SD-Host **Grupo 4**

IE-523 Digitales II
Fabián Meléndez, Javier Peralta, Mario Zamora
Profesor: Enrique Coen Alfaro
Fecha: 07 de Octubre 2016

Resumen

Se implmentaron varios bloques de un SD-Host

Descripción Arquitectónica

El sd-hst consiste en 5 bloques: cmd control, dat control, dma, el register set, y un buffer. Las entradas y salidas consisten en, hacia el lado del bus del sistema, la interfaz de los registros con el CPU y las señales de control y transferencia de datos del DMA. Hacia el lado del bus de SD está una salida del CMD control a la sd para darle comandos a la tarjeta y 3 salidas de DAT control para la transferencia de datos entre el SD Host y la tarjeta. Los detalles de cada señal se detallan en la descripción de los bloques correspondientes.

Registros

El estándar define 256 registros de 1 byte, esta compuesto de el bloque de registros, una interfaz para el CPU, otra para los demás bloques. Tiene las siguientes señales de entrada:

- **clk:** la señal de reloj, los registros están sincronizados al reloj del bus del sistema.
- **reset** La señal pone a todos los registros en 0.

Interfaz del CPU:

- **address;** Dirección del registro al cual se va a leer o escribir.
- **wnr:** Si es 0, es una lectura, si es 1 es una escritura.
- **req:** si es 0 no hay ninguna solitcitud, si es 1 se desea hacer una operación (lectura/escritura) usando 1 byte, si es 2 la operación es de 2 bytes y si es 3 la operación es de una palabra.
- **data_in:** Entrada de datos para escribir es de 1 una palabra de ancho, si se hacen operaciones de menos los bits más significativos son ignorados.
- **data_out:** Salida de datos, es de 1 palabra de ancho, si se hacen operaciones de menos los bits más significativos son 0.
- **ack:** Esta señal indica que la operación solicitada ya fue finalizó.

Interfaz del SD-Host:

- Para cada diferente grupo de bits que haya que escribir hay una linea de datos y un enable. Cuando el enable esta en alto los datos son escritos al registro correspondiente, esta escritura es fija, es decir no hay dirección y las lineas siempre escriben al mismo registro, hay una de estas interfaces por cada registro que se escribiría (Nota: Esto no se implemento, solo esta aquí para documentar como se tenía planeado implenetar).
- Para lectura a una serie de cables que salen a los otros bloques, hay acceso a cada bit de cada registro. el modulo remap.v es un wrapper que les asigna nombres de acuerdo a lo que contienen para facilitar su uso a los demás bloques.

DAT

Descripción

La necesidad del bloque DAT surge debido a que debe haber alguna parte encargada del manejo de los datos , la transición desde el buffer hacia la SD y de la SD hacia el buffer; además debe llevar control de las interrupciones debido transmisiones fallidas, el tamaño de los bloques y señales de confirmación para el correcto funcionamiento del bloque.

El bloque de Dat esta compuesto de dos módulos principales, el modulo de control y el modulo de capa física. Ambos módulos poseen sus propios sub bloques con funcionalidades especificas y definidas para los modos de operación en escritura y lectura de la SD.

La característica principal del diseño dividido primordialmente en dos secciones se debe a que el bloque debe trabajar con dos dominios de reloj diferentes uno mas lento debido a las limitaciones de la tarjeta SD y el otro mas rápido el dominio de reloj del host.

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 fisica trabaja con dominio de reloj de la SD ,lleva control del tiempo de envió y recepción hacia y desde la SD ,además este bloque lleva registro de los bits entrantes y salientes desde la linea DAT mediante 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 fisica en los modos de funcionamiento de lectura y escritura respectivamente.

El serializador y deserializador trabajan a una magnitud de bits de 32 que es el numero que generalmente maneja bloque DAT , pero estos dos bloques soportan trabajar con menos bits y son configurables para distintas necesidades.

Control

EL bloque control inicia en reiniciado y procede al estado RESET , este estado le avisa a la capafisica que debe reiniciarse y espera a que esta termine de otro modo ocasiona errores.

Procede al estado IDLE el cual es el estado por defecto de la maquina de estados con salidas apagadas y esperando la señal de nuevo servicio proveniente del bloque DMA.

El estado checkfifo revisa el estado del fifo , si este esta listo procede al estado solicitud hacia la capafisica.En el estado solicitud avisa a la capa fisica que debe comenzar un proceso ademas pone en la salida la cantidad de bloques a trasmitir.

