

Plan de verificación: “SDRAM CONTROLLER”

Autores:

Ismael Jiménez Carballo - B94009

Kristel Herrera Rodríguez - C13769

Luis Felipe Aguero Peralta - C10089

Leonardo Leiva Vasquez - C14172

Fecha

14 de Julio del 2024

Revisiones

Número de Revisión	Fecha de revisión	Descripción	Autor
1	6/7/24	Revisión parcial de la función CAS LATENCY	Luis Felipe Agüero
2	8/7/24	Revisión completa de la función CAS LATENCY	Leonardo Leiva Vasquez
3	9/7/24	Revisión completa del CAS LATENCY	Luis Felipe Agüero Peralta
4	9/7/24	Revisión de la función Data mask signals for partial write operations	Kristel Herrera Rodríguez
5	12/7/24	Revisión del Data mask signals for partial write operations	Kristel Herrera Rodríguez
6	12/7/24	Revisión de la función Support SDRAM with four bank	Luis Felipe Agüero Peralta e Ismael Jimenez Carballo
7	13/7/24	Revisión de Automatic controlled refresh	Luis Felipe Agüero Peralta
8	13/7/24	Revisión de función Support SDRAM with four bank	Ismael Jimenez Carballo
9	13/7/24	Revisión de One-chip signal selected	Ismael Jimenez Carballo
10	13/7/24	Revisión de Data mask signals for partial write operations	Ismael Jimenez Carballo
11	13/7/24	Revisión de Data mask signals for partial write operations	Kristel Herrera Rodríguez
12	13/7/24	Revisión de One-chip signal selected	Leonardo Leiva Vasquez
13	13/7/24	División del address para escritura en rangos de 25%, 50%, 75% y 100% de la memoria	Leonardo Leiva Vasquez
14	14/7/24	Revisión del programa en conjunto, funciones implementadas en un mismo ambiente de trabajo	Ismael Jimenez Carballo
15	14/7/24	Revisión de la función print memory contents	Leonardo Leiva Vasquez
16	14/7/24	Revision de Data mask signals for partial write operations y errores en el código general	Kristel Herrera Rodríguez

Table of Contents

1. Estrategia de verificación:	5
a. Describir si es aleatoria o dirigida.	5
b. Describir si es caja negra, blanca o gris.	5
2. Niveles de verificación	6
a. Descripción de la jerarquía de verificación	6
3. Ambiente verificación:	7
a. Describir nivel abstracción, capas y metodología.	7
Metodología de Verificación	8
4. Alcance del plan de verificación	9
5. Requerimientos de cobertura y métricas.	9
a. Describir tipo de cobertura estructural y funcional a utilizar.	9
b. Definir metas del proyecto.	10
6. Lista de bugs encontrados	11
Reset:	13
7. Función #1	14
8. Función #2	15
9. Función #3	16
10. Función #4	17
11. Función #5	18
12. Función #6	19
13. Función #7	20
14. Función #8	21

1. Estrategia de verificación:

a. Describir si es aleatoria o dirigida.

La estrategia de verificación utilizada en este proyecto es predominantemente aleatoria con elementos de verificación dirigida, esto se hizo así debido a que como se vió a lo largo del curso, esta combinación ayuda a garantizar que se verifiquen tanto los casos de uso esperados como los inesperados. Para este proyecto, se utilizan secuencias aleatorias mediante la clase `gen_item_seq` en `driver.sv`, lo que permite explorar diversos estados del DUV y detectar posibles errores no previstos, y aunque la verificación es predominantemente aleatoria, como se mencionó hay algunos elementos de verificación dirigida que se utilizan para asegurar que ciertos aspectos críticos del diseño se prueban exhaustivamente un ejemplo de lo implementado fueron las aserciones que son una forma de verificación dirigida, ya que verificaron condiciones específicas y críticas en el diseño.

b. Describir si es caja negra, blanca o gris.

Este proyecto utiliza una verificación de Gray Box porque combina elementos de las metodologías de caja negra y caja blanca para lograr una cobertura exhaustiva. Se utilizan aserciones internas, como se muestra en ``assertions.sv``, para verificar condiciones críticas dentro del diseño, tales como la correcta activación de las señales de control (``sdr_cs_n``, ``sdr_cas_n``, ``sdr_ras_n``, ``sdr_we_n``), lo que demuestra un conocimiento parcial de la implementación interna del DUV. Además, en ``coverage.sv``, se definen coberturas funcionales específicas que monitorean combinaciones de señales internas, asegurando que todos los escenarios críticos del diseño sean probados. Por ejemplo, la cobertura funcional para las señales de selección de chip y de escritura (``sdr_cs_n``, ``wb_we_i``) garantiza que se evalúen todas las combinaciones importantes de estas señales. Finalmente, la generación de estímulos aleatorios a través de secuencias definidas en ``driver.sv`` considera parámetros internos del diseño, como la latencia CAS y las configuraciones de auto-refresh, lo que guía las pruebas de manera dirigida y específica.

