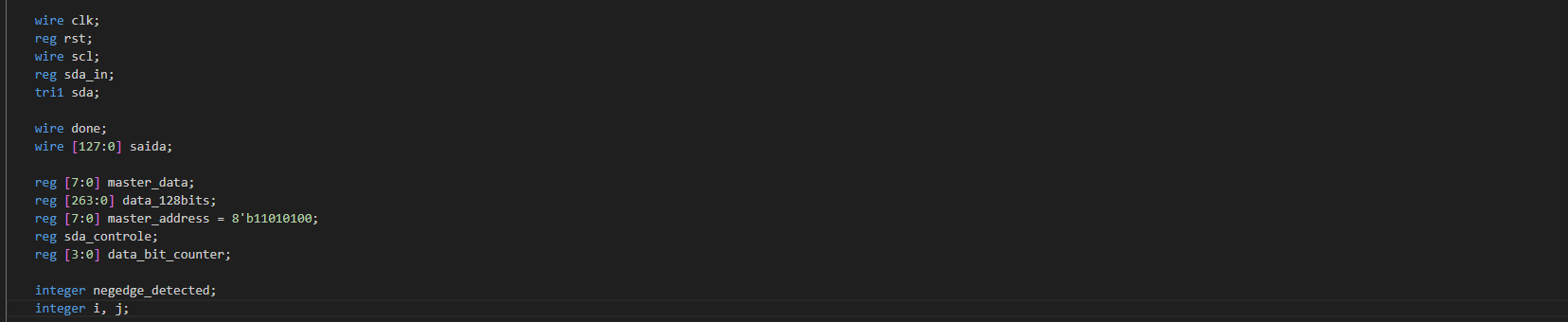
**Explicação Testbench módulo TOP**

A **testbench** para testar a criptografia e descriptografia AES-128 utilizando o protocolo de comunicação I2C. O objetivo principal é verificar se a implementação AES-128 (no módulo top\_i2c\_aes128) está correta, tanto para criptografar quanto para descriptografar dados.

**1. Estrutura do Testbench**

O testbench é composto por várias seções que incluem:

* **Declarações de Sinais:** São definidos sinais para clock, reset, comunicação I2C (SCL e SDA), e dados de entrada e saí
* **Instanciação do Módulo DUT (Design Under Test):** O módulo top\_i2c\_aes128 é instanciado, onde é realizado o processo de criptografia e descriptografia AES-128.
* **Clock Generators:** Gera os sinais de clock para a operação do sistema.
* **Tasks de Teste:** O testbench utiliza tasks para realizar os testes de criptografia e descriptografia, enviar os dados via I2C e verificar os resultados.

**2. Explicação dos Sinais e Parâmetros**

* **clk:** O clock principal do testbench. O gerador de clock (clockGenerator) cria um sinal de clock com o período configurado.
* **rst:** O sinal de reset que é utilizado para inicializar o sistema.
* **scl (Serial Clock Line):** Linha de clock do protocolo I2C.
* **sda\_in (Serial Data In):** Linha de dados de entrada para o protocolo I2C.
* **sda (Serial Data):** Linha de dados bidirecional do I2C. Está conectada ao sda\_in ou recebe um valor triestado controlado pelo sinal sda\_controle.
* **done:** Sinal de controle que indica quando a criptografia ou descriptografia foi completada no módulo DUT.
* **saida:** Contém o valor de saída gerado pelo módulo após o processo de criptografia ou descriptografia.

Além desses sinais, temos as variáveis:

* **master\_data:** Dados de 8 bits a serem enviados via I2C.
* **data\_128bits:** Dados de 128 bits que serão criptografados ou descriptografados, composto pela palavra e pela chave.
* **master\_address:** Endereço do mestre I2C.
* **data\_bit\_counter:** Contador para os bits de dados enviados.

**3. Geradores de Clock**

Dois geradores de clock são instanciados:

* clkGenerator: Para o sinal de clock principal.
* clkGeneratorSCL: Para o sinal de clock I2C (SCL). Ambos os geradores são configurados com diferentes períodos.