En el estado waitResponse espera que la capa fisica termine la transferencia y este procede hacia el estado IDLE de la capa fisica .En el estado IDLE espera que la capafisica este libre y con fifo listo para una nueva transferencia, dependiendo de la opción de múltiples bloques se devuelve a checkfifo para trasmitir el siguiente bloque o procede al estado IDLE.

Capa Fisica

Inicia en el estado RESET y avanza autoaticamente al estado IDLE que tiene salidas en cero y no estan activos el serializador , deserializador ni la señal de pad Enable.

Figura 1: Maquina de estados del bloque control.

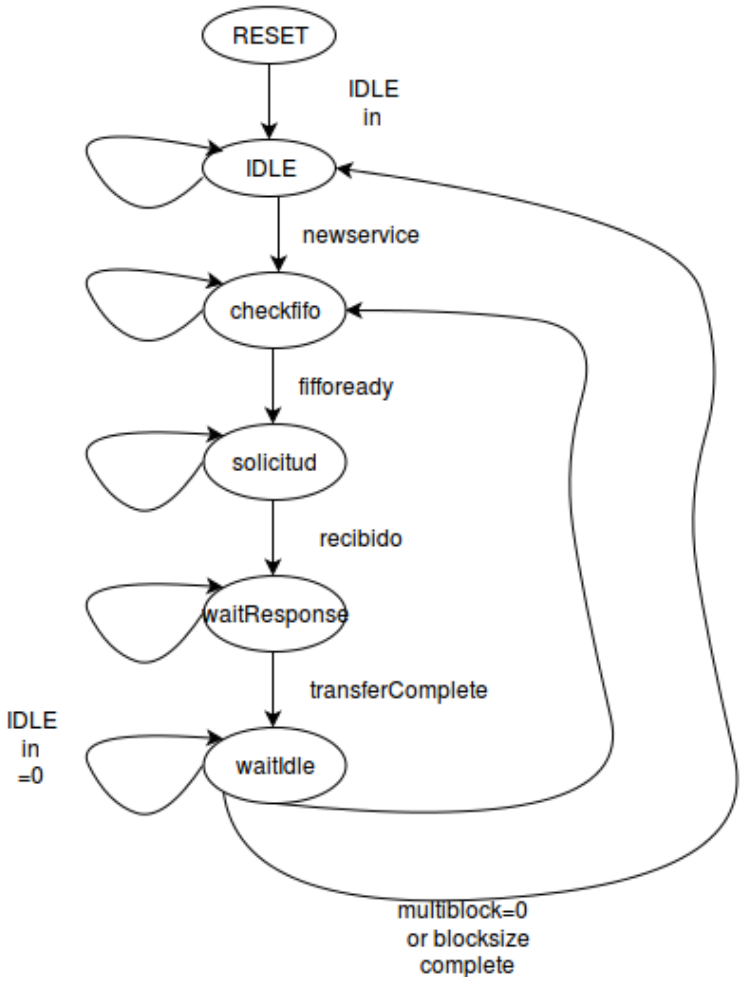
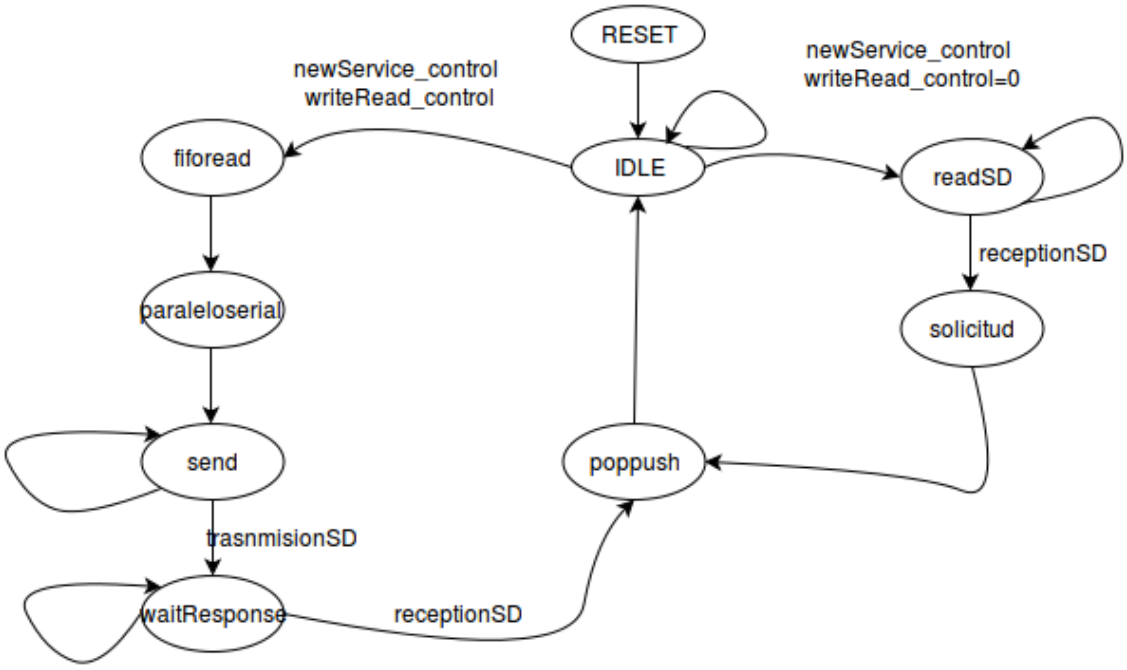