2. Niveles de verificación

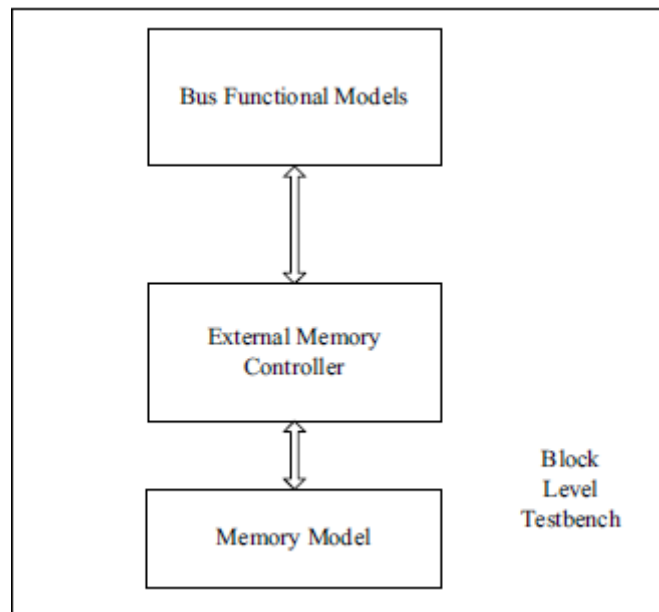
a. Descripción de la jerarquía de verificación

La jerarquía de verificación se estructura en tres niveles principales:

Modelos Funcionales del Bus (Bus Functional Models): Este nivel se enfoca en la simulación de los componentes funcionales del bus, verificando que el diseño cumpla con las especificaciones del protocolo del bus. Se utilizan modelos abstractos que emulan el comportamiento de los dispositivos conectados al bus, permitiendo la identificación de problemas en las primeras etapas del desarrollo.

Controlador de Memoria Externa (External Memory Controller): Este nivel se encarga de la verificación del controlador de memoria externa, asegurando que interactúe correctamente con los modelos funcionales del bus y el modelo de memoria. Se prueban diversas configuraciones y escenarios de uso para garantizar la robustez y funcionalidad del controlador.

Modelo de Memoria (Memory Model): En este nivel, se verifica el modelo de memoria que representa las memorias físicas utilizadas en el diseño. Se aseguran operaciones correctas de lectura y escritura, manejo adecuado de tiempos de acceso y respuesta, y la correcta gestión de la memoria bajo diversas condiciones operativas



3. Ambiente verificación:

a. Describir nivel abstracción, capas y metodología.

El ambiente de verificación se organiza en varias capas y niveles de abstracción:

Nivel de Abstracción:

- En este caso, se maneja en un nivel de comandos y secuencias, lo que significa que se generan y verifican secuencias de operaciones de lectura y escritura en la memoria SDRAM.

Capas:

1. La capa de señales:

Esta capa actúa como el intermediario directo entre el driver y el DUT, manejando las señales eléctricas o lógicas que se intercambian entre ellos. Su función principal es garantizar que las señales generadas por el driver sean entregadas de forma precisa al DUT, y que las señales de respuesta del DUT sean capturadas y dirigidas adecuadamente hacia el monitor.

2. La capa de comandos:

- **Driver:** El driver convierte las transacciones abstractas en señales específicas del DUT, interactuando directamente con las interfaces del DUT para aplicar los estímulos.

- **Monitor:** El monitor observa las salidas del DUT, capturando las respuestas para su análisis posterior.
- **Assertions:** Las aserciones son verificaciones embebidas que aseguran que ciertas condiciones se cumplan durante la simulación, proporcionando una validación continua de propiedades críticas.

3. La capa Funcional:

- **Agente:** El agente coordina la interacción entre el generador y el driver, asegurando que los estímulos se envíen correctamente al DUT.
- **Scoreboard:** El scoreboard compara las salidas observadas del DUT con los resultados esperados, identificando discrepancias y validando el comportamiento correcto del DUT.
- **Checker:** El checker realiza verificaciones adicionales sobre las salidas capturadas por el monitor, asegurando que se cumplan todas las condiciones y reglas de diseño especificadas.

4. La capa Escenario:

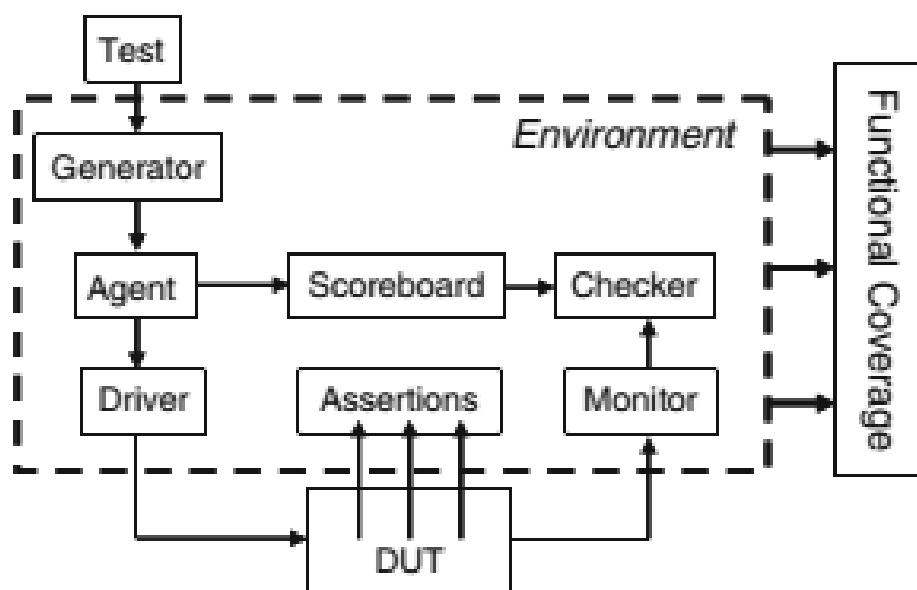
- **Generator:** El generador crea estímulos de prueba a partir de los escenarios definidos en los tests. Estos estímulos se basan en transacciones abstractas que representan las operaciones a realizar en el DUT.

Metodología de Verificación

La metodología utilizada es UVM (Universal Verification Methodology), que es un marco basado en SystemVerilog para la verificación de circuitos integrados.

