Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Carrera de Maestría en Electrónica

Verificación Funcional

MP6134

Prof. Gerardo Castro Jiménez

Proyecto 3:

Plan de Verificación para dos funciones del controlador de la SDRAM

Est. Johan Solis Arbustini

200933247

II Cuatrimestre, 2016

Contenido

Lista de tablas

Lista de figuras

I. Funciones a verificar

La SDRAM es una memoria dinámica de acceso aleatorio (DRAM) de alta velocidad con una interfaz síncrona. La interfaz síncrona y la arquitectura interior plenamente pipeline de la SDRAM permite velocidades de datos muy rápido si se usa de manera eficiente.

La SDRAM está organizada en bancos de memoria direccionada por fila y columna. El número de bits de fila y dirección de columna depende del tamaño y la configuración de la memoria.

Cuando se emite la lectura o escritura de comandos, la dirección inicial de la columna se presenta a los dispositivos SDRAM. Los datos iniciales se presentan simultáneamente con el comando de escritura. Para el comando de lectura, los datos iniciales aparecen en el bus de datos de 1-4 ciclos de reloj más tarde.

El controlador de la SRAM corresponde al dispositivo a verificar (DUV). Este se divide en cuatro sub-bloques:

- *SDRAM Bus convertor*: Este bloque convierte y vuelve a alinear el sistema de 32 bits en formatos de dispositivos estándar de SDRAM discretos equivalentes de 8/16/32 bits.

- *SDRAM Request Generator*: Si la señal de wrap es cero y existe un cruce de página este separa la solicitud en dos solicitudes sino se envía directamente al bloque del controlador de banco. En caso de que la señal de wrap sea uno, no se modificara la solicitud. Además basado en el ancho del bus de la SDRAM, se tiene las siguientes modificaciones:

Table 1. Internal address and burst length

Sdr_width = 2'b00 32 Bit SDRAM	Internal req address	App_req_addr
	Internal Req Length	App_req_Len
Sdr_width = 2'b01 16 Bit SDRAM	Internal req address	{App_req_addr,1'b0}
	Internal Req Length	{App_req_Len,1'b0
Sdr_width = 2'b10 8 Bit SDRAM	Internal req address	{App_req_addr,2'b0}
	Internal Req Length	{App_req_Len,2'b0

- *SDRAM Bank Controller*: Este módulo toma las peticiones del generador de solicitudes de la SDRAM, chequea el éxito/fallas de visitas a la página, cuestiones de precargar/activar comandos y luego pasa la petición al controlador de transferencias de SDRAM.

- *SDRAM Transfer Controller*: Este módulo toma solicitudes del controlador de banco de la SDRAM, ejecuta la transferencia y los controles del flujo de datos a/desde la aplicación.

Esta descripción general indica que el controlador posee distintas características para lograr el funcionamiento correcto sobre la SDRAM. Por lo tanto, las funciones a verificar del controlador de la SDRAM son:

- Programmable column address
- Support SDRAM with four bank

Dirección de columna programable

Para reducir la cantidad de pines, las direcciones de filas y columnas de la SDRAM son multiplexados en los mismos pines. La decodificación de las direcciones de columna, banco y fila están basadas en la configuración de la siguiente manera:

Table 2 Address decoding base on configuration

cfg_col_bits	
2'b00	Column Address[11:0] = {4'h0,SDRAM Mapping Address[7:0]}; Bank Address[1:0] = {SDRAM Mapping Address[9:8]}; Row Address[11:0] = {SDRAM Mapping Address[21:10};
2'b01	Column Address[11:0] = {3'h0,SDRAM Mapping Address[8:0]}; Bank Address[1:0] = {SDRAM Mapping Address[10:9]}; Row Address[11:0] = {SDRAM Mapping Address[22:11};
2'b10	Column Address[11:0] = {2'h0,SDRAM Mapping Address[9:0]}; Bank Address[1:0] = {SDRAM Mapping Address[11:10]}; Row Address[11:0] = {SDRAM Mapping Address[23:12};
2'b11	Column Address[11:0] = {1'h0,SDRAM Mapping Address[10:0]}; Bank Address[1:0] = {SDRAM Mapping Address[12:11]}; Row Address[11:0] = {SDRAM Mapping Address[24:13};

Basado en estos bits de configuración de columna, las direcciones de columna, fila y banco serán generados por el bloque del controlador encargado de generar las solitudes de la SDRAM.