Figura 2: Maquina de estados del bloque de capa fisica



Dependiendo de la naturaleza de la trasnmision del estado IDLE avanza hacia fiforead o hacia read SD.En fiforead reinicia el deserializador y pone en la entrada del serilizador el valor del fifo ademas comunica que recibió correctamente la señal de nuevo proceso a control.

En el estado paraleloserial habilita el serializador y el pad , procede hacia el estado send para trasnmitir datos hasta que trasmita todos los bits y procede al estado waitResponde donde habilita el deserializador para esperar la respuesta de la SD y deshabilita el serializador.Procede hacia el estado poppush el cual es un estado para asegurar que los datos del fifo estan listos cuando se procede hacia IDLE.

El estado readSD habilita los deserializadores y espera hasta que el deserializador comunique que tiene todos los datos para proceder al estado serialParalelo en el cual se pone a la entrada del fifo los datos serializados de forma paralela y se procede al estado poppush donde se empujan los datos al fifo.

CMD

Este módulo se ocupa de emitir instrucciones hacia el dispositivo SD, así como de entregar las respuestas al host driver en caso de haberlas. La comunicación entre el driver y este módulo se hace por medio del bloque de registros. Se puede dividir el el diseño en dos partes.

Control

Escribe y lee del bloque de registros, con el fin de tomar decisiones a la hora de emitir comandos y respuestas. Controla el bloque de comunicación y permite sincronizar las operaciones en ambos dominios del reloj (SD Host y Driver Host).

- **Host:** Maneja la verificación de comandos y respuestas. Emite errores (interrupciones) en caso fallar alguna de las verificaciones (index, crc, timeout, command origin, last bit). Comunica al siguiente

módulo cuándo debe iniciar la emisión de un comando, y luego de verificar la respuesta la transfiere a los registros

- **SD:** Se mantiene en IDLE hasta que se le comunica que debe emitir un comando. Seguidamente emite una señal para notificar al módulo de comunicación que debe empezar a transmitir los datos a la SD, definiendo el tamaño esperado de la respuesta según el registro response type. Al completarse dicho proceso se mantiene en modo de espera (por un lapso máximo de 63 ciclos de reloj), hasta que la tarjeta SD responda y se empiece a recibir la respuesta, del mismo modo, a través del módulo de comunicación. Una vez se completa el proceso de transmisión se notifica al control del host para que valide y transfiera la respuesta a los registros.

Plan de Pruebas

Registros

- **Descripción**: Se resetean los registros, luego se carga una entrada aleatoria y se escribe, luego se lee del registro se compara lo que se escribió con lo leído y se levanta un error sino son iguales. También se leveanta un error si al modulo sintetizado no tiene exactamente las mismas salidas que el modulo conductual. Se itera sobre todos los registros con etradas aleatorias.
- Estado: Pasó

DAT

- **Descripción** :La prueba consiste en dos partes de las cuales se debe modificar una variable en el probador 40 writeread esto debido a que las pruebas son muy largas.
- **Descripción Prueba 1**.Se resetea el bloque DAT y se prueba en modo escritura hacia la SD , se prueban con una transmisión de dos bloques y se verifica que pase por todos los estados activando salidas de relevancia y observando que su respuesta concuerde con el dato del fifo utilizado en el probador, el probador esta configurado para cambiar el valor después de la primera transicion para simular el fifo.
- **Descripción Prueba 2**Se resetea el bloque DAT y se prueba en modo de lectura de la SD, para simular los efectos de la SD serializando sus datos en el probador se cambia el valor de la entrada que simula la SD antes de cada ciclo de reloj , se verifica que proceda por todas los estados correctamente activando las salidas correspondientes y que la salida sea la esperada del probador.