- **Verificación Aleatoria:** Se utiliza para generar una amplia variedad de casos de prueba, lo que permite detectar errores que no se encontraron con pruebas determinísticas.
- **Verificación Determinística:** Para funciones simples y casos de prueba específicos.
- **Cobertura Estructural:** Se realizará cobertura de código después de completar todos los tests. Esto nos ayudará a identificar y focalizar los esfuerzos en las áreas del código que no han sido probadas, asegurando una verificación del diseño.

- **Cobertura Funcional:** Se implementarán puntos de cobertura en los testbenches que nos permitirán medir y garantizar que todas las funcionalidades esperadas del diseño sean probadas.
- **Aserciones:** Se utilizarán aserciones en los testbenches. Esto permite monitorear el comportamiento del diseño bajo prueba y detectar fallos.
- **Pruebas Random:** Además de las pruebas dirigidas, las pruebas aleatorias serán implementadas para explorar escenarios menos predecibles.



4. Alcance del plan de verificación

El plan de verificación abarcó la validación exhaustiva del controlador SDRAM a través de pruebas aleatorias y dirigidas que aseguran que todas las funciones críticas y comportamientos esperados del diseño sean correctamente implementados. Este plan incluyó la creación de secuencias de estímulos, aserciones, y coberturas funcionales que se enfocan en las señales clave y estados del sistema.

5. Requerimientos de cobertura y métricas.

a. Describir tipo de cobertura estructural y funcional a utilizar.

Para el análisis de cobertura estructural, se decidió integrar únicamente la parte del Wishbone. Esto se debió a que el DUV presentaba algunas inconsistencias en el análisis estructural general. Se considera que la **Cobertura de Memoria (Memory Coverage)** es el tipo de cobertura estructural más adecuado debido a que asegura que todas las ubicaciones de memoria sean ejercitadas, verificando tanto las operaciones de lectura como de escritura, lo cual es importante para el diseño de Wishbone que maneja transacciones de memoria. Esta cobertura identifica áreas de memoria no accedidas, previniendo errores ocultos y garantizando una verificación completa y robusta. Además, asegura que todas las transacciones de memoria cumplen con las especificaciones del protocolo de Wishbone y maneja correctamente las condiciones de borde relacionadas con la memoria.

En cobertura funcional se decidió utilizar el subtipo de cobertura de funcionalidades, para asegurar que todas las funcionalidades del controlador SDRAM fueran verificadas exhaustivamente en driver.sv, se definieron tareas específicas para las operaciones de lectura (burst_read) y escritura (burst_write), asegurando que todas las combinaciones posibles de señales internas se evaluarán durante estas operaciones. También se utilizó la tarea cas_latency para configurar y verificar la latencia CAS del controlador SDRAM, asegurando que las operaciones respetaran los tiempos de latencia configurados. En coverage.sv, se definieron grupos de cobertura, como cov_chip_select, para asegurarse de que la señal de selección de chip (sdr_cs_n) y otras señales relacionadas (wb_we_i) se evaluarán correctamente durante las operaciones. Además, la tarea auto_refresh en driver.sv configuró y verificó las operaciones de auto-refresh del controlador, probando esta funcionalidad crítica bajo diversas configuraciones.

Cabe resaltar que también se utilizó cobertura basada en aserciones para capturar y verificar condiciones internas críticas del diseño. En assertions.sv, se definieron propiedades para verificar el correcto funcionamiento de las señales de control y de los comandos de lectura y escritura. Por ejemplo, la propiedad check_read_command valida que los datos se reciban correctamente después de los ciclos de espera configurados, mientras que la propiedad

check_chip_select_signal verifica que la señal de selección de chip (sdr_cs_n) se active correctamente en conjunto con otras señales de control (sdr_cas_n, sdr_ras_n, sdr_we_n). Estas aserciones permiten detectar y reportar errores internos del diseño, complementando la cobertura funcional y proporcionando una verificación más robusta y completa.

b. Definir metas del proyecto.

La primera meta es lograr una cobertura funcional parcial del controlador SDRAM. Esto implica validar exhaustivamente varias de las funcionalidades del controlador, como las operaciones de lectura y escritura, la configuración de la latencia CAS y las operaciones de auto-refresh. Para ello, se han definido tareas específicas en driver.sv, como burst_read, burst_write, cas_latency y auto_refresh, y se han utilizado grupos de cobertura en coverage.sv para asegurar que se evalúen todas las combinaciones relevantes de señales internas.

La segunda meta es alcanzar una cobertura estructural y de aserciones robusta. Esto asegura que todos los caminos posibles en el diseño sean ejecutados y que se cumplan las condiciones críticas internas. Se han utilizado aserciones en assertions.sv para verificar el correcto funcionamiento de señales de control y comandos, como check_read_command y check_chip_select_signal, complementando la cobertura estructural con coberturas definidas en coverage.sv para evaluar todas las combinaciones posibles de señales internas.

Además, se busca verificar exhaustivamente las señales críticas del controlador SDRAM, como sdr_cs_n, sdr_cas_n, sdr_ras_n y sdr_we_n, para asegurar su comportamiento correcto durante las operaciones. Esta meta se logra mediante tareas específicas y aserciones que verifican el comportamiento de estas señales, asegurando que respondan correctamente en diversas condiciones operativas.

Otra meta más, es cubrir todos los escenarios de uso previstos del controlador SDRAM, incluyendo tanto las operaciones normales como las condiciones de error. Para esto, se han definido secuencias de estímulos aleatorios y dirigidos

en `gen_item_seq` y se han utilizado tareas en `driver.sv` para simular una variedad de operaciones y configuraciones, asegurando una verificación completa de todos los casos de uso.

Además, el proyecto busca automatizar la verificación para mejorar la eficiencia y la cobertura. Utilizando la metodología UVM, se ha estructurado y automatizado el entorno de verificación, definiendo secuencias, monitores y scoreboards que facilitan la ejecución automática y repetitiva de pruebas, mejorando así la efectividad de la verificación.