Soporta SDRAM con cuatro bancos

Los dispositivos SDRAM suelen dividirse en cuatro bancos. Estos bancos deben abrirse antes de un rango de direcciones para que se puedan escribir o leer. La apertura y cierre de los bancos tienen costos en el ancho de banda de memoria, por lo que el Core del controlador de la SDRAM ha sido diseñado para controlar y gestionar el estado de los cuatro bancos de forma simultánea. Esto permite al controlador abrir y cerrar de forma inteligente los bancos sólo cuando sea necesario.

II. Niveles de verificación

Un sencillo sistema en un chip (SoC) puede necesitar verificación a nivel de unidad en ciertos bloques nuevos, seguido de un esfuerzo de chip y a nivel de sistema. Por otro lado, un servidor complejo necesitaría la verificación a nivel de diseño, de unidad, de chips, sistemas, y co-verificación.

Los niveles inferiores de la jerarquía de verificación proporcionan un mayor control en el entorno más pequeño. Los niveles más altos proporcionan un sistema de visión amplia, pero pierden el control estricto.

A nivel de unidad se contienen varias piezas del nivel de diseño de HDL que forman en unidades. Interfaces y funciones son más estables que a nivel de diseño, ya que las unidades tienden a tener especificaciones formalizadas y contratos físicos o de temporización a la que se adhieren los diseñadores. Debido a que las interfaces y las especificaciones son más estables, se puede crear un entorno más avanzado (utilizando estímulos aleatorios y la comprobación autónoma). Una vez que el nivel de la unidad se verifica, se puede proceder a un nivel más alto de verificación, a sabiendas de que la funcionalidad básica de la unidad es correcta. Los niveles más altos comprueban que los protocolos de conectividad y de interfaz hacia y desde la unidad son correctos.

Hay factores técnicos que ayudan a decidir qué niveles escoger para un diseño dado. Estos factores se destacan a continuación:

- Elija siempre el nivel más bajo que contiene por completo la función objetivo.
- Cada pieza verificable debería tener su propio documento de especificación.
- El nivel apropiado de control y observabilidad impulsa las decisiones sobre la cual los niveles de verificar.
- Función a verificar puede dictar los niveles de verificación.

Para el controlador de la SDRAM se tiene el siguiente nivel de verificación, en el cual no se tiene el sistema completo de un SoC, sino únicamente se quiere verificar el Core del controlador de la memoria. Por lo tanto, se tienen 4 unidades que corresponden al modelo de la SDRAM, la interface de la SDRAM con el controlador, la interface Wish Bone y el controlador de memoria.

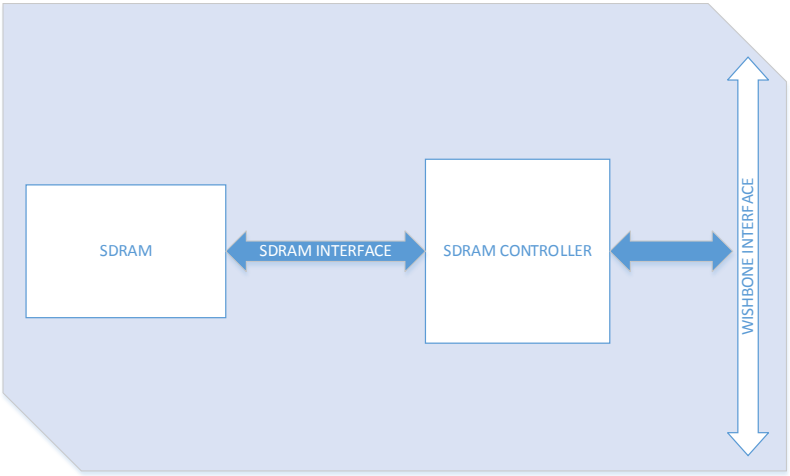

Figure 1 Nivel de verificación seleccionado

Se agrega la interface Wish Bone, debido a que el manipulador del bus Wish Bone se encarga del protocolo de enlace entre el Wish Bone master y controlador de SDRAM personalizado. Además, se encarga de los dominios de reloj necesarios. Este bloque incluye:

- Command Async FIFO.
- Write Data Async FIFO
- Read Data Async FIFO.

