SÍNTESIS LÓGICA AUTOMATIZADA PARA ESQUEMAS DE TEMPORIZACIÓN DE LATCHES ALTERNANTES

D. Guerrero, M. J. Bellido, J. Juan, P. Ruiz, A. Millán, E. Ostúa y J. Viejo
Grupo de Diseño Digital, Instituto de Microelectrónica de Sevilla, IMSE-CNM-CSIC
& Departamento de Tecnología Electrónica de la Universidad de Sevilla,
Avenida de Reina Mercedes s/n, 41012 Sevilla, Spain
{guerre,bellido,jjchico,pruiz,amillan,ostua,julian}@dte.us.es

ABSTRACT

Para sintetizar de forma automática circuitos digitales usando esquemas de temporización de latches alternantes, los autores proponen una técnica de codificación en VHDL. Esta técnica ha sido verificada por simulación lógica y mediante síntesis lógica en distintos procesos de fabricación CMOS.

1. INTRODUCCIÓN

Para evitar que el desplazamiento de reloj cause mal funcionamiento pueden utilizarse esquemas de temporización de múltiples fases de reloj como el esquema Master-Slave (MS) [1]. Una alternativa al esquema MS son los esquemas de temporización de latches alternantes (Parallel Alternating Latches Clocking Scheme, PALACS) [2]. Al igual que el esquema MS, PALACS proporciona tolerancia al desplazamiento untilizando múltiples señales de reloj, pero en lugar de usar como estructura básica dos latches conectados en cascada emplea latches con salida tri-estado conectados en paralelo. Esto permite leer y escribir simultáneamente el bloque de registros, lo que tiene notables ventajas en cuanto a velocidad y consumo de energía. Por otro lado, los lenguajes de descripción de hardware (Hardware Description Languages, HDL) [3] son herramientas muy convenientes para diseñar circuitos digitales. A menudo se usa software de síntesis lógica para producir descripciones a nivel lógico a partir de código HDL de alto nivel. Al codificar un diseño en un HDL el diseñador debe seguir un conjunto de reglas para asegurar que la descripción puede ser manejada de forma adecuada por la herramienta de síntesis lógica. En un sistema completamente síncrono, las técnicas de descripción usuales suelen dar como resultado flip-flops disparados por un solo flanco de la misma señal de reloj. Para usar PALACS en descripciones HDL, los elementos secuenciales deben codificarse de forma que su descripción pueda manejarse por las herramientas de síntesis lógica. En este artículo los autores describen técnicas de descripción en VHDL que permiten la síntesis automatizada de circuitos que empleen los esquemas PALACS de dos y cuatro fases.

2. TÉCNICAS DE CODIFICACIÓN EN VHDL PARA PALACS

El proceso para codificar un diseño que use PALACS puede dividirse en cuatro pasos: Descripción de un latch con salida tri-estado, descripción de la estructura PALACS, codificación de la parte combinacional e instanciación de las estructuras PALACS. El primer paso, mostrado en la figura 1, es común a cualquier diseño con PALACS y describe un latch cuya salida alimenta un buffer tri-estado, lo que constituye el bloque básico de construcción de la estructura PALACS. Las herramientas de síntesis lógica pueden implementarlo usando un latch dotado de señal de habilitación de salida o con un latch seguido de un buffer tri-estado dependiendo de los elementos disponible en la biblioteca estandar. En el segundo caso, que también es común a todos los diseños, dos instancias de la descripción anterior se usan para construir la estructura PALACS empleando la descripción VHDL estructural de la figura 2. La descripción estructural permite tener mejor control sobre el proceso de síntesis automática y evita que las herramientas de síntesis cambien la topología deseada. En los pasos tercero y cuarto se añaden instancias de la estructura PALACS al diseño para implementar la parte secuancial. A modo de ejemplo, en la figura 3 se describe un contador ascendente de cuatro bits. Esta descripción es similar al estilo de codificación canónico de máquinas de estado, salvo que los procesos que controlan la evolución al próximo estado han sido sustituidos por la instanciación de un conjunto de celdas PALACS usando una instrucción 'generate'. Nótese que tanto el PALACS de dos fases como el de cuatro fases pueden implementarse de este modo, ya el el esquema PALACS de dos fases es un caso particular del PALACS de cuatro fases en el que las señales de habilitación de salida coinciden con las señales de control de carga.