6. Lista de bugs encontrados

Bug	Descripción
Señal Active en la descripción del Automatic controlled refresh.	En las especificaciones del auto-refresh, se indica el patrón de precarga, auto-refresh y activo. El estado activo no ocurre durante las lecturas por los comandos.
Aplicación de máscaras al final de rafagas de escritura.	En las escrituras de máscaras, suele haber fallos en las penúltimas o últimas escrituras; no están presentes cuando se leen.
No se recibe la confirmación (acknowledge) para algunas configuraciones.	Para ciertas configuraciones del autorefresh, se ocasiona que el acknowledge no se reciba, lo que provoca errores en las operaciones de lectura y escritura
Datos en direcciones de memoria no escritas	Se detectan datos no esperados en direcciones que no se han utilizado en operaciones de escritura previas. Estos datos aparecen en algunas direcciones de la memoria pero nunca fueron escritos ni introducidos en el scoreboard.

Reset:

Función a verificar:	Reset
Sección de la especificación:	5.3
Descripción:	La función reset restablece todas las señales de control del bus y asegura que el sistema SDRAM esté en un estado conocido antes de iniciar las operaciones normales. Esto incluye poner las señales en sus estados iniciales, aplicar un reset general, y esperar a que el SDRAM se inicialice completamente..
Prioridad:	Crítica
Autor (Dueño de la función):	Todos los integrantes del equipo
Requerimientos del banco de pruebas:	Debe haber capacidad para verificar que la señal sdr_init_done se active después del reset.
Pruebas:	Aplicar el reset y verificar que todas las señales se restablecen a sus valores iniciales. Confirmar que el SDRAM se inicializa correctamente (sdr_init_done = 1) después del reset. Medir los tiempos de retardo para asegurarse de que el reset se aplica correctamente.
Criterio de chequeo:	El controlador debe permitir operaciones de lectura y escritura después de aplicar el reset. Verificar que no haya interrupciones o errores registrados durante el proceso de reset..
Cobertura funcional:	Comprobar el estado del módulo SDRAM antes y después de aplicar el reset. Se utiliza el init_done que nos indica que la memoria está correctamente inicializada.
Observaciones:	<ul style="list-style-type: none">• Asegurarse de que los tiempos de retardo se midan con precisión durante el proceso de reset.• El reset debe aplicarse de manera consistente y confiable cada vez que se invoca la función.

7. Función #1

Función a verificar:	Support SDRAM with four bank													
Sección de la especificación:	4													
Descripción:	<div>SDRAM Address Mapping Format</div> <table><tr><td>Row Address[11:0]</td><td>Bank Address[1:0]</td><td>Column Address[11:0]</td></tr></table> <p>Column, Bank and Row Address decoding base on configuration:</p> <table><tr><td>cfg_col_bits</td><td></td></tr><tr><td>2'b00</td><td>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]};</td></tr><tr><td>2'b01</td><td>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]};</td></tr><tr><td>2'b10</td><td>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]};</td></tr><tr><td>2'b11</td><td>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]};</td></tr></table>	Row Address[11:0]	Bank Address[1:0]	Column Address[11:0]	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]};
Row Address[11:0]	Bank Address[1:0]	Column Address[11:0]												
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]};													
Prioridad:	Crítica													
Autor (Dueño de la función):	Leonardo Leiva Vasquez													
Requerimientos del banco de pruebas:	Selección de Column, bank, row del address.													
Pruebas:	Prueba aleatoria: aleatorizar bank del cfg_col_bits, aleatorizar direcciones para selección entre 0-3 bank. Cantidad de escrituras/lecturas > 32 sin repetición.													
Criterio de chequeo:	Predecir la dirección DDR y comparar con la dirección enviada dado en cfg_col_bits.													
Cobertura funcional:	Cfg_col_bits 2 4 opciones. Para cada cfg_col_bits, utilizar todos los bancos. Tanto para lectura/escritura.													
Observaciones:	Ninguna													