A nivel de funcionalidad se puede apreciar el nivel de verificación en el siguiente diagrama.

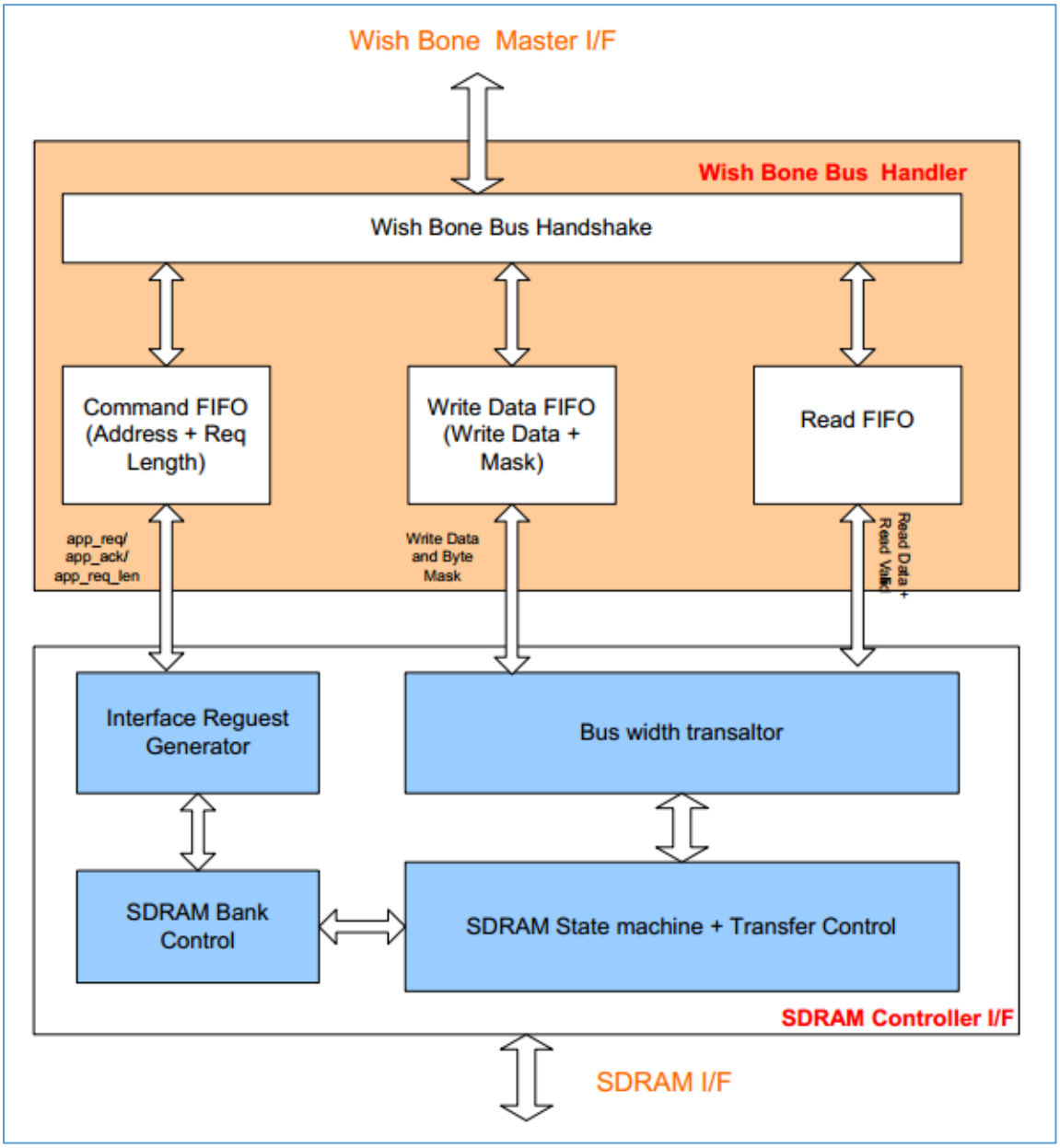

Figure 2 Diagrama de funcionalidad para el nivel de verificación.

