

UNIVERSIDAD NACIONAL DE SAN LUIS
FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS Y
NATURALES

Tema

*Pseudo Random Binary
Sequence con Vivado
HLS*

Leandro Marsó

Índice

1. Introducción	1
2. Implementación	1
3. Simulación	2
4. Optimizaciones	3
5. Conclusión	3

junio 2017

1. Introducción

En el presente informe, mostraremos los resultados de utilizar la herramienta de síntesis de alto nivel, aplicada a un código fuente en C. El circuito que queremos implementar es un generador de PRBS (Pseudo Random Binary Sequence). Implementamos una PRBS de 31.

2. Implementación

PRBS31 En este caso, el polinomio generador que usamos es el siguiente:

$$\text{PRBS7} = x^{31} + x^{28} + 1$$

Y el código fuente a sintetizar es el siguiente

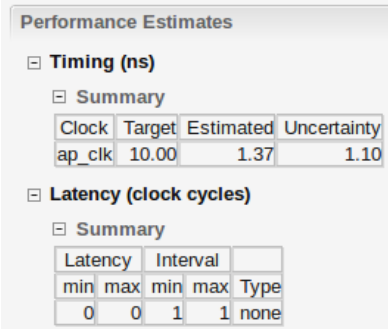
```
1 #include "prbs31.h"
2
3 void prbs31(result_t * hw_out) {
4     static data_t a = SEED;
5     int newbit = (((a >> 30) ^ (a >> 27)) & 1);
6     a = ((a << 1) | newbit) & 0x7fffffff;
7     *hw_out = a;
8 }
```

Cuyo archivo de definiciones es:

```
1 #ifndef _PRBS31_H
2 #define _PRBS31_H
3 #define SEED 0x02
4 typedef int data_t;
5 typedef int result_t;
6 void prbs31(result_t * out);
7 #endif
```

En el *testbench* del circuito hemos calculado por software la misma función y comparado con lo que devuelve la función a sintetizar. Además, declaramos una condición de finalización de la simulación, ya una PRBS una vez terminado el ciclo, vuelve a repetir todos los valores. Declaramos que pare en la iteración 1000 aproximadamente.

Resultado de la síntesis El primer aspecto a resaltar es la performance de este circuito. Vemos en la figura 1 que la latencia es un ciclo de clock, y se estima que el período del mismo puede ser $1,37ns$



The screenshot shows the 'Performance Estimates' window in Vivado. It has two main sections: 'Timing (ns)' and 'Latency (clock cycles)'. Both sections have a 'Summary' sub-section with a table.

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	1.37	1.10

Latency		Interval		Type
min	max	min	max	
0	0	1	1	none

Figura 1: Velocidad del clock y latencia

Sobre los recursos utilizados, el reporte de la figura 2 nos muestra 32 flip flops y una LUT (que implementa la semi-suma). Esto nos muestra que la implementación de la PRBS si hizo muy eficientemente, ya que si lo hubiésemos descrito en algún HDL, esperaríamos 31 FF y una XOR.

Sobre los puertos creados para el circuito, vemos en la figura 3 el clock, reset, y otras señales auxiliares, además del puerto *hw_out* de salida de nuestro circuito.

Resumimos en la tabla 1 los resultados de esta síntesis:

Utilization Estimates				
Summary				
Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	1
FIFO	-	-	-	-
Instance	-	-	-	-
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	32	-
Total	0	0	32	1
Available	280	220	106400	53200
Utilization (%)	0	0	~0	~0

Figura 2: Recursos utilizados

Register				
Name	FF	LUT	Bits	Const Bits
a	31	0	32	1
ap_CS_fsm	1	0	1	0
Total	32	0	33	1

Interface					
Summary					
RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	prbs31	return value
ap_rst	in	1	ap_ctrl_hs	prbs31	return value
ap_start	in	1	ap_ctrl_hs	prbs31	return value
ap_done	out	1	ap_ctrl_hs	prbs31	return value
ap_idle	out	1	ap_ctrl_hs	prbs31	return value
ap_ready	out	1	ap_ctrl_hs	prbs31	return value
hw_out	out	32	ap_vld	hw_out	pointer
hw_out_ap_vld	out	1	ap_vld	hw_out	pointer

Figura 3: Puertos y registros

Cuadro 1: Resumen

Puerto	Descripción
Estimated clock period	1.37ns
Worst case latency	1
Number of FFs used:	32
Number of LUTs used:	1

3. Simulación

Por último, hacemos una co-simulación con el RTL sintetizado para asegurarnos que todo está bien:

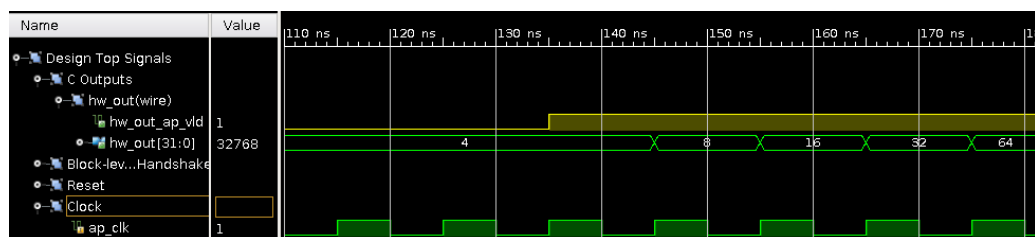


Figura 4: Simulación del RTL

4. Optimizaciones

Por el tipo de circuito elegido, encontramos que no hay optimización alguna que se pueda realizar en este nivel del flujo de diseño, ya que el resultado de la primera síntesis nos brinda un bloque que resuelve el cálculo en un ciclo de reloj.

5. Conclusión

Partiendo del código fuente en C del circuito, hemos utilizado la herramienta de síntesis HLS de Vivado, con un resultado óptimo en la primera corrida. También hemos implementado una PRBS7 (no incluida en el informe por tener resultados análogos) con mucha facilidad y reutilizando el testbench. Por lo que pudimos ver que este flujo de diseño nos permite el desarrollo rápido de hardware utilizando conceptos de programación imperativa.