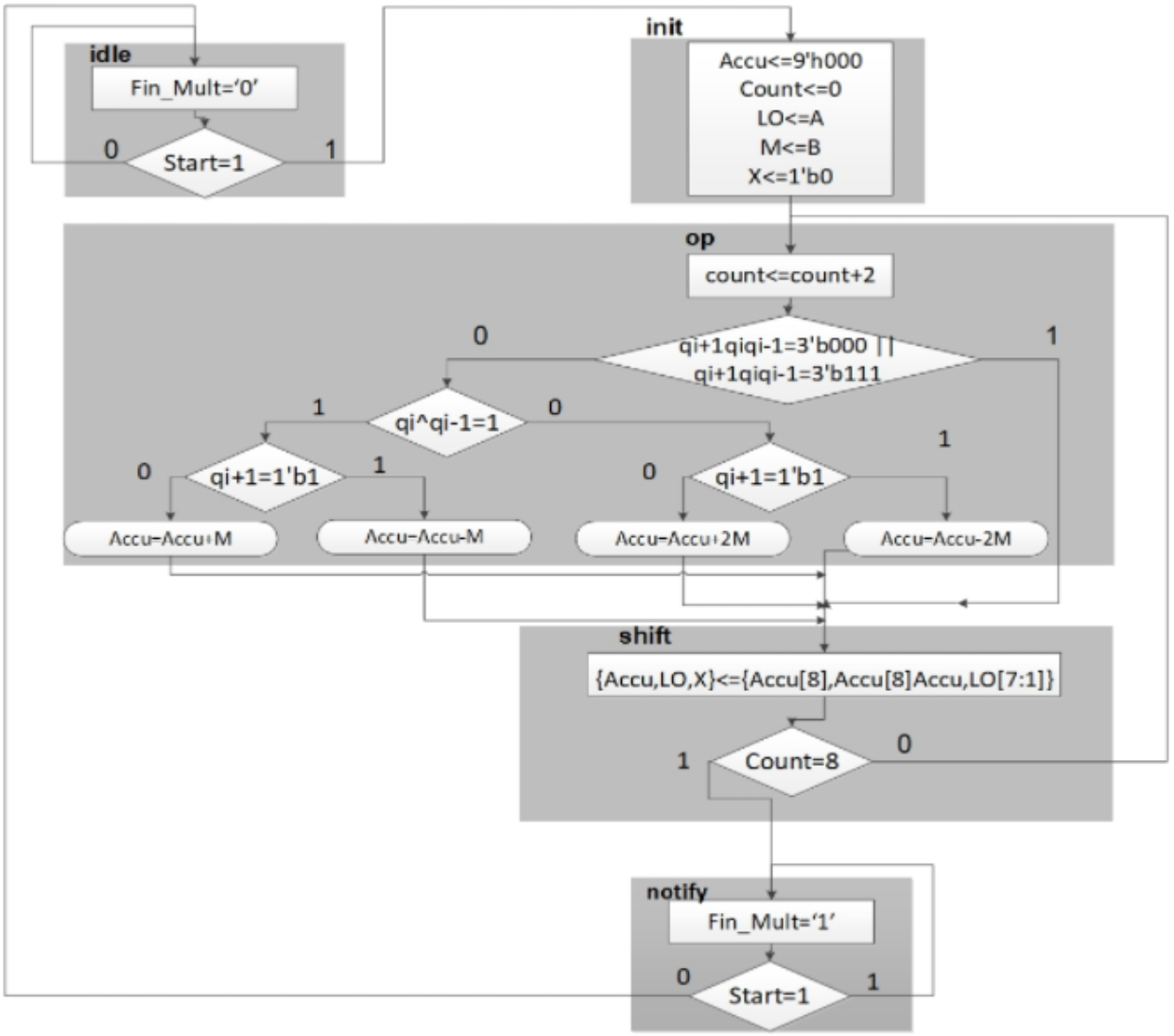
Tarea 2- Diseño y verificación de un multiplicador secuencial basado en sumas y desplazamientos



Por: Pasqual Moya, Iván Vivas, Javier Presmanes y Samuel García

Explicación del trabajo

En esta tarea llevaremos a cabo el diseño y la verificación de un multiplicador binario basado en el algoritmo de Booth modificado. Esto lo realizaremos en 3 partes principalmente.

Data-path

En primer lugar tenemos el data-path del propio multiplicador, este se encargará del control de los datos y resultados de las multiplicaciones como tal y se compone de lo siguiente :

-Un registro M donde almacenamos el multiplicando cuya entrada es B.

- Un registro X, cuya entrada son los bits qi+1 y qi enviados por SHIFTER_LO, donde almacenamos el bit qi-1 el cual enviamos como señal de estado al control-path y que utilizaremos para el comportamiento de la FSM.