III. Estrategia de verificación y ambiente

La discusión de controlabilidad y observabilidad divide el trabajo en dos tareas independientes: la conducción (controlabilidad) y de control (observabilidad) el diseño que se está probando.

Estas tareas son separadas pero deben trabajar juntos para tener éxito. Se capta un error si las entradas al diseño agravan la condición de falla y si las banderas entran en un estado ilegal. Uno de ellos es insuficiente sin el otro. Manejar todas las combinaciones posibles no puede cubrir un error si los chequeadores no logran identificar una mala condición. Del mismo modo, la comprobación de todos los fallos posibles es inútil si no se estimulan las condiciones que causan la falla. No se puede encontrar un error sin crear las condiciones que fallan y la detección incorrecta en el comportamiento del HDL. Por lo tanto, el manejo de estímulos y la comprobación son el yin-yang de la verificación.

Los detalles del ambiente de verificación describen si se tratará al DUV como “black box”, “white box” o “gray box”; proporciona detalles sobre la estrategia de verificación, incluyendo la cantidad de aleatoriedad o determinismo en un ambiente de simulación.

Ambiente:

La elección de un ambiente determinista o aleatorio impulsa componentes de ambiente divergentes. Los diseños simples con funcionalidad directa se prestan a enfoques deterministas de prueba. Funciones complejas requieren la asignación aleatoria, debido a que no puede prever todas las permutaciones de entrada.

El enfoque determinista requiere poner características en el ambiente para permitir la escritura todas las permutaciones de las pruebas deterministas. El entorno determinista debe ser lo suficientemente robusto para ejercer la función del DUV. En este entorno, el estímulo, el chequeo en las salidas y la intención del caso de prueba permanece en la mente del ingeniero de verificación. El entorno permite el flujo no inhibido de que la intención en el DUV.

El ambiente de aleatoriedad requiere que todas las permutaciones posibles ocurran en las interfaces de entrada del DUV. Aunque las pruebas determinista conduce un solo evento legal, los modos de verificación al requieren deshabilitar explícitamente estímulos ilegales y permitir todo lo demás. En lugar de considerar todos los escenarios posibles, la verificación aleatoria se aproxima a prevenir escenarios que prohíbe la especificación. El ambiente de aleatoriedad requiere codificar conocimientos en componentes de “checkers”, “scoreboard” y “monitor”.

La decisión sobre un ambiente al azar o determinístico afecta a las funciones dentro del modelo. El exceso de aleatoriedad puede prevenir los problemas que están siendo ocultos, como puede también conducir a falsos fallos. Una ventaja es que se puede controlar y limitar adecuadamente la aleatoriedad con controles "unrandomizing" incorporados.

Por lo tanto el ambiente implementado posee un ambiente que permite controlar la reducción en la asignación aleatoria para un enfoque más dirigido. Además, permite generar una secuencia determinista en el ambiente aleatorio para pruebas especializadas que permite moverse por un fallo de la arquitectura o para crear un escenario conocido.

Estrategia Verificación:

Si la estrategia de verificación elige un enfoque de la caja negro, puede no ser necesario tener los monitores probando el diseño; sin embargo, si el enfoque es de caja blanca, puede requerir muchos monitores para el DUV.

Se le elige un ambiente de caja gris para crear puntos de observación dentro del DUV, para asegurar que un conjunto de aspectos interesantes se producen durante la simulación. Al seleccionar este enfoque, puede ser factible para resolver algunos puntos de observación estándar que no va a cambiar, lo que provoca un menor mantenimiento debido a la estabilidad dentro de la aplicación.

