کامپایل کرده و FPGA را پروگرام نمایید. نرم‌افزاری بنویسید که در رجیسترهای مذکور مقادیری را بنویسد و آن را خوانده و با مقدار نوشته شده مقایسه کند. همچنین مطمئن شوید مقدار ۱۸ بیت پایین رجیسترها به درستی توسط LED ها نشان داده می‌شود.

۳-۲ طراحی رابط Avalon Memory-Mapped Master

در این بخش رابط Master را برای ارتباط با کنترلر حافظه SDRAM پیاده‌سازی خواهیم کرد. مدار بخش قبل را به گونه‌ای تغییر دهید که با نوشتن در رجیستر Go (توسط پردازنده) به تعداد مشخص شده (Num) از آدرس راست (.Right Addr) و چپ (.Left Addr) را خوانده و جمع کند و نتیجه را در آدرس خروجی (.Out Addr) بنویسد. پس از این کار سیگنال Done فعال می‌شود. پس از ساخت و انتقال سیستم به FPGA، با نوشتن نرم‌افزار مناسب، صحت عملکرد آن را تست کنید.

نکته: در طراحی رابط Master در نظر داشته باشید که سیگنال waitrequest می‌تواند توسط کنترلر SDRAM منتشر شود. در واقع در ارتباط بین Master و Slave توسط Avalon Memory-Mapped، زمانی که Slave تا لبه کلاک قادر به پاسخ‌گویی نباشد، می‌تواند سیگنال waitrequest را قبل از لبه بالارونده کلاک منتشر کند. بدین ترتیب انتقال اطلاعات به تأخیر می‌افتد و Master، آدرس و سیگنال‌های کنترلی را تا غیرفعال شدن waitrequest، ثابت نگه می‌دارد. برای آشنایی بیشتر با رابط Avalon به اسلایدهای درس و در صورت نیاز به فصل ۳ از مرجع [۱] مراجعه کنید.

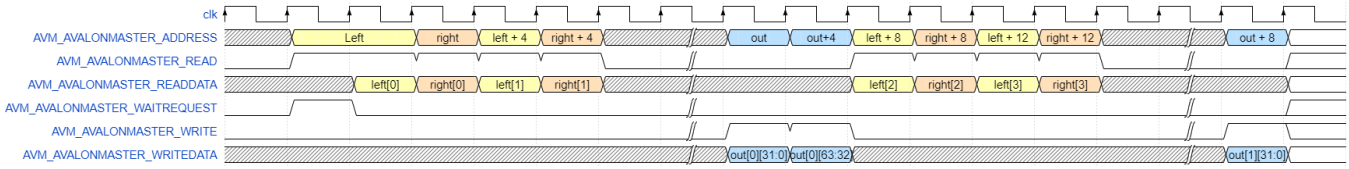
مشابه بخش قبل، اسامی سیگنال‌های مربوط به رابط Master را با پیشوند avm_avalonmaster_[signal] وارد نمایید. شکل ۶ زمانبندی Avalon را برای انتقال خواندن و نوشتن به همراه سیگنال waitrequest نشان می‌دهد.



شکل ۶ - زمانبندی Avalon برای انتقال خواندن و نوشتن به همراه سیگنال waitrequest.

در شکل ۷ یک نمونه زمان‌بندی سیگنال‌ها برای ارتباط Master شتاب‌دهنده با حافظه برای درک بهتر رفتار این ارتباط آمده است. توجه کنید که رفتار ماژول شما می‌تواند متفاوت با این مثال بوده و این فقط یک نمونه است.

در این شکل left و right، آدرس و داده بافرهای چپ و راست و out آدرس و داده خروجی می‌باشد. فرض شده است که تعداد داده‌ها در هر بازه (size) برابر ۲ می‌باشد. همچنین دقت شود که در این شکل فرض شده است که در کد شتاب‌دهنده و در تنظیمات component داخل ابزار Qsys، روش آدرس‌دهی برای رابط Master به جای Word به Symbol تغییر پیدا کرده است که در نتیجه، دسترسی به رجیسترهای ۳۲ بیتی، با گام‌های ۴تایی در آدرس امکان‌پذیر خواهد بود (واحد آدرس یک Symbol ۸ بیتی است.).



شکل ۷ - زمانبندی رابط Master شتاب دهنده برای ارتباط با حافظه.