-Un registro de desplazamiento “Shifter-LO”, cuya entrada en serie es la salida en serie de “SHIFTER_SHI” y su entrada en paralelo es la señal A, conteniendo el multiplicador y que a su misma vez enviará los bits qi y qi+1 al registro X previamente mencionado.

-Un registro de desplazamiento “Shifter-SHI” que actúa como el acumulador del sistema y cuya entrada en paralelo es la salida del sumador.

- Un sumador con entradas la salida en paralelo de “SHIFTER-SHI” y la salida del registro M.

Control-Path

En lo que respecta al control-path, este controla la generación de estímulos que activarán los diferentes componentes del data-path. Dicho control-path está formado por :

-Un contador parametrizable síncrono con cuenta ascendente, el cual particularizaremos para n=8 y que utilizaremos para el comportamiento de la FSM.

-Una FSM de 5 estados cuyo comportamiento explicaremos ahora:

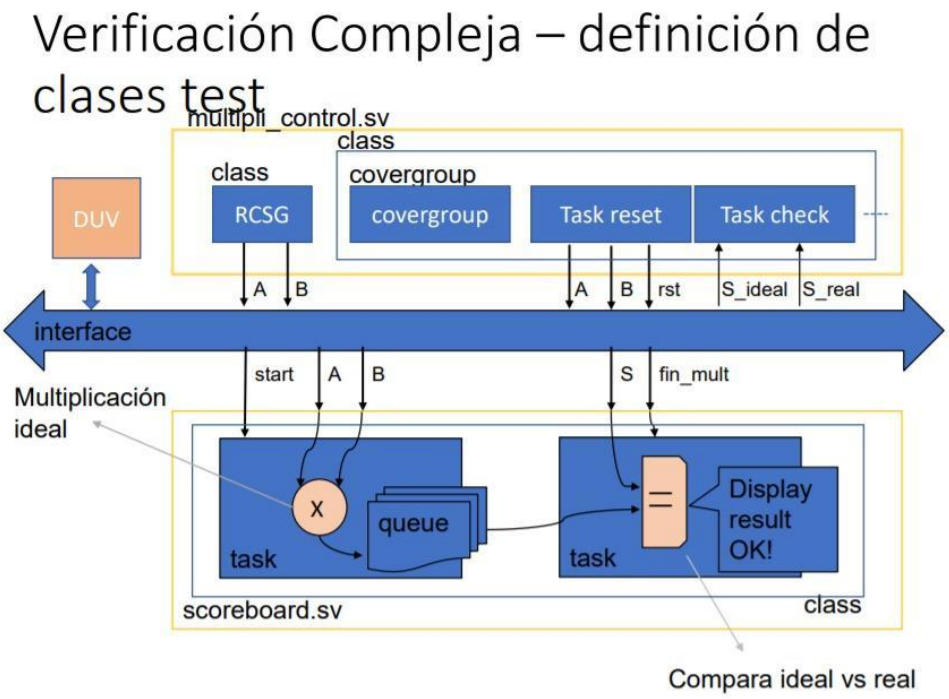
El primer estado, y default, es el estado IDLE. En él pondremos la señal de control Fin_Mult a 0 y permanecemos en este estado hasta que la señal “Start” se active, en este caso pasaremos al estado INIT.

El segundo estado, INIT, es en el cual comenzaremos con las operaciones, en este estado tan solo limpiaremos el “Shifter-SHI” e introduciremos las señales A y B a los registros LO y M respectivamente.

El tercer estado que encontramos es el estado OP, donde realizaremos las operaciones. El contador aumentará su cuenta en 2 y si la señal CONTROL (formada por los bits qi-1,qi y qi+1) está a todo unos o todo ceros pasaremos directamente al estado SHIFT. De no ser así realizaremos una serie de comprobaciones de dicha señal CONTROL y en base al resultado de dichas comparaciones realizaremos una operación sobre “SHIFTER-SHI”.

En el cuarto estado, SHIFT, asignamos como entrada de “SHIFTER-SHI” su último bit de salida, como entrada de “SHIFTER-LO” , la salida de “SHIFTER-SHI” y como entrada del registro X los bits [7:1] de “Shifter-LO”.