Implementación

Las capas del testbench a implementar se puede aprecia en la siguiente figura.

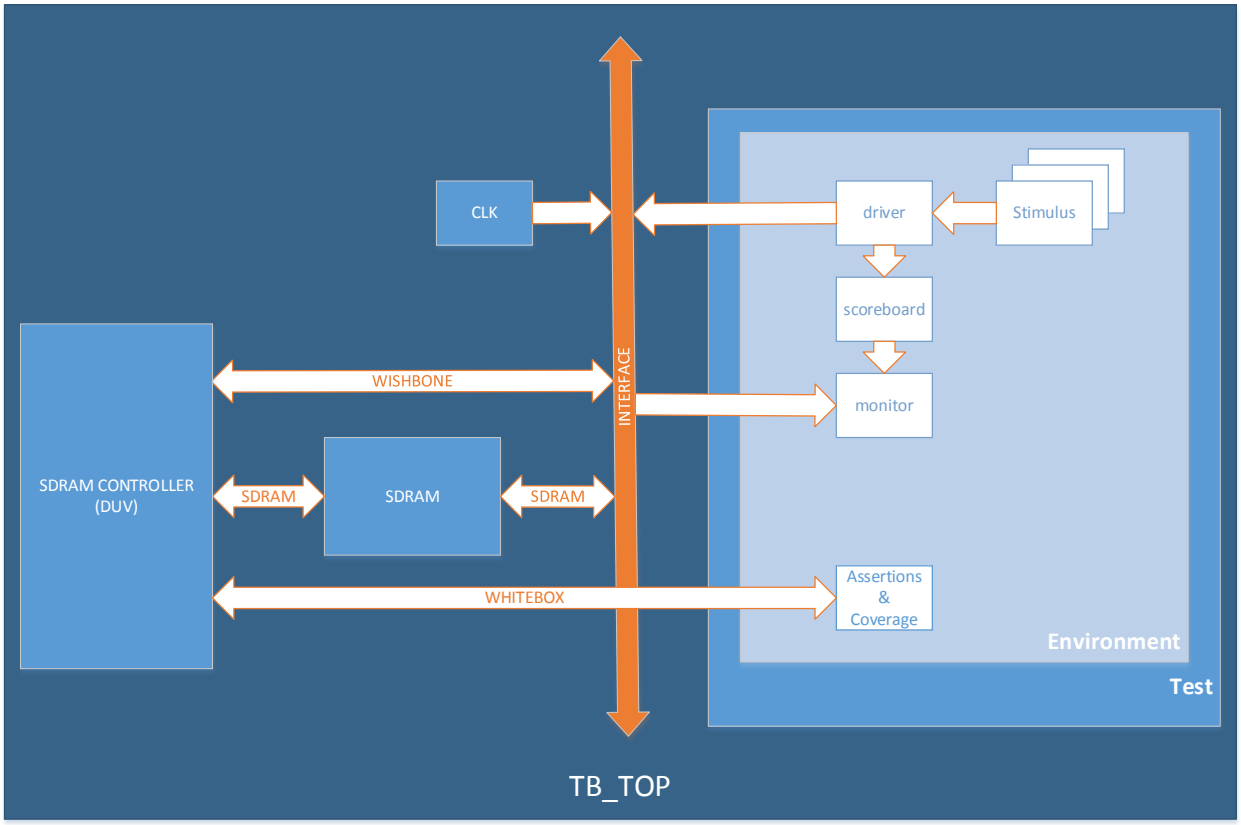

Figure 3 Testbench basado en capas para la implementar la estrategia de verificación.

Este top del testbench contiene:

- SDRAM controller: RTL a verificar.
- SDRAM: modelo de SDRAM 8/16/32 bits.
- Bloque CLK: genera los relojes para el controlador,la SDRAM y el test.
- Bloque interface: incluye las señales de reloj, interfaz WISHBONE e interfaz SDRAM.
- Bloque test: un módulo program que incluye una clase environment, driver, scoreboard y monitor.