۴-۲ طراحی مدار محاسبه دامنه

مطابق توضیحات بخش‌های قبل، مداری طراحی کنید که با دریافت آدرس بافر راست و چپ صدا، آدرس قرارگیری نتیجه عملیات، تعداد بازه‌های محاسبه‌ی دامنه و تعداد نقاط هر بازه، پس از نوشتن یک در رجیستر Go شروع به کار کرده، مجموع قدر مطلق داده‌ها را در هر بازه محاسبه کرده و در مکان مربوطه در آدرس نتایج بنویسد. در انتهای کار سیگنال Done، یک می‌شود. عملکرد این سیستم باید دقیقاً مشابه بخش ۱ (بخش طراحی نرم‌افزاری) باشد. با انتقال سیستم طراحی شده به FPGA، سرعت عملکرد سخت‌افزار خود را نسبت به نرم‌افزار بسنجید و زمان اجرا را در هر دو حالت گزارش کنید. توجه نمایید که عملکرد و اعداد بدست آمده توسط نرم‌افزار و شتاب‌دهنده **دقیقاً** باید یکسان باشند. همچنین برای ارتباط با سخت‌افزار نوشتن مستقیم در رجیسترها صحیح نبوده و منجر به کسر نمره خواهد شد.

ابتدا HAL مورد نیاز برای ارتباط با سخت‌افزار خود را بسازید و سپس با استفاده از توابع آن، کد اصلی را بنویسید. تابعی که در نهایت باید در کد اصلی استفاده شود در پیوست با نام amplitude_calculation.c قرار داده شده است. کتابخانه خواسته شده را در یک فایل با نام amplitude_circute.h به صورت جداگانه بنویسید و به فایل‌های پروژه اضافه کنید (این بخش خواسته‌ی آزمایش می‌باشد).

به صورت خلاصه در بخش ۲ قدم‌های زیر باید انجام شوند:

- طراحی رابط Avalon Memory-Mapped Slave و قراردادن کد توصیف سخت‌افزاری آن در فایل accelerator_slave.v

- طراحی رابط Avalon Memory-Mapped Master و قراردادن کد توصیف سخت‌افزاری آن در فایل accelerator_master.v
- طراحی مدار محاسبه دامنه و قراردادن کد توصیف سخت‌افزاری آن در فایل accelerator.v
- ساخت HAL مورد نیاز برای ارتباط با شتاب‌دهنده سخت‌افزاری طراحی شده و قراردادن کد توابع مربوط به آن در کتابخانه با عنوان amplitude_circute.h
- استفاده از توابع داخل HAL ساخته شده در فایل amplitude_calculation.c برای راه‌اندازی شتاب‌دهنده سخت‌افزاری طراحی شده
- سنتز و پیاده‌سازی سخت‌افزار توصیف شده در قالب یک شتاب‌دهنده برای پردازنده Nios II بر روی FPGA
- نوشتن کد نرم‌افزاری و انجام عملیات زیر:
  - صدازدن تابع ()amplitude_operation برای راه‌اندازی شتاب‌دهنده سخت‌افزاری و محاسبه بازه زمانی از قبل از شروع راه‌اندازی شتاب‌دهنده تا بعد از اتمام کار آن.
  - انجام عملیات محاسبه دامنه به صورت نرم‌افزاری (بخش ۱) و محاسبه بازه زمانی لازم برای انجام آن
  - مقایسه زمان محاسبات به صورت سخت‌افزاری با محاسبات به صورت نرم‌افزاری

بخش امتیازی (٪۲۰): رابط Master را به گونه‌ای طراحی کنید که امکان Burst داشته باشد. از Burstهای حداکثر ۱۶تایی برای انتقال داده استفاده کنید. فاصله بین گرفتن سیگنال Go تا صدور سیگنال Done را در هر دو حالت با و بدون Burst با اضافه کردن یک شمارنده اندازه گرفته و باهم مقایسه نمایید.

خواسته‌ها:

- کد تابع قسمت نرم‌افزاری به صورت پیوست جداگانه
- کد تابع قسمت سخت‌افزاری به صورت پیوست جداگانه
- زمان اجرای کد نرم‌افزاری به همراه روش به دست آوردن آن
- زمان اجرای کد سخت‌افزاری به همراه روش به دست آوردن آن
- کدهایی که به عنوان HAL برای ارتباط با سخت‌افزار نوشته شده‌اند به صورت پیوست جداگانه
- کدهای سخت‌افزاری نوشته شده

- تصاویری از نتایج شبیه‌سازی کدهای سخت‌افزاری (از طریق نرم‌افزارهای جداگانه مانند ModelSim)
- پاسخ مربوط به سوالات داخل متن در گزارش آورده شوند.

نکات مهم:

پیروی از قالب خاصی در گزارش مد نظر نیست، اما ترجیحاً از قالب ارائه شده برای تکالیف کامپیوتری استفاده نمایید.

مراجع

[1] Altera, “Avalon Interface Specification”, Chapter 3: Avalon Memory-Mapped Interfaces, May 2007.Available: http://www.altera.com/literature/manual/mnl_avalon_spec_1_3.pdf

موفق باشید