8. Función #2

Función a verificar:	Burst write operation in SDRAM controller																																												
Sección de la especificación:	5.3 SDRAM Interface (SDRAM Overview y Functional Description) Tabla de comandos SDRAM (página 7)																																												
Descripción:	<p>Verificar que la operación de escritura por ráfaga (burst write) en el controlador SDRAM se realice correctamente según la especificación, incluyendo la configuración y manejo de señales relevantes como wb_stb_i, wb_cyc_i, wb_we_i, y wb_addr_i.</p> <p>2.2 WISHBONE interface signals</p> <table><tr><th>Port</th><th>Width</th><th>Direction</th><th>Description</th></tr><tr><td>wb_clk_i</td><td>1</td><td>Input</td><td>Master clock</td></tr><tr><td>wb_rst_i</td><td>1</td><td>Input</td><td>Synchronous reset, active high</td></tr><tr><td>wb_addr_i</td><td>32</td><td>Input</td><td>Lower address bits</td></tr><tr><td>wb_dat_i</td><td>32</td><td>Input</td><td>Data towards the core</td></tr><tr><td>wb_dat_o</td><td>32</td><td>Output</td><td>Data from the core</td></tr><tr><td>wb_sel_i</td><td>4</td><td>Input</td><td>Write Data Valid</td></tr><tr><td>wb_we_i</td><td>1</td><td>Input</td><td>Write enable input</td></tr><tr><td>wb_stb_i</td><td>1</td><td>Input</td><td>Strobe signal/Core select input</td></tr><tr><td>wb_cyc_i</td><td>1</td><td>Input</td><td>Valid bus cycle input</td></tr><tr><td>wb_ack_o</td><td>1</td><td>Output</td><td>Bus cycle acknowledge output</td></tr></table>	Port	Width	Direction	Description	wb_clk_i	1	Input	Master clock	wb_rst_i	1	Input	Synchronous reset, active high	wb_addr_i	32	Input	Lower address bits	wb_dat_i	32	Input	Data towards the core	wb_dat_o	32	Output	Data from the core	wb_sel_i	4	Input	Write Data Valid	wb_we_i	1	Input	Write enable input	wb_stb_i	1	Input	Strobe signal/Core select input	wb_cyc_i	1	Input	Valid bus cycle input	wb_ack_o	1	Output	Bus cycle acknowledge output
Port	Width	Direction	Description																																										
wb_clk_i	1	Input	Master clock																																										
wb_rst_i	1	Input	Synchronous reset, active high																																										
wb_addr_i	32	Input	Lower address bits																																										
wb_dat_i	32	Input	Data towards the core																																										
wb_dat_o	32	Output	Data from the core																																										
wb_sel_i	4	Input	Write Data Valid																																										
wb_we_i	1	Input	Write enable input																																										
wb_stb_i	1	Input	Strobe signal/Core select input																																										
wb_cyc_i	1	Input	Valid bus cycle input																																										
wb_ack_o	1	Output	Bus cycle acknowledge output																																										
Prioridad:	Crítica																																												
Autor (Dueño de la función):	Kristel Herrera Rodriguez																																												
Requerimientos del banco de pruebas:	<p>Generación de estímulos para la operación de burst write.</p> <p>Monitoreo de las señales durante la operación.</p> <p>Verificación de la correcta escritura de datos en las direcciones de memoria especificadas, en rangos de 25%, 50%, 75% y 100% de la memoria</p>																																												
Pruebas:	<p>Ejecutar secuencias de escritura con diferentes longitudes de ráfaga.</p> <p>Comparar los datos escritos en la memoria con los valores esperados.</p> <p>Verificar la correcta activación de las señales de control durante la escritura.</p>																																												
Criterio de chequeo:	<p>Las direcciones de memoria deben actualizarse correctamente en cada ciclo de la ráfaga.</p> <p>Los datos escritos en memoria deben coincidir con los valores especificados en la secuencia.</p>																																												
Cobertura funcional:	<p>Cobertura de todas las combinaciones posibles de datos.</p> <p>Cobertura de las señales de control (wb_stb_i, wb_cyc_i, wb_we_i, wb_addr_i). Todas en alto.</p>																																												
Observaciones:	Ninguna																																												

9. Función #3

Función a verificar:	Burst read operation in SDRAM controller																																												
Sección de la especificación:	5.3 SDRAM Interface (SDRAM Overview y Functional Description) Tabla de comandos SDRAM (página 7)																																												
Descripción:	<p>Verificar que la operación de lectura por ráfaga (burst read) en el controlador SDRAM se realice correctamente según la especificación, incluyendo la configuración y manejo de señales relevantes como wb_stb_i, wb_cyc_i, wb_we_i, y wb_addr_i.</p> <p>2.2 WISHBONE interface signals</p> <table><tr><th>Port</th><th>Width</th><th>Direction</th><th>Description</th></tr><tr><td>wb_clk_i</td><td>1</td><td>Input</td><td>Master clock</td></tr><tr><td>wb_rst_i</td><td>1</td><td>Input</td><td>Synchronous reset, active high</td></tr><tr><td>wb_addr_i</td><td>32</td><td>Input</td><td>Lower address bits</td></tr><tr><td>wb_dat_i</td><td>32</td><td>Input</td><td>Data towards the core</td></tr><tr><td>wb_dat_o</td><td>32</td><td>Output</td><td>Data from the core</td></tr><tr><td>wb_sel_i</td><td>4</td><td>Input</td><td>Write Data Valid</td></tr><tr><td>wb_we_i</td><td>1</td><td>Input</td><td>Write enable input</td></tr><tr><td>wb_stb_i</td><td>1</td><td>Input</td><td>Strobe signal/Core select input</td></tr><tr><td>wb_cyc_i</td><td>1</td><td>Input</td><td>Valid bus cycle input</td></tr><tr><td>wb_ack_o</td><td>1</td><td>Output</td><td>Bus cycle acknowledge output</td></tr></table>	Port	Width	Direction	Description	wb_clk_i	1	Input	Master clock	wb_rst_i	1	Input	Synchronous reset, active high	wb_addr_i	32	Input	Lower address bits	wb_dat_i	32	Input	Data towards the core	wb_dat_o	32	Output	Data from the core	wb_sel_i	4	Input	Write Data Valid	wb_we_i	1	Input	Write enable input	wb_stb_i	1	Input	Strobe signal/Core select input	wb_cyc_i	1	Input	Valid bus cycle input	wb_ack_o	1	Output	Bus cycle acknowledge output
Port	Width	Direction	Description																																										
wb_clk_i	1	Input	Master clock																																										
wb_rst_i	1	Input	Synchronous reset, active high																																										
wb_addr_i	32	Input	Lower address bits																																										
wb_dat_i	32	Input	Data towards the core																																										
wb_dat_o	32	Output	Data from the core																																										
wb_sel_i	4	Input	Write Data Valid																																										
wb_we_i	1	Input	Write enable input																																										
wb_stb_i	1	Input	Strobe signal/Core select input																																										
wb_cyc_i	1	Input	Valid bus cycle input																																										
wb_ack_o	1	Output	Bus cycle acknowledge output																																										
Prioridad:	Crítica																																												
Autor (Dueño de la función):	Luis Felipe Aguero Peralta																																												
Requerimientos del banco de pruebas:	<p>Generación de estímulos para la operación de burst read.</p> <p>Verificación de la correcta lectura de datos desde las direcciones de memoria especificadas en las que ya se escribieron anteriormente..</p>																																												
Pruebas:	<p>Ejecutar secuencias de lectura con diferentes longitudes de ráfaga.</p> <p>Comparar los datos leídos de la memoria con los valores esperados.</p>																																												
Criterio de chequeo:	<p>Las direcciones de memoria deben actualizarse correctamente en cada ciclo de la ráfaga.</p> <p>Los datos leídos de la memoria sí coinciden con los valores esperados según la secuencia.</p>																																												
Cobertura funcional:	<p>Cobertura de todas las combinaciones posibles de datos.</p> <p>Cobertura de las señales de control (wb_stb_i, wb_cyc_i, wb_we_i, wb_addr_i). Todas en alto, (wb_we_i = 0)</p>																																												
Observaciones:	Ninguna.																																												