**4. Funcionalidade das Tasks**

**4.1. Task executarTeste**

A **task executarTeste** é responsável por realizar os testes de criptografia e descriptografia, verificando se os resultados obtidos pelo módulo AES-128 estão corretos em comparação com os resultados esperados.

* Definição dos Parâmetros da Task



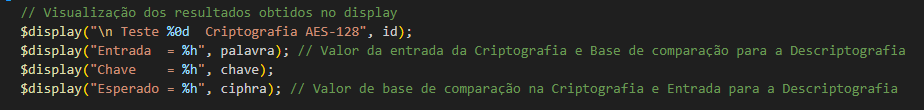
*id:* Identificador para o teste. Ele será usado para identificar qual teste está sendo executado (por exemplo, "Teste 0", "Teste 1", etc.).

*palavra:* Valor de entrada para a criptografia (a mensagem que será criptografada ou descriptografada).

*chave:* Chave de 128 bits que será usada tanto para a criptografia quanto para a descriptografia.

*ciphra:* Valor esperado de saída para a criptografia, que será comparado com o resultado obtido.

* **Visualização dos Resultados no Display**



*$display:* Comando para exibir informações no console de simulação. Aqui, ele exibe:

O número do teste (*ID*).

A entrada da criptografia (*palavra*).

A chave usada para a criptografia(*chave)*.

O valor esperado da criptografia (*ciphra*).

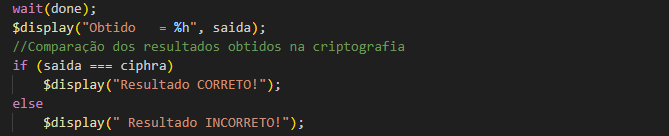
* **Configuração dos Dados e Envio para criptografia via I2C para o DUT**



A variável *data\_128bits* é configurada com a entrada e a chave. O valor *8'h00* é adicionado no final para indicar que o teste está no modo de criptografia (modo 0).

A task *envia\_via\_i2c()* é chamada para enviar os dados via protocolo I2C para o módulo AES-128 (DUT), que irá processá-los e retornar o resultado.

* **Processamento e Verificação da Criptografia**



*wait(done):* A simulação aguarda até que o sinal done seja acionado, indicando que o módulo AES-128 terminou o processamento (criptografia).

*$display("Obtido = %h", saida):* Exibe o valor obtido após a criptografia.

Verificação do resultado obtido: O valor obtido (saida) é comparado com o valor esperado (ciphra).

Se os valores forem iguais, significa que a criptografia foi realizada corretamente, e "Resultado CORRETO!" será exibido.

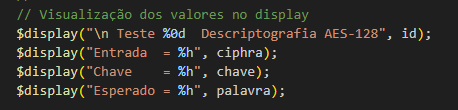
Se os valores forem diferentes, significa que houve um erro na criptografia, e "Resultado INCORRETO!" será exibido.

* **Reset e Atraso de Tempo para Preparação para o Próximo Teste**



*Reset:* Após cada teste, o sinal de reset (rst) é acionado por 15 unidades de tempo para reiniciar o módulo AES-128 antes de começar o próximo teste. Após isso, o reset é desativado e a simulação espera por 3000 unidades de tempo.

* **Visualização dos Resultados para Descriptografia**



Aqui, as informações sobre o teste de descriptografia são exibidas:

O número do teste.

O valor esperado de entrada para a descriptografia (ciphra).

A chave usada na descriptografia.

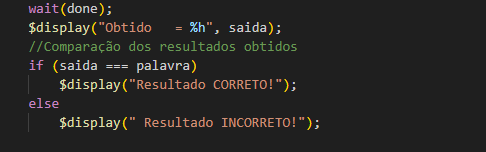
O valor esperado após a descriptografia (palavra).