3. VERIFICACIÓN DEL ESTILO DE DESCRIPCIÓN VHDL PARA PALACS

Para verificar esta técnica de codificación se ha usado la descripción del contador de cuatro bits de la figura 3. Se ha usado el esquema PALACS de dos fases. Para chequear la funcionalidad el circuito ha sido simulado a nivel lógico. Además, para comprobar que el código es sintetizable en cualquier tecnología se ha compilado usando la herramienta Design Analyser de Synopsys (Synopsys Inc. http://www.synopsys.com/). Los componentes fueron sintetizados con éxito en un proceso CMOS de 0.35 μm de AMS (Austria Micro Systems, http://www.austriamicrosystems.com/), así como en procesos de 0.18 μm y 0.13 μm de UMC (United Microelectronics Corporation,

http://www.umc.com/) dando resultados similares. Se añadieron cuatro estructuras PALACS como se indica en la descripción VHDL de la figura 3, y se sintetizó lógica adicional de forma automática para conseguir la funcionalidad deseada.

4. CONCLUSIONES

En trabajos previos se introdujeron los esquemas PALACS para solventar problemas relacionados con el desplazamiento de reloj. En este trabajo se explora una técnica de codificación en VHDL que permite sintetizar automáticamente circuitos lógicos que usen PALACS. Esta técnica puede aplicarse fácilmente simplemente redefiniendo el bloque de registros de un diseño. Un diseño VHDL de ejemplo se ha implementado con éxito en tres procesos CMOS distintos de dos foundries distintas, lo que muestra que las descripciones VHDL propuestas son completamente sintetizables y producen las estructuras y comportamiento correctos. Esto se ha comprobado usando simulación lógica. Esto permite una aplicación mas fácil y amplia de PALACS a diseños digitales en general.

5. REFERENCIAS

[1] D. Harris, *Skew-Tolerant Circuit Design*, Morgan Kaufmann Publishers, San Francisco, USA, 2001.

[2] D. Guerrero et al. "Automated performance evaluation of skew-tolerant clocking schemes", *International Journal of Electronics*, Taylor & Francis, Andover, UK, pp. 819-842, 2006.

[3] T. Pollán, *Electrónica Digital*, Prensas Universitarias de Zaragoza, Zaragoza, Spain, 2004.


```
entity latchOEclr is
 generic( n: integer:= 1);
 port (d,noe, ld, nclr: in std_logic;
  q: out std_logic);
end latchOEclr;
architecture behaviour of latchOEclr is
  signal qi: std_logic;
begin
 asign: process(ld, d, nclr)
 begin
  if ld='1' then
    qi<=d;
  end if;
  if nclr='0' then
   qi<='0';
  end if;
 end process;
 myoutput: process(qi,noe)
 begin
  q<='Z';
  if noe='0' then
   q<=qi;
  end if;
 end process;
end behaviour;
```

Figura 1: Descripción de un latch con salida tri-estado

```
entity palacs4 is
 port (d, noe0, noe1, clk0,
 clk1, nclr : in std_logic;
 q : out std_logic);
end palacs4;
architecture mystruct of palacs4 is
 component latchOEclr
  port (d, noe, ld, nclr : in
std_logic;
     q : out std_logic);
 end component;
begin
 latch0 : latchOEclr port map (d=>d,
      noe=>noe0,ld=>clk0,
      nclr=>nclr,q=>q);
 latch1 : latchOEclr port map (d=>d,
      noe=>noe1,ld=>clk1,
      nclr=>nclr,q=>q);
end mystruct;
```

Figura 2: Descripción de la estructura PALACS

```
entity cntMod16 is
 port (noe0,noe1,clk0,
 clk1,nclr: in std_logic;
 myoutput: out
 std_logic_vector (3 downto 0));
end cntMod16;
architecture mystruc of cntMod16 is
 signal std_cnt,nxt_std:
  unsigned (3 downto 0);
 component palacs4
  port (d,noe0,noe1,clk0,clk1
  ,nclr: in std_logic;q:out std_logic);
 end component;
begin
 my_logic: process(std_cnt)
 begin
  myoutput<=std_logic_vector(std_cnt);
  nxt_std<=std_cnt+1;
 end process;
 generate_registers: for i in 3 downto 0
  generate
   digito : palacs4 port map
        (q=>std_cnt(i),
        noe0=>noe0,
        noe1=>noe1,
        clk0=>clk0,
        clk1=>clk1,
        nclr=>nclr,
        d=>nxt_std(i));
 end generate generate_registers;
end mystruc;
```

Figura 3: Descripción de un contador de cuatro bits