Instrucciones de utilización de la simulación

Se provee un Makefile con este proyecto, para correr las simulaciones solo hace falta correr el comando make en el directorio raíz de proyecto, el programa make se encarga de compilar cada uno de los testbench y correr las simulaciones. En el directorio de tests se guardan los .vcd que se pueden visualizar con gtkwave, además se genera un archivo de texto sim.log que contiene el nombre del testbench, el dumpfile, y cualquier error detectado.

Para volver a correr las pruebas se puede correr el comando “make clean” y después make nuevamente, esto limpiará los resultados viejos y los volverá a generar.

Figura 3: Prueba de escritura de la SD

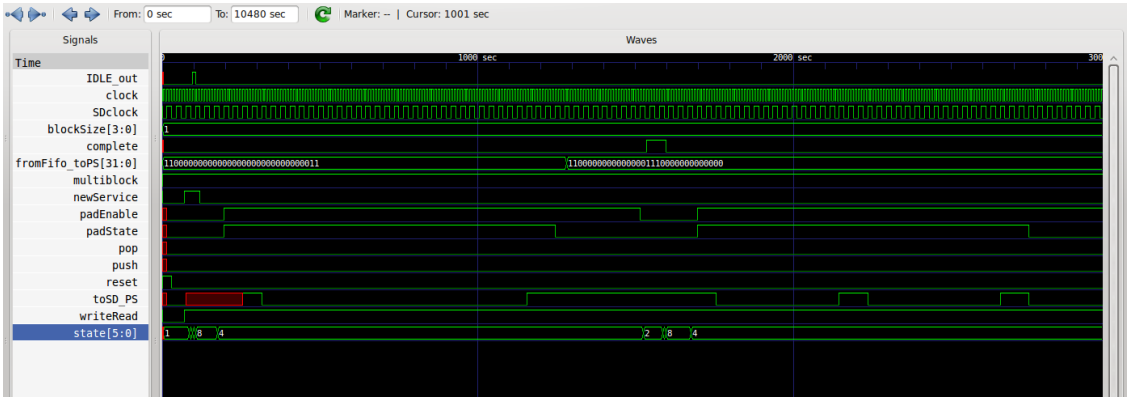
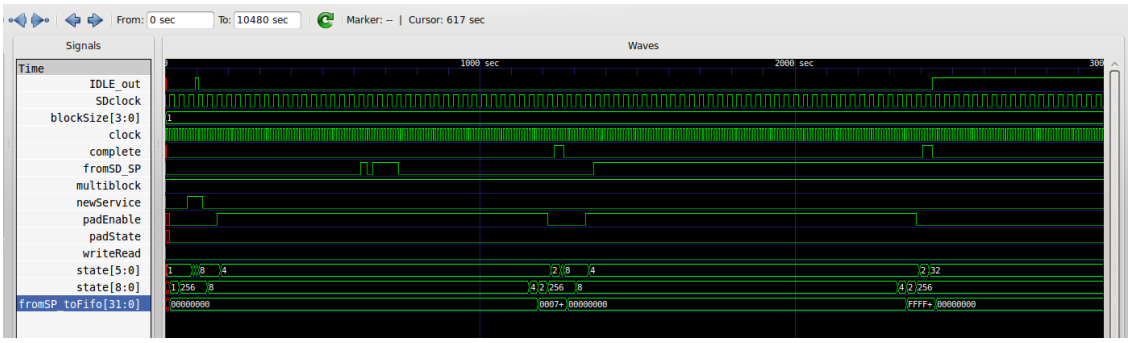


Figura 4: Prueba de lectura de la SD



Resultados

En las figuras 3 y 4 se muestra la simulación de las pruebas antes descritas, las pruebas son exitosas debido a que se dio un seguimiento de los errores en las pruebas y se fueron corrigiendo a medida que se identificaba el error.

Conclusiones y recomendaciones

Se logró implmentar un bloque de registros de acuerdo al estandar sd, pero este no tiene las protecciones de escritura que dice el estandar y no implementa la escritura desde el SD Host.