* **Configuração dos Dados para Descriptografia e Envio via I2C**

A variável data\_128bits agora é configurada para descriptografar: a entrada será ciphra e a chave permanece a mesma. O valor 8'h01 é adicionado para indicar que o teste está no modo de descriptografia (modo 1).

A task envia\_via\_i2c() é chamada novamente para enviar os dados via I2C ao módulo AES-128 para que ele realize a descriptografia.

* **Aguardando a Conclusão do Processamento e Verificando o Resultado da Descriptografia**

Após a execução da descriptografia, o processo é o mesmo do teste de criptografia:

O testbench aguarda o sinal done.

O valor obtido (saida) é exibido.

A comparação é feita com o valor esperado para a descriptografia (palavra).

Se os valores coincidirem, o resultado é considerado correto, caso contrário, incorreto.

* **Resumo do Funcionamento**

Objetivo: A task executarTeste é usada para testar tanto a criptografia quanto a descriptografia de dados usando AES-128. A entrada é criptografada com uma chave e comparada com o valor esperado, e então, a criptografia resultante é usada como entrada para a descriptografia, com o objetivo de obter de volta os dados originais.

Execução do Teste: O teste é realizado em dois estágios:

1. Criptografia: Os dados são criptografados e o resultado é verificado.
2. Descriptografia: O valor criptografado é então descriptografado e o resultado é comparado com a entrada original.

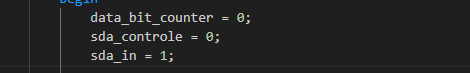
Controle de Fluxo: A task aguarda o processamento ser concluído usando o sinal done antes de verificar os resultados.

Essa task facilita a repetição dos testes com diferentes conjuntos de dados, automatizando a verificação da operação correta do algoritmo de criptografia AES-128.

**4.2. Task envia\_via\_i2c**

A **task envia\_via\_i2c** é responsável por enviar dados através do protocolo I2C para o módulo AES-128. O I2C é um protocolo de comunicação de dois fios, sendo eles SCL (Serial Clock Line) e SDA (Serial Data Line), usados para enviar e receber dados entre dispositivos.

* **Inicialização dos Sinais e Contadores**

*data\_bit\_counter:* Inicializa o contador de bits de dados com zero.

*sda\_controle:* Controla a linha SDA (dados) do I2C. Inicialmente, está configurado para 0, indicando que o dispositivo pode controlar a linha SDA (ou seja, o dispositivo está transmitindo dados).

*sda\_in:* Inicializa o sinal SDA com 1, ou seja, a linha SDA começa com o valor lógico 1 (alto).

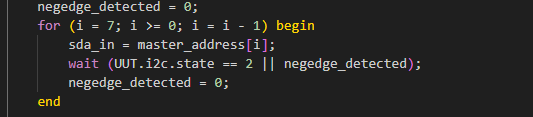
* **Primeiro Ciclo de Clock (Início da Comunicação I2C)**

*@(posedge scl):* Espera pela bordo de subida do sinal SCL (clock do I2C). Isso indica o início de um novo ciclo de clock.

*#20 sda\_in = 0;:* Após o ciclo de subida de SCL, o valor de SDA é definido como 0, indicando o início da transmissão de dados (geralmente, uma condição de start no protocolo I2C).

*@(negedge scl);:* Aguarda o bordo de descida do sinal SCL (fim de um ciclo de clock).

* **Envio do Endereço do Mestre (Master Address)**

*negedge\_detected = 0;:* Inicializa a variável de controle negedge\_detected (indica se o negedge (bordo de descida) de SCL foi detectado).

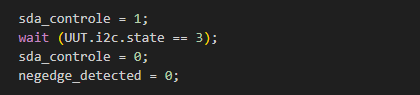
O laço for itera sobre os 8 bits do endereço do mestre (master\_address) e os transmite um por vez.

*sda\_in = master\_address[i];:* A cada iteração, o bit correspondente do endereço do mestre é transmitido pela linha SDA.

