Abiel Bermudez A80909
Fabián Meléndez B34144
Javier Peralta B35191
Mario Zamora B07016
Grupo 4

Plan de Trabajo SD-Host

Este documento contiene las especificaciones iniciales y más detalladas del proyecto así como una tabla con la repartición de tareas.

I. Especificación inicial del proyecto

El SD-Host está conformado por 5 bloques y dos interfaces. Adicionalmente de manera opcional se puede soportar transferencia de datos por DMA y en ese caso habría un 5 bloque funcional con el DMA. Para este proyecto se diseñará un SD Host con soporte de DMA.

Las dos interfaces del SD Host son la del bus de SD y la del bus del sistema. Estas dos interfaces son asincrónicas; el bus del sistema funciona con el reloj del CPU y el bus de SD funciona con reloj que es más lento que del CPU llamado SDCLK. El driver de sd host al ser software que se ejecuta en el CPU corre al tiempo del reloj del bus de sistema y la tarjeta SD está sincronizada al SDCLK. Es trabajo del SD Host sincronizar todas las señales entre estas dos interfaces. Los bloques de datos deberán ser sincronizados en el buffer. Todos los registros de estado deberán estar sincronizados al bus de sistema(excepto si se crean registros especiales para realizar pruebas). A una diferencia de temporización entre los registros que puede utilizar el driver y la interfaz del SD, por lo que el driver no puede controlar directamente esta interfaz y debe de depender del SD-Host para controlar el bus dependiendo de los valores de los registros de configuración.

El conjunto de registros es la interfaz con el CPU y lo único que driver puede alterar directamente. En este bloque se encuentran registros de estado para todo el SD Host, para controlar interrupciones, para configurar el bus de SD, para controlar el DMA, con la información de que capacidades del estándar soporta el controlador, y un espacio para leer y escribir datos al buffer, (esto permite transferencias de datos por medio del driver controladas por software).

El bloque de DMA (Direct Memory access) permite la transferencia de datos directamente a la memoria principal sin interrumpir la ejecución normal del CPU. Hay dos algoritmos definidos para DMA, el SDMA(definido en la versión 1 del estándar del SD-Host) y ADMA (definido en la versión 2). El bloque puede implementar cualquiera de los dos algoritmos, la indicación de cual DMA es soportado debería estar indicada en los registros de capacidades. Es trabajo del driver asegurarse que

tanto el bus del sistema como el SD-Host soporten DMA antes de intentar realizar una transacción usando este protocolo.

La interfaz del bus del SD consta de 2 bloques, el SD CMD control que se encarga de conseguir información y enviar comandos a la tarjeta sd y el bloque de SD DAT control que se encarga de controlar la transferencia de datos entre el SD Host y la tarjeta.

II. Especificación detallada

II.a) Conjunto de registros

El conjunto de registros consta de 256 registros de 8 bits, las direcciones se dan en base a bytes. Los registros se deberían de poder leer secciones de 1 byte, media palabra o una palabra. Este es el único bloque al que el el driver puede manipular directamente, por lo que se necesita una interfaz de lectura y escritura de los registros hacia el CPU. Además varios de los otros bloques del SD Host deben poder y leer cambiar varios de los registros así que también se necesita una interfaz para esos bloques, pero es mucho más sencilla que la del CPU ya que no tiene que manejar el direcciones, coas que la del CPU sí requiere.

Se definen varios tipos de registros, no solo por su contenido sino también por las operaciones que le están permitidas al driver, Hay registro de solo lectura, lectura escritura, solo escritura, escritura durante la inicialización del hardware al ser encendido, lectura-escritura con limpieza automática, lectura que sólo deben ser alterados en el reset de encendido (y no por otras operaciones de reset, como un reset por software) y registros reservados. Estas protecciones de escritura no son fundamentales para la transmisión de tramas y multi-tramas por lo que en general no son prioridad para esta implementación.

Además solo se implementará el reset de encendido, el cual el estándar define que todos los registros deben inicializarse en 0, excepto los de capacidades y versión del SD-Host cuyos valores dependen del diseño del Host en sí.

Para esta implementación es importante implementar 3 partes: la interfaz con el CPU la interfaz con el resto del host y el bloque de los registros.

- **Bloque de registros**

Consiste en los 256 registros de 1 byte cada uno, y asegurarse que inicien en 0 ante un reset.

- **Interfaz del CPU**

Consiste en una entrada de datos, una salida de datos una señal de petición (req) para señalarle al bloque que el driver quiere leer o escribir y cuál es el tamaño que se desea leer o escribir, una señal para indicarle si se desea escribir o leer, una entrada con la dirección a la cual se le va a realizar la

operación y una señal para indicarle al CPU que la operación está completa. Además hay que implementar los decodificadores necesarios.

- **Interfaz SD Host**

A diferencia de la interfaz del CPU la interfaz al resto del sd host es simple. Para lectura habrá una salida que consta cables con de los valores de los 2048 bits de los registros, se puede conectar directamente los bits necesarios como entradas a los bloques que las necesiten.

