Tecnológico de Costa Rica

Diseño de sistemas digitales

Tarea 1: FIFO

Profesor

Ing. Juan José Montero

Alumno

Ronald Duarte Barrantes | 2021004089

Semestre 2, 2024

Código Testbench;

Demostración del funcionamiento del código del FIFO, en primeras instancias se tiene el agregado al testbench para hacer el llenado de los registros, en este caso se usó un repeat para poder parametrizarlo. También se muestra el código necesario para hacer las acciones de push y pop, estos también parametrizados.

```
initial begin
    Din  = 'b0;
    #20;
    repeat(DEPTH-1) begin
        #20;
        Din = $random;
    end
end

initial begin

    $display("running...");

    rst  = 1;
    push = 0;
    pop  = 0;
    #10;
    rst  = 0;
    #5;
    repeat ((DEPTH)) begin
        #10;
        push= 1;
        pop = 0;
        #10;
        push = 0;
    end
    #10;
    repeat ((DEPTH)) begin
        #10;
        push= 0;
        pop = 1;
        #10;
        pop = 0;
    end
```

Figura 1. Llenado de los registros del FIFO

Nota: Es importante no cambiar los valores de tiempo dentro de los repeat, ya que estos fueron puestos de tal manera que permite las acciones de push y pop, y que los flancos de los mismo no se activen en los mismos flancos del reloj, ya que esto genera bugs.

Resultados;

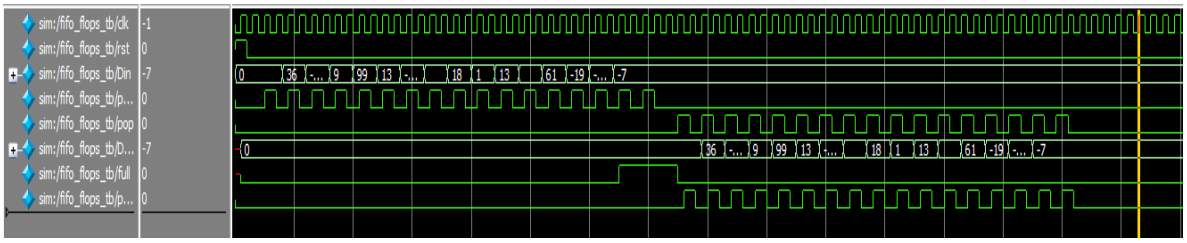

Figura 2. Ondas de las entradas y salidas del FIFO

En este caso de deben tomar en consideración las ondas vistas en el diagrama de tiempo, la primera de todos es;

Clk: Es la onda que permite habilitar algunas funciones de la estructura, esto es debido a la sensibilidad de los always con algún flanco del reloj, tiene un periodo de 10 ns. Tiene principal importancia en los contadores, los punteros y el pndng.

Rst: Dicha onda nos permite habilitar algunas de las ondas en valores conocidos cuando esta se encuentra en nivel alto, inicializa los valores del Dout, full, punteros, contadores y el flip flops.

Din: Son los valores de entra de “n” bits tomando en cuenta los “k” DEPTH utilizados.

Push: Onda que permite que el demux vaya guardando los valores en los registros, según el contador y el guardado lo hace por medio de flancos del push.

Pop: Onda que permite el vaciado del mux y generando la salida de los bits. Funciona mediante los flancos positivos de los pop.

Dout: son los valores de entrada que fueron guardados en los registros y expulsados a partir de los pop’s vistos anteriormente.

Full: Es una bandera que permite determinar que los registros ya están llenos, por medio del flanco positivo.

Pndng: Es otra bandera que nos permite saber si aun hay operaciones pendientes que realizar, si se hacen16 registros, el pndng será de 15, ya que después de este ya no habrá más operaciones al realizar.