*wait (UUT.i2c.state == 2 || negedge\_detected);:* A task aguarda até que o estado do I2C (UUT.i2c.state) seja igual a 2 (presumivelmente indicando que o I2C está pronto para receber o próximo bit), ou que um bordo de descida de SCL (indicado por negedge\_detected) tenha ocorrido.

*negedge\_detected = 0;:* Reseta o sinal de detecção do negedge para a próxima iteração.

* **Troca de Controle de SDA (Configuração para Envio de Dados)**



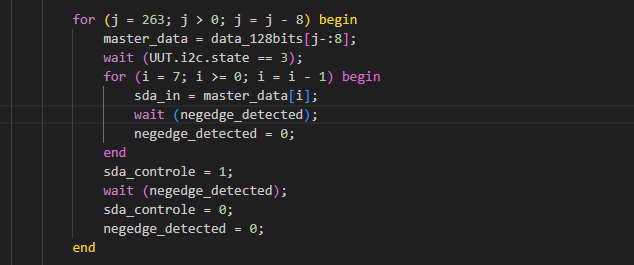
*sda\_controle = 1;:* O controle da linha SDA é transferido para o mestre, que começa a enviar os dados.

*wait (UUT.i2c.state == 3);:* Aguarda o estado do I2C ser igual a 3, que provavelmente indica que o dispositivo está pronto para a troca de dados.

*sda\_controle = 0;:* Após aguardar, o controle da linha SDA é liberado, permitindo que o próximo bit seja enviado.

*negedge\_detected = 0;:* Reseta a variável de controle de borda de descida para ser usada nas próximas iterações.

* **Envio de Dados (Data Transmission)**

 Este loop for itera sobre data\_128bits (que contém os dados a serem enviados) em blocos de 8 bits.

*master\_data = data\_128bits[j-:8];:* A cada iteração, 8 bits de dados são extraídos de data\_128bits.

*wait (UUT.i2c.state == 3);:* A task aguarda o estado do I2C ser igual a 3, indicando que o I2C está pronto para a troca de dados.

O laço interno (for i) envia os 8 bits de dados, um por vez:

*sda\_in = master\_data[i];:* Envia um bit de master\_data na linha SDA.

*wait (negedge\_detected);:* Espera o sinal SCL cair (bordo de descida) para enviar o próximo bit.

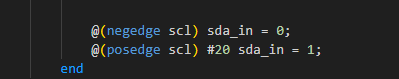
*negedge\_detected = 0;:* Reseta a variável de controle de borda de descida para a próxima iteração.

*sda\_controle = 1;:* Após o envio dos 8 bits, o controle da linha SDA é transferido para o mestre novamente.

*wait (negedge\_detected);:* Aguarda o bordo de descida de SCL.

*sda\_controle = 0;:* O controle de SDA é liberado, e o processo se repete.

* **Finalizando a Comunicação**



*@(negedge scl):* Espera o bordo de descida de SCL.

*sda\_in = 0;:* Coloca a linha SDA em 0 no bordo de descida de SCL, finalizando a comunicação (por exemplo, pode indicar um stop condition).

*@(posedge scl):* Espera o bordo de subida de SCL.

*#20 sda\_in = 1;:* Após um atraso de 20 unidades de tempo, coloca SDA em 1, encerrando a transmissão de dados.

* **Resumo do Funcionamento**

Objetivo: A task envia\_via\_i2c envia um conjunto de dados pela linha de dados SDA do protocolo I2C para o módulo DUT (no caso, o módulo AES-128).

Passos Principais:

Envia o endereço do mestre.

Envia os dados (128 bits) através do I2C, bit a bit, sincronizado com o clock SCL.

Após cada byte de dados, o controle da linha SDA é alternado.

A cada bit, o código aguarda o bordo de descida de SCL para sincronizar o envio de dados.

A task finaliza com uma operação de stop condition, indicando o fim da comunicação.

Essa estrutura de envio é essencial para a comunicação serial síncrona entre dispositivos utilizando o protocolo I2C, e a task envia\_via\_i2c gerencia todo esse processo dentro do testbench.