La clase driver contiene los siguientes métodos:

- Reset: Inicializa la memoria DRAM.
- Burst_write: ejecuta una escritura a la memoria DRAM y almacena la información de la escritura en el scoreboard.
- Clases de estimulos: generan de manera aleatoria las direcciones de memoria usando restricciones para casos determinísticos.

La clase monitor con el siguiente método:

- Burst_read: ejecuta una lectura a la memoria DRAM y compara con la información obtenida de la lectura con la información de la escritura almacenada en el scoreboard.
- La clase scoreboard incluye un queue para datos, dirección, tamaño de ráfaga de escritura.

IV. Lista de casos de prueba

En esta sección del plan de verificación se describen los objetivos de estímulo destinados al ambiente de verificación. Los objetivos deben cubrir todos los requisitos funcionales del estímulo y asegurarse de que los componentes de estímulo en el ambiente de verificación hacen lo que deberían.

La cobertura es el mecanismo de retroalimentación que evalúa la calidad de los componentes de generación de estímulos del ambiente verificación. El medio de verificación proporciona "control de calidad" en el DUV.

Las aserciones tienen como objetivo la implementación del diseño. Formalizan condiciones dentro de diseño que deben cumplirse durante todo el tiempo. Los checkers y las aserciones tienen propósitos relacionados, los checkers realizan el chequeo dentro del test bench las aserciones realizan el chequeo dentro del RTL. Una perspectiva sistemática del uso de aserciones es la noción de un diseño HDL defensivo donde cada aserción chequea un cono de lógica de condiciones frontera y protege a la lógica subsiguiente.

Casos de pruebas: Función “Programmable column address”

Table 3 Casos de prueba para la dirección de columna programable.

Numero de referencia de prueba	Descripción	Puntos de cobertura	Aserciones
PCA_1	Escribir y leer en el mismo banco, fila y columna para los bits de configuración de columna.	Cantidad de veces que se escribe en la misma dirección de memoria es la cantidad de estímulos generados.	La dirección de memoria en el SDRAM request generator ante la variación de los bits de configuración de memoria es la misma que la solicitada.
PCA_2	Escribir y leer en los 4 bancos usando la misma columna y fila con distinta configuración de columna.	Cantidad de veces que se escribe en la misma dirección de bancos, fila y columna.	El banco está listo para recibir una solicitud de escritura
PCA_3	Realizar PCA_1 y PCA_2 para una SDRAM de 8/16/32 bits.	PCA_1 y PCA_2	PCA_1 y PCA_2

Casos de pruebas: Función “Support SDRAM with four bank”

Table 4 Casos de prueba para la función de soporte de SDRAM con 4 bancos.

Numero de referencia de prueba	Descripción	Puntos de cobertura	Aserciones
S4B_1	Escribir y leer en los 4 bancos para cada una de las 4 filas usando valores de columna random y una configuración de columna de 2’b00.	Cantidad de veces que se escribe en la misma dirección de memoria es cantidad de estímulos generados.	El banco está listo para recibir una solicitud de escritura
S4B_2	Escribir y leer en direcciones aleatorias de memoria para los distintos bancos de memoria generando un cruce de banco	Cantidad de veces que realiza el cruce de banco.	El RTL notifica si hubo un cruce de banco.
S4B_3	Realizar S4B_1 y S4B_2 para una SDRAM de 8/16/32 bits.	PCA_1 y PCA_2	PCA_1 y PCA_2

Casos de pruebas: Ambas funciones”

Table 5 Casos de prueba para la función de soporte de SDRAM con 4 bancos.

Numero de referencia de prueba	Descripción	Puntos de cobertura	Aserciones
AF_1	Escribir y leer en direcciones de memoria aleatoria para todas las configuraciones de columna.	Cantidad de veces que se escribe en la misma dirección de bancos, fila y columna.	El banco está listo para recibir una solicitud de escritura
AF_2	Realizar S4B_1 y S4B_2 para una SDRAM de 8/16/32 bits.	PCA_1 y PCA_2	PCA_1 y PCA_2