**Validação do Módulo I2C**

**1. Objetivo do Testbench**

O objetivo principal deste testbench é simular uma comunicação I2C, onde ele atua como um master I2C. O testbench simula o envio de 33 bytes de dados para o módulo slave I2C, sendo: 16 bytes para a palavra, 16 bytes para a chave e 1 byte para o comando de leitura/escrita (W/R). O módulo slave I2C deve receber esses 33 bytes corretamente, respondendo com um ACK (Acknowledge) ao master, conforme os padrões do protocolo I2C.

**2. Estrutura do Módulo**

O módulo i2c\_slave\_tb é um testbench Verilog responsável por simular o comportamento de um master I2C para validar o funcionamento de um módulo slave I2C (i2c\_slave).

**2.1. Entradas e Saídas do Testbench**

As entradas e saídas do testbench são as seguintes:

* clk: Sinal de clock principal (100 MHz).
* rst: Sinal de reset assíncrono.
* scl: Sinal de clock da linha I2C (5 MHz).
* sda\_in: Sinal de dados da linha I2C, usado para o master enviar dados.
* sda: Sinal bidirecional da linha de dados I2C, com alta impedância forçada em 1.
* start: Saída do slave que indica o início de uma comunicação I2C.
* bit\_done: Saída do slave que indica a conclusão da transmissão de um bit.
* data\_bit\_counter: Contador interno de 4 bits para os bits de dados.
* data\_out: Saída do slave que contém os dados recebidos (264 bits).
* data\_ready: Sinal de 10 bits do slave que indica a prontidão dos dados.

**2.2. Variáveis Internas**

O testbench utiliza as seguintes variáveis internas:

* master\_data: Registrador de 8 bits para os dados que o master envia ao slave.
* data\_128bits: Registrador de 264 bits que armazena o dado a ser enviado para o slave.
* slave\_address: Endereço do slave I2C (7 bits + bit R/W), inicializado como 8'b11010100.
* sda\_controle: Sinal para controlar a linha SDA, permitindo que o master libere o barramento.
* negedge\_detected: Variável inteira para detectar a borda de descida do sinal SCL.
* i, j: Variáveis inteiras de loop.

**2.3. Instanciação do Módulo Slave**

O testbench instancia o módulo i2c\_slave (Unidade Sob Teste - UUT) e conecta suas portas aos sinais correspondentes do testbench:

i2c\_slave uut (

.clk(clk),

.reset(rst),

.scl(scl),

.sda(sda),

.data\_out(data\_out),

.data\_ready(data\_ready),

.start(start),

.bit\_done(bit\_done)

);

**2.4. Controle da Linha SDA**

A linha sda é controlada de forma bidirecional. Quando sda\_controle é alto (1'bz), a linha está em alta impedância (liberada pelo master para o slave enviar ACK). Caso contrário, sda\_in define o valor da linha.

**2.5. Geração de Clocks**

* **SCL (I2C Clock)**: Gerado com uma frequência de 5 MHz (período de 200 ns).
* **CLK (Módulo Clock)**: Gerado com uma frequência de 100 MHz (período de 10 ns).

**3. Lógica de Validação**

A validação do módulo slave I2C segue os seguintes passos:

1. **Inicialização**:
   * rst é configurado como 1 (reset ativo).
   * clk é configurado como 1.
   * scl é configurado como 0.
   * data\_128bits é inicializado com um valor hexadecimal específico para ser enviado ao slave.
   * data\_bit\_counter é inicializado como 0.
   * sda\_controle é configurado como 0 (master controla o SDA).
   * sda\_in é configurado como 1 (SDA em nível alto).
   * Após 15 ns, o rst é liberado (0).
2. **Envio da Condição de Start**:
   * O testbench simula a condição de start I2C: sda\_in passa de 1 para 0 enquanto scl está em nível alto.
3. **Envio do Endereço do slave (Escrita)**:
   * O endereço do slave (slave\_address) é enviado bit a bit (7 bits + 1 bit de R/W = 0 para escrita).
   * O testbench aguarda a detecção da borda de descida de SCL (negedge\_detected) ou o módulo slave atingir o estado 2 antes de enviar o próximo bit.
   * Após o envio do endereço, o master libera o barramento (sda\_controle = 1) e aguarda o ACK do slave (estado 3 do UUT). Em seguida, o master retoma o controle do SDA (sda\_controle = 0).
4. **Envio dos Dados (264 bits)**:
   * Os dados de data\_128bits são enviados em blocos de 8 bits.
   * Para cada bloco de 8 bits (master\_data), o testbench aguarda o estado 3 do UUT.
   * Cada bit de master\_data é enviado individualmente, aguardando a borda de descida de SCL (negedge\_detected) antes de prosseguir para o próximo bit.
   * Após cada byte, o master libera o barramento (sda\_controle = 1) para o slave enviar o ACK e aguarda a borda de descida de SCL. Em seguida, o master retoma o controle do SDA (sda\_controle = 0).
5. **Envio da Condição de Stop e finalização**:
   * O testbench simula a condição de stop I2C: sda\_in passa de 0 para 1 enquanto scl está em nível alto.
   * O testbench aguarda a detecção da borda de descida de SCL e encerra a simulação.

**3.1. Detecção da Borda de Descida de SCL**

Um bloco always @(negedge scl) é utilizado para detectar a borda de descida do sinal SCL e configurar a variável negedge\_detected para 1. Esta variável é crucial para sincronizar o envio de bits e a espera por ACKs.

**4. Resultados Esperados**

Espera-se que o módulo slave I2C receba todos os 264 bits de dados enviados pelo master corretamente, respondendo com ACKs nos momentos apropriados, seguindo estritamente o protocolo I2C. A simulação deve ser concluída sem erros de protocolo, indicando que o módulo slave é capaz de processar uma comunicação de escrita I2C completa e receber múltiplos bytes de dados quando endereçado para ele.

**5. Conclusão**

Este testbench é uma ferramenta eficaz para verificar a funcionalidade do módulo slave I2C, simulando um cenário de comunicação de escrita completo com o envio de uma quantidade significativa de dados. A abordagem detalhada na simulação das condições de start, envio de endereço, envio de dados e condição de stop garante uma validação abrangente do comportamento do slave I2C.