### 5. Valores escolhidos para os testes

### 5.1. Testes Baseados em FIPS-197

Teste 0 (Entrada 128'h00112233445566778899aabbccddeeff):

* + Entrada (Plaintext): 128'h00112233445566778899aabbccddeeff
  + Chave (Key): 128'h000102030405060708090a0b0c0d0e0f
  + Cifra (Ciphertext): 128'h69c4e0d86a7b0430d8cdb78070b4c55a

Esses valores são diretamente extraídos dos vetores de teste do FIPS-197 para AES-128. O vetor de teste com entrada 00112233445566778899aabbccddeeff e chave 000102030405060708090a0b0c0d0e0f gera o ciphertext esperado 69c4e0d86a7b0430d8cdb78070b4c55a.

Teste 1 (Entrada 128'h3243f6a8885a308d313198a2e0370734):

* + Entrada (Plaintext): 128'h3243f6a8885a308d313198a2e0370734
  + Chave (Key): 128'h2b7e151628aed2a6abf7158809cf4f3c
  + Cifra (Ciphertext): 128'h3925841d02dc09fbdc118597196a0b32

Esses valores também são parte dos vetores de teste do FIPS-197. O plaintext 3243f6a8885a308d313198a2e0370734 criptografado com a chave 2b7e151628aed2a6abf7158809cf4f3c resulta no ciphertext 3925841d02dc09fbdc118597196a0b32.

Teste 2 (Entrada 128'h00000000000000000000000000000000):

* + Entrada (Plaintext): 128'h00000000000000000000000000000000
  + Chave (Key): 128'h2b7e151628aed2a6abf7158809cf4f3c
  + Cifra (Ciphertext): 128'h66e94bd4ef8a2c3b884cfa59ca342b2e

A entrada nula (com todos os bits igual a zero) também é um caso de teste definido pelo FIPS-197. O resultado esperado para a criptografia de dados nulos com a chave especificada é 66e94bd4ef8a2c3b884cfa59ca342b2e.

Teste 3 (Entrada 128'hffffffffffffffffffffffffffffffff):

* + Entrada (Plaintext): 128'hffffffffffffffffffffffffffffffff
  + Chave (Key): 128'h2b7e151628aed2a6abf7158809cf4f3c
  + Cifra (Ciphertext): 128'h3f5b8cc9ea855a0afa7347d23e8d664e

Testes com dados máximos (todos os bits definidos como 1) também fazem parte dos testes típicos de verificação de segurança e funcionamento do algoritmo. O resultado da criptografia desses dados nulos com a chave fornecida gera o ciphertext 3f5b8cc9ea855a0afa7347d23e8d664e.

### 5.1. Teste Não Baseados em FIPS-197

Teste 4 (Entrada Arbitrária 128'haabbccddeeff00112233445566778899):

* + Entrada (Plaintext): 128'haabbccddeeff00112233445566778899
  + Chave (Key): 128'h102030405060708090a0b0c0d0e0f000
  + Cifra (Ciphertext): 128'h2be52b98821c28a467897944fa4ac1bc

Este teste não é baseado diretamente nos vetores de teste do FIPS-197, mas utiliza dados arbitrários para testar o AES-128. Esse tipo de teste é útil para verificar se o algoritmo lida corretamente com entradas variadas e não segue um padrão conhecido. Embora os valores esperados para esse teste sejam definidos, eles não fazem parte dos vetores oficiais fornecidos pelo FIPS-197.

4 dos 5 testes no código fornecido são baseados diretamente nos vetores de teste do FIPS-197 para o AES-128, que incluem casos de teste com entradas e chaves conhecidas, além dos resultados esperados para criptografia e descriptografia.

O 5º teste utiliza dados arbitrários e não segue os vetores de teste oficiais do FIPS-197, mas ainda assim serve para validar a implementação do AES-128 com entradas não padronizadas.