Qué pasa si se hacen más push y pop, y el número de registro no varía;

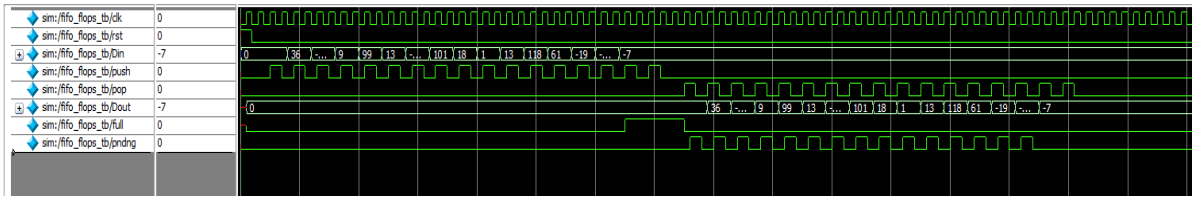

Figura 3. Exceso de operaciones en comparación con los registros

En este caso se hacen mas push y pop de la cantidad de registros que hay, el sistema detecta por medio de la bandera de full, cuando ya todos los registros fueron llenados, por lo cual el puntero se deja de mover entre los registros, haciendo que para el proceso del push, en el caso del pop, existe otra bandera dada por un cable interno la cual cuenta cuantas operaciones de push se hicieron y esta ayuda a que el puntero de salida deje de moverse entre los registros, dicha onda se puede ver en el siguiente diagrama de tiempos y se llama empty, se encuentra al final del diagrama de tiempo.

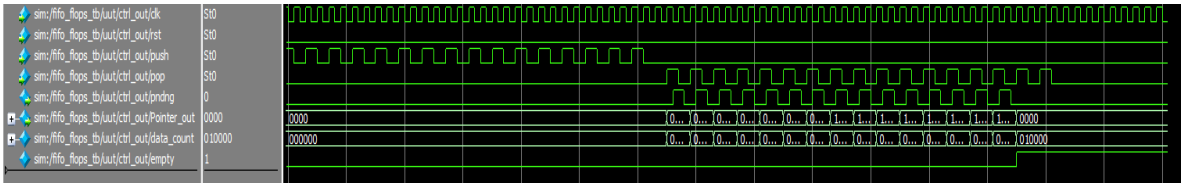

Figura 4. Bandera de empty

Push y Pop en flanco positivo al mismo tiempo;

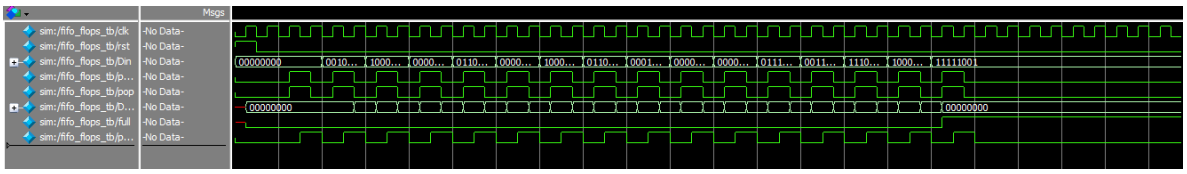

Figura 5. Push y pop al mismo tiempo

Cuando se hace push y pop al mismo los flacos pueden llegar a ser incompresible para la simulación, generando algunos bugs, en este caso se nota que el pop no influye en el full. Los valores que en teoría fueron guardados en los registros por medio del push, si son tomados por el pop, pero de manera que entre ellos se guarden 0 bits y después el valor que se espera que salga.

Posible solución;

Lo que podría ocasionar dicho error sería la puntera, ya que el en algunos casos este no se disminuye a pesar de que se hagan los pop, lo cual ocasiona que siga aumenta y apuntando a registros que no tengan datos aun, haciendo que aparezcan valores de bits de 0.

```
// write logic
always @(posedge clk) begin
    if(rst)       Pointer_in <= 'b0;
    else if(push & !full) Pointer_in <= Pointer_in + 'b1;
    else          Pointer_in <= Pointer_in;
end
```

Figura 6. Posible error

Sería bueno que el puntero disminuya su valor por cada Pop.

```
always @(posedge clk) begin
    if(rst)       Pointer_out <= 'b0;
    else if(pop & !empty) Pointer_out <= Pointer_out + 'b1;
    else          Pointer_out <= Pointer_out;
end
```

Figura 7. Posible error dos

Sería bueno que el puntero disminuya su valor por cada Push.

Video;

Video FIFO.mp4