10. Función #4

Función a verificar:	Full memory read operation in SDRAM controller (print_memory_contents)
Sección de la especificación:	N/A (Función realizada para pruebas)
Descripción:	Verificar que la operación de lectura completa de memoria (print_memory_contents) en el controlador SDRAM se realice correctamente, cubriendo todas las direcciones de la memoria y asegurando la correcta lectura de los datos almacenados.
Prioridad:	Crítica
Autor (Dueño de la función):	Leonardo Leiva Vásquez
Requerimientos del banco de pruebas:	Generación de estímulos para la operación de lectura completa de memoria. Monitoreo de las señales durante la operación. Verificación de la correcta lectura de datos desde todas las partes de la memoria(25%, 50%, 75% y 100%)
Pruebas:	Ejecutar secuencias de lectura completa de memoria desde la dirección 0 hasta el final de la memoria. Comparar los datos leídos de la memoria con los valores esperados.
Criterio de chequeo:	Las direcciones de memoria deben actualizarse correctamente en cada ciclo. Los datos leídos de la memoria deben coincidir con los valores esperados según la secuencia (scoreboard)
Cobertura funcional:	Se espera que se lean datos en la memoria que pertenezcan a alguno de los 4 rangos establecidos: <ul style="list-style-type: none"> • bins range_0 = {[32'h00000000 : 32'h000000C7]}; • bins range_1 = {[32'h1FFFFFF38 : 32'h200000BF]}; • bins range_2 = {[32'h3FFFFFF38 : 32'h400000BF]}; • bins range_3 = {[32'h7FFFFFF38 : 32'h800000BF]};
Observaciones:	Ninguna

11. Función #5

Función a verificar:	CAS latency
Sección de la especificación:	5
Descripción:	<p>La CAS Latency es el retraso, medido en ciclos de reloj, entre el registro de un comando de lectura y la disponibilidad del primer dato de salida. Esta latencia puede configurarse en dos o tres ciclos de reloj. La latencia CAS depende de la velocidad del reloj y la frecuencia operativa del SDRAM.</p> <p>The diagrams illustrate the CAS latency for two configurations: CL=2 and CL=3. In the CL=2 diagram, the command sequence is READ at T0, followed by NOP at T1 and T2. The first data is available at T2. In the CL=3 diagram, the command sequence is READ at T0, followed by NOP at T1, T2, and T3. The first data is available at T3. Both diagrams show the timing parameters tLZ (load-to-zero), tOH (output hold), and tAC (access time).</p>
Prioridad:	Crítica
Autor (Dueño de la función):	Ismael Jimenez Carballo
Requerimientos del banco de pruebas:	Permitir la configuración de diferentes latencias CAS (2 y 3 ciclos)
Pruebas:	El banco de pruebas debe cambiar aleatoriamente los parámetros <code>cfg_sdr_mode_reg</code> y <code>cfg_sdr_cas</code> para probar diferentes latencias CAS. Específicamente, debe configurar <code>cfg_sdr_mode_reg</code> a 13h033 y <code>cfg_sdr_cas</code> a 3h3 para un retardo de 3 ciclos, y <code>cfg_sdr_mode_reg</code> a 13h023 y <code>cfg_sdr_cas</code> a 3h2 para un retardo de 2 ciclos.
Criterio de chequeo:	El criterio de chequeo incluye verificar que los datos se leen 2 ciclos después del comando de lectura si la latencia CAS es 2, y 3 ciclos después si la latencia CAS es 3. También se debe asegurar que los parámetros <code>cfg_sdr_mode_reg</code> y <code>cfg_sdr_cas</code> se configuren correctamente y comparar los datos leídos con los esperados.
Cobertura funcional:	La cobertura funcional incluye probar configuraciones de latencia CAS de 2 y 3 ciclos, asegurando que ambas configuraciones ocurran durante operaciones de lectura y escritura, y verificando la correcta inicialización del SDRAM en diversas situaciones.
Observaciones:	Ninguna