Verificación

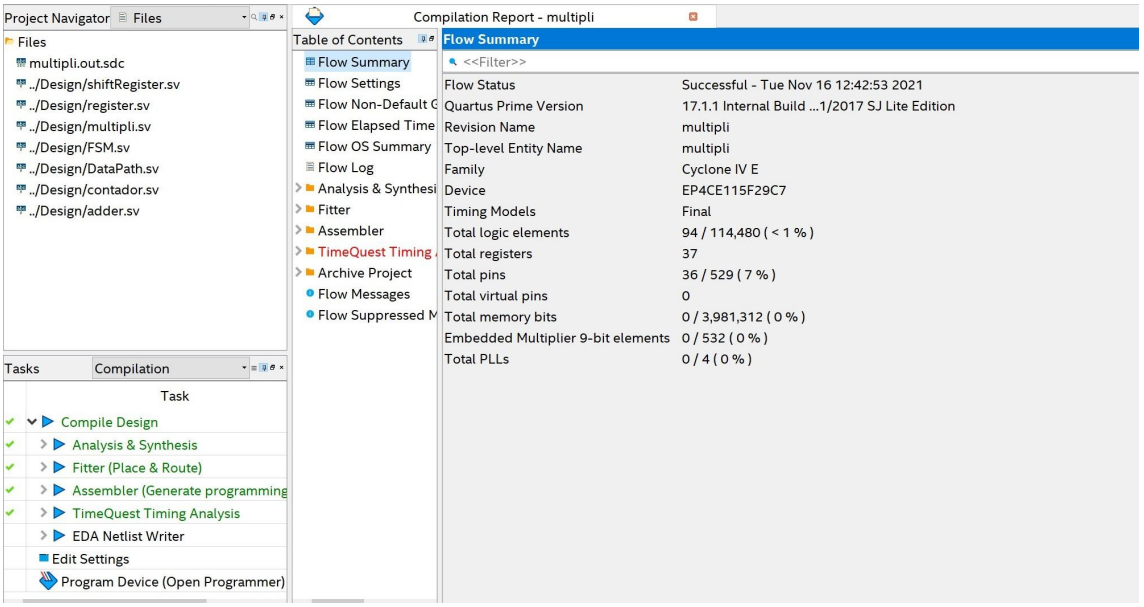


Para la parte de verificación hemos utilizado el modelo de verificación compleja que se nos mostraba en la tarea. Este se basa en 4 bloques principales, el DUV, el bloque encargado de generar estímulos, otro encargado de generar los estímulos que debería generar el DUV y comprueba que estos coincidan y un interfaz encargado de interconectar todos los bloques.

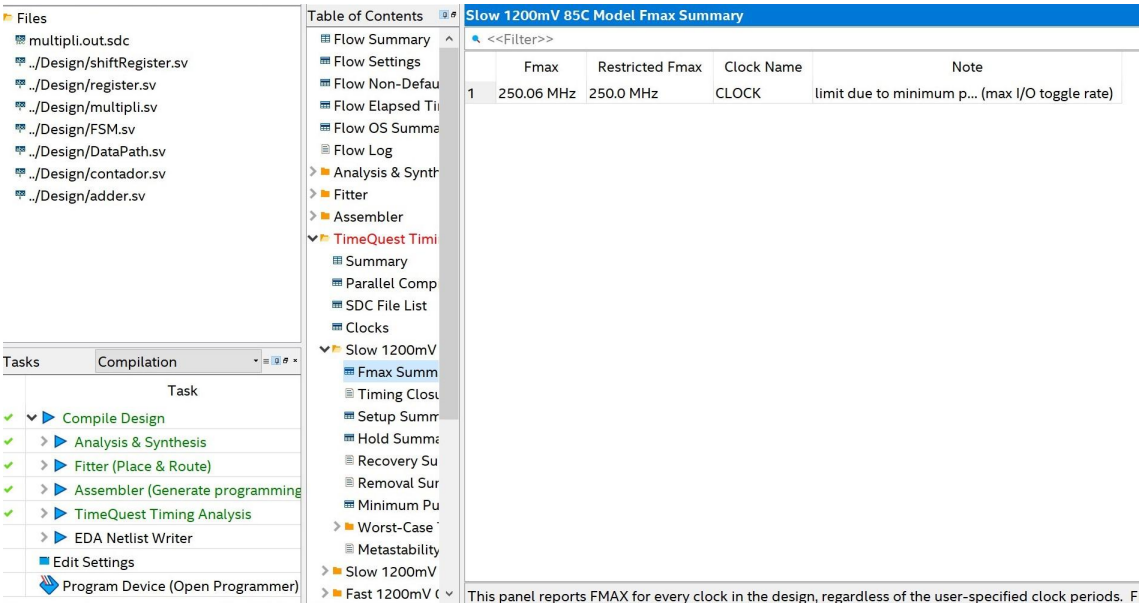
El bloque encargado de generar estímulos está hecho con bloques de tipo class, y algunas tasks. Para conectarnos al DUV hemos usado modports para pasarle las señales teniendo más control sobre qué y cómo le pasa.

El reloj lo controlamos principalmente mediante un clocking block para que sea más estable la señal original de reloj y para asegurarnos de que muestreamos e introducimos valores en los momentos correctos en todo momento.

Prestaciones del diseño



Como podemos observar en el resumen de compilación, el diseño cuenta con 94 logic elements y 37 flip flops.



Desde el Time Quest, tras ajustar algunos parámetros hemos logrado obtener una frecuencia máxima de 250,06 MHz, aunque la hemos restringido a 250MHz.