Para escritura por sección que se necesite modificar se necesitan dos señales, una con el dato que se va a modificar y otra para habilitar la entrada de datos.

II.b) Buffer

El buffer debe ser capaz de almacenar , aceptar y proveer hasta 32 bits por ciclo de reloj, su almacenamiento es como una matriz donde la salida y entrada son 32 bits y el largo de tamaño variable sin embargo el bloque de DAT debe saber desde antes el largo de paquetes(cantidad de paquetes de 32 bits).

El buffer debe comunicar cuando está lleno y cuando esta vacio ademas de tener un puntero para poder acceder a cualquiera de los datos almacenados y llevar registro de todos los datos almacenados.

II.c) SD data

El bloque de DAT debe ser capaz de realizar transmisiones desde y hacia la tarjeta SD cuando se le indica que debe realizar el proceso, también debe soportar dos dominios de reloj debido a las diferencias de velocidad entre el host y la tarjeta SD. Se debe verificar la correcta transmisión de los paquetes mediante confirmaciones entre ambas partes además de controlar el bus de salida bidireccional aunque este último no se programe es una parte funcional en el diseño real.

El bloque de Control trabaja a frecuencia de reloj de la computadora , se encarga de las señales que provienen de registro y del DMA que solicitan nuevos procesos y especifican la naturaleza del proceso a realizar; controla cuando la capa física debe iniciar estos procesos y espera a que esta finalice debido a que esta ultima es mas lenta.

EL bloque capa física trabaja con dominio de reloj de la SD ,lleva control del tiempo de envio y recepcion hacia y desde la SD ,además este bloque lleva registro de los bits entrantes y salientes desde la línea DAT mediantes sub bloques que se conocen como serializadores y deserializador . Posee señales para indicarle al buffer que trabaja al mismo dominio de reloj que debe empujar o sacar datos desde la capa física en los modos de funcionamiento de lectura y escritura respectivamente.

Los bloques de serializacion y deserializacion deben soportar hasta 32 bits y retener estos valores para la transmisión hacia el fifo o hacia la SD, además deben poder comunicar que sus propios procesos está debidamente terminados para el correcto funcionamiento de la capa física.

II.d) SD command

II.e) DMA

Etapa/Tarea	Dueño	Tiempo Dedicado	Tiempo Estimado	Porcentaje Avanzado	Fecha de Entrega	Comentarios
Detalle inicial de las especificaciones	Todos	Mario 1 hora Javier 2hrs	4 hrs	100%	15-10-16	Todos leen para decidir quien va a hacer cada bloque, Javier escribe descripción en el documento. Terminado
Detalle final de las especificaciones	Todos	Mario 1 horas Javier 1hrs	4 hrs			Cada uno realiza la de su bloque y coordina con los demás las señales que vienen/van de otros bloques
Programación del bloque de registros	Javier	4hrs	4 hrs	90%	16-11-16	Este bloque era el más sencillo, no requiere una máquina de estados, los otros
Pruebas del bloque de registros	Javier	3hrs	5 hrs	100%	16-11-16	Toda la funcionalidad implementada funciona como es esperado
Programación del bloque de SD Data	Mario	15 hrs	6 hrs	100%	16-11-16	Se reescribió el código un par de veces cuando encontraba errores de lógica
Pruebas del bloque de SD data	Mario	10hrs	6 hrs	80%	16-11-16	Esto incluye seguimiento en errores de lógica no previstos y seguimientos de errores debido a problema con la sintetización.Las pruebas no son exhaustivas
Programación del bloque de SD command	Fabián	7hrs	6 hrs		16-11-16	
Pruebas del bloque de SD data	Fabián	5hrs	6 hrs		16-11-16	
Programación del bloque de DMA	Abiel	0hrs	6 hrs	0%	16-11-16	

Pruebas del bloque de DMA	Abiel	0hrs	6 hrs	0%	16-11-16	
Encontrar y probar un Buffer	Mario	3hrs	6 hrs	70%	16-11-16	El buffer no esta probado del todo por lo que se pego por cuesiones de tiempo
Integración de los bloques	Todos	30 min	20 hrs		3-12-16	Falta un DMA para integrar todo
Testbench del SD-Host: Señales del CPU			4 hrs		3-12-16	
Testbench del SD-Host: Señales de la tarjeta SD			4hrs		3-12-16	
Pruebas de sistema	Todos		10 hrs		3-12-16	
Creación y actualización del plan de trabajo	Javier	1 hrs 40 min Mario 30min	2 hrs	50%	9-12-16	Falta que los demás pasen sus descripciones y horas usadas
Creación y actualización de la bitácora	Javier	1 hrs 18 min	2 hrs	100%	9-12-16	
Reporte escrito	Todos	Mario 2hrs Javier 1hrs	5 hrs		9-12-16	
Presentación	Todos		5 hrs		9-12-16	
Otros	Todos	1 hrs	-	-	-	Incluye limpieza del repositorio, automatización de pruebas y otras tareas que no son código pero es importante realizarlas para el proyecto