12. Función #6

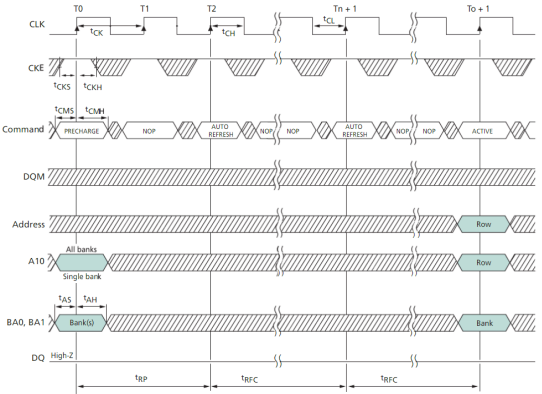
Función a verificar:	Data mask signals for partial write operations																																												
Sección de la especificación:	2.2																																												
Descripción:	<p>Las "Data mask signals" permiten modificar selectivamente partes específicas de una palabra de memoria durante una escritura, sin afectar el resto de la palabra, para esto se utiliza el señal wb_sel_i del interfaz Wishbone especifica qué bytes dentro de un bus de datos deben ser activos durante una transferencia.</p> <p>2.2 WISHBONE interface signals</p> <table><tr><th>Port</th><th>Width</th><th>Direction</th><th>Description</th></tr><tr><td>wb_clk_i</td><td>1</td><td>Input</td><td>Master clock</td></tr><tr><td>wb_rst_i</td><td>1</td><td>Input</td><td>Synchronous reset, active high</td></tr><tr><td>wb_adr_i</td><td>32</td><td>Input</td><td>Lower address bits</td></tr><tr><td>wb_dat_i</td><td>32</td><td>Input</td><td>Data towards the core</td></tr><tr><td>wb_dat_o</td><td>32</td><td>Output</td><td>Data from the core</td></tr><tr><td>wb_sel_i</td><td>4</td><td>Input</td><td>Write Data Valid</td></tr><tr><td>wb_we_i</td><td>1</td><td>Input</td><td>Write enable input</td></tr><tr><td>wb_stb_i</td><td>1</td><td>Input</td><td>Strobe signal/Core select input</td></tr><tr><td>wb_cyc_i</td><td>1</td><td>Input</td><td>Valid bus cycle input</td></tr><tr><td>wb_ack_o</td><td>1</td><td>Output</td><td>Bus cycle acknowledge output</td></tr></table>	Port	Width	Direction	Description	wb_clk_i	1	Input	Master clock	wb_rst_i	1	Input	Synchronous reset, active high	wb_adr_i	32	Input	Lower address bits	wb_dat_i	32	Input	Data towards the core	wb_dat_o	32	Output	Data from the core	wb_sel_i	4	Input	Write Data Valid	wb_we_i	1	Input	Write enable input	wb_stb_i	1	Input	Strobe signal/Core select input	wb_cyc_i	1	Input	Valid bus cycle input	wb_ack_o	1	Output	Bus cycle acknowledge output
Port	Width	Direction	Description																																										
wb_clk_i	1	Input	Master clock																																										
wb_rst_i	1	Input	Synchronous reset, active high																																										
wb_adr_i	32	Input	Lower address bits																																										
wb_dat_i	32	Input	Data towards the core																																										
wb_dat_o	32	Output	Data from the core																																										
wb_sel_i	4	Input	Write Data Valid																																										
wb_we_i	1	Input	Write enable input																																										
wb_stb_i	1	Input	Strobe signal/Core select input																																										
wb_cyc_i	1	Input	Valid bus cycle input																																										
wb_ack_o	1	Output	Bus cycle acknowledge output																																										
Prioridad:	Crítica																																												
Autor (Dueño de la función):	Kristel Herrera Rodriguez																																												
Requerimientos del banco de pruebas:	<p>Generación de señales de máscara para verificar la escritura parcial.</p> <p>Verificación de la no alteración de los bits no mascarados.</p>																																												
Pruebas:	<p>Prueba con todas las combinaciones de bits de máscara activados y desactivados.</p> <p>Validación de escrituras parciales en la memoria SDRAM.</p>																																												
Criterio de chequeo:	<p>Comprobar que sólo los bits permitidos por las señales de máscara son modificados en la memoria. Los datos leídos después de la escritura deben coincidir exactamente con los patrones de escritura esperados, respetando las máscaras aplicadas.</p>																																												
Cobertura funcional:	<p>Evalúa el rango de valores de la señal wb_sel_i, e han definido varios rangos para esta señal:</p> <ul style="list-style-type: none">• range_0_3: abarca valores de 0 a 3.• range_4_7: abarca valores de 4 a 7.• range_8_11: abarca valores de 8 a 11.• range_12_15: abarca valores de 12 a 15. <p>Cada uno de estos rangos es tratado como un bin para determinar si la señal está siendo probada en todos los segmentos de su rango operativo.</p> <p>La cobertura de señales altas (feature_wb_signals_high) monitorea simultáneamente las señales wb_stb_i,</p>																																												

	wb_cyc_i, wb_we_i, y wb_ack_o para verificar el estado de escritura. Además, la cobertura cruzada (feature_wb_sel_i) evalúa combinaciones de rangos de wb_sel_i (0-3, 4-7, 8-11, 12-15) con todas las señales para la escritura en alto.
Observaciones:	Ninguna

13. Función #7

Función a verificar:	One chip-select signals																																																
Sección de la especificación:	2.4 SDRAM External connections																																																
Descripción:	<p>Verificar que la señal de selección de chip (sdr_cs_n) se activa y desactiva correctamente, permitiendo y bloqueando el acceso a la memoria SDRAM según corresponda. Esta señal es crítica para la operación del SDRAM, ya que controla cuándo la memoria está lista para recibir y ejecutar comandos.</p> <p>Table 1. Command Truth Table</p> <table><tr><th>Command</th><th>CS</th><th>RAS</th><th>CAS</th><th>WE</th><th>ADDR</th></tr><tr><td>Command inhibit (NOP)</td><td>H</td><td>X</td><td>X</td><td>X</td><td>X</td></tr><tr><td>No operation (NOP)</td><td>L</td><td>H</td><td>H</td><td>H</td><td>X</td></tr><tr><td>Active (select bank and activate row)</td><td>L</td><td>L</td><td>H</td><td>H</td><td>Bank/row</td></tr><tr><td>READ (select bank and column, and start READ burst)</td><td>L</td><td>H</td><td>L</td><td>H</td><td>Bank/column</td></tr><tr><td>WRITE (select bank and column, and start WRITE burst)</td><td>L</td><td>H</td><td>L</td><td>L</td><td>Bank/column</td></tr><tr><td>PRECHARGE (deactivate row in bank or banks)</td><td>L</td><td>L</td><td>H</td><td>L</td><td>X</td></tr><tr><td>AUTO REFRESH or SELF REFRESH (enter self refresh mode)</td><td>L</td><td>L</td><td>L</td><td>H</td><td>X</td></tr></table>	Command	CS	RAS	CAS	WE	ADDR	Command inhibit (NOP)	H	X	X	X	X	No operation (NOP)	L	H	H	H	X	Active (select bank and activate row)	L	L	H	H	Bank/row	READ (select bank and column, and start READ burst)	L	H	L	H	Bank/column	WRITE (select bank and column, and start WRITE burst)	L	H	L	L	Bank/column	PRECHARGE (deactivate row in bank or banks)	L	L	H	L	X	AUTO REFRESH or SELF REFRESH (enter self refresh mode)	L	L	L	H	X
Command	CS	RAS	CAS	WE	ADDR																																												
Command inhibit (NOP)	H	X	X	X	X																																												
No operation (NOP)	L	H	H	H	X																																												
Active (select bank and activate row)	L	L	H	H	Bank/row																																												
READ (select bank and column, and start READ burst)	L	H	L	H	Bank/column																																												
WRITE (select bank and column, and start WRITE burst)	L	H	L	L	Bank/column																																												
PRECHARGE (deactivate row in bank or banks)	L	L	H	L	X																																												
AUTO REFRESH or SELF REFRESH (enter self refresh mode)	L	L	L	H	X																																												
Prioridad:	Crítica																																																
Autor (Dueño de la función):	Luis Felipe Aguero Peralta																																																
Requerimientos del banco de pruebas:	<p>Un generador de reloj para simular las operaciones de tiempo del SDRAM.</p> <p>Una señal de reinicio para inicializar el SDRAM al comienzo de la simulación.</p> <p>Generar las señales de control necesarias (sdr_cs_n, sdr_ras_n, sdr_cas_n, sdr_we_n).</p> <p>Señales para manejar los datos (pad_sdr_din, sdr_dout) y las direcciones (sdr_addr).</p> <p>Incluir el módulo SDRAM que se va a verificar.</p>																																																
Pruebas:	<p>Prueba 1: Activar sdr_cs_n y enviar comandos de lectura/escritura al SDRAM, verificando que responde correctamente.</p> <p>Prueba 2: Desactivar sdr_cs_n y enviar comandos de lectura/escritura, verificando que el SDRAM no responde.</p>																																																
Criterio de chequeo:	<p>El SDRAM debe responder a los comandos solo cuando sdr_cs_n está activa (baja).</p> <p>El SDRAM no debe responder a ningún comando cuando sdr_cs_n está inactiva (alta).</p>																																																
Cobertura funcional:	<p>Verifica la interacción entre la señal de selección del chip sdr_cs_n y la señal de escritura wb_we_i. Se define un coverpoint para cada señal, con bins para sus estados alto y bajo. Además, una cobertura cruzada, cross_chip_select_signals, examina específicamente la condición donde sdr_cs_n está en bajo y wb_we_i en alto.</p>																																																
Observaciones:	Ninguna																																																

14. Función #8

Función a verificar:	Automatic controlled refresh
Sección de la especificación:	4
Descripción:	<p>El auto-refresh es una característica del controlador SDRAM que garantiza la integridad de los datos almacenados en la memoria. Esta función emite automáticamente comandos de refresh a intervalos regulares, sin necesidad de intervención del usuario.</p> 
Prioridad:	Crítica
Autor (Dueño de la función):	Ismael Jimenez Carballo
Requerimientos del banco de pruebas:	El banco de pruebas debe permitir diferentes configuraciones de refresco aleatorias para las señales <code>cfg_sdr_trp_d</code> (<code>tRP</code>), <code>cfg_sdr_trcar_d</code> (<code>tRC</code>) y <code>cfg_sdr_rfmax</code> (<code>tRFSH</code>), asegurando que el sistema se refresque adecuadamente en todos los casos.
Pruebas:	El banco de pruebas debe cambiar aleatoriamente el período de precharge usando la señal <code>cfg_sdr_trp_d</code> , configurándolo a valores cortos para pruebas más intensivas. Estos valores aleatorios deben estar limitados para evitar configuraciones que sean demasiado cortas o largas y no prácticas. También debe cambiar aleatoriamente el período de active a active / auto-refresh usando la señal <code>cfg_sdr_trcar_d</code> , asegurando que se prueben diferentes configuraciones de refresco. Además, debe cambiar aleatoriamente la cantidad de veces que se refresca usando la señal <code>cfg_sdr_rfmax</code> . Las restricciones en los valores de <code>cfg_sdr_trp_d</code> , <code>cfg_sdr_trcar_d</code> y <code>cfg_sdr_rfmax</code> deben garantizar que el sistema funcione dentro de los parámetros seguros y esperados.
Criterio de chequeo:	El criterio de chequeo es que el refresco ocurra correctamente en los ciclos de reloj proporcionados y que la cantidad de veces y el

	número de filas refrescadas coincidan con los parámetros configurados.
Cobertura funcional:	La cobertura funcional incluye probar diferentes configuraciones de períodos de precharge usando <code>cfg_sdr_trp_d</code> , verificar varias configuraciones de períodos de active a active / auto-refresh usando <code>cfg_sdr_trcar_d</code> , y probar la cantidad de filas en un mismo auto-refresh usando <code>cfg_sdr_rfmax</code> . Estas pruebas aseguran que todas las combinaciones posibles de estas configuraciones se evalúan para garantizar que el sistema se refresque adecuadamente en todas las situaciones.
Observaciones:	Ninguna