Proceedings of the 5th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Madrid, Spain, February 15-17, 2006 (pp74-77)

The 5-Phase Block-Encoded Frame Partitioning Circuit

Dr. SERAFIM PORIAZIS
Phasetronic Laboratories
6 Depasta Str., 17122, N. Smirni, Athens
GREECE
http://www.phasetroniclab.com

Abstract: The behavior of the 5-Phase Block-Encoded Frame Partitioning (EFP5) circuit is analyzed. The circuit performs the partitioning of the input clock signal into one out of fifty-two available partitions for a given five-phase frame. Each partition is being selectable by a 6-bit control word. The output signals of the circuit present the blocks of the selected partition in a phase encoded format which utilizes the given timing of the phases of the input frame. A phase difference equal to the half period of the clock signal is used internally to achieve the correct pulse timing at the output. The VHDL description of the EFP5 cell is given and the simulation and synthesis results are presented.

Key words: block, clock, circuit design, frame, partitioning, phase, VHDL.

1 Introduction

Automatic test-pattern generation (ATPG) and design for testability (DFT) techniques are based on clock partitioning and selective clock freezing [1] in order to break the global feedback loops and to generate clock waves to test a sequential circuit with self-loops. Clock partitioning increases the testability of the sequential circuit. Algorithms can be run on the circuit with partitioned clocks to identify combinationally and sequentially untestable faults. A model with two time-frames is utilized, where a vector is generated in the first one and then the changed values of internal FFs are propagated to the QFI inputs of the next time-frame before a new fault is targeted; these values are frozen for ATPG.

Considering the Power Reduction aspect of FSM design, this can be achieved through clock gating and disabling the primary inputs to the sub-FSMs not active [2]. Clock-gating is an effective approach to reduce power consumption of Finite State Machines (FSMs) which have plenty self-loop events [3]. A low-power asynchronous communication controller is proposed by [4] for the interaction between the sub-FSMs. These are controlling the gating of the global clock. They utilize a circuit called Clock Controller Block (CCB) and the decomposed FSM consists of: (1) a number of sub-FSMs (partitions), (2) an equally large number of asynchronous CCBs, (3) nand-gates for gating the local clocks, and (4) one inverter for the global clock signal. The number of CCBs is equal to the number of partitions.

Instead of adapting techniques of gating the clock, we consider in this paper a dedicated circuit that can produce sets of periodic patterns of clock phases under the control of an external binary word. In particular, we analyze the behavior of a 5-Phase Block-Encoded Frame Partitioning (EFP5) circuit, which generates specific patterns of clock pulses for each of the available fifty-two clock phase partitions for a time-frame of length 5, under the control of a 6-bit word. The blocks of the selected partition are presented at the five output signals in an encoded format by assigning all elements of a block to a phase of the given input frame. The subject circuit represents a reliable solution to the challenging problem of synchronizing the individual modules of a multiphase model [5], whose operation adopts a 5-phase timing pattern. The present work is based on the principles of operation of the Two-Phase Twisted Ring Counter (2P-TRC) circuit [6] and is targeting data streaming applications. The unit phase duration that is used at the outputs is equal to the period of the input clock signal, while internally the half period of the clock is used for the generation of a set of phased signals. The VHDL description of the circuit is given. The simulation of the EFP5 and the synthesis results are presented.

2 The EFP5 Cell

A. The basic cell operation

The fundamental cell, which partitions the frame of length 5 of the clock signal CLK of frequency *f*, is called the 5-Phase Block-Encoded Frame Partitioning (EFP5) circuit. For a set of five phases we have fifty-two partitions and each one can be selected by a 6-bit control word. The content of each block of a partition, for the given frame of the clock, is encoded by using the mathematical Restricted Growth String notation, e.g. for the binary control word “100100”=36 we have a three block partition with blocks B_0=(phase-1, phase-4), B_1=(phase-2, phase-5) and B_2=(phase-3) where the first block holds the phase-1 and phase-4 of the frame, the second block holds the phase-2 and phase-5 of the frame, and the third block holds the phase-3 of the frame of the input clock signal. The above partition is encoded by the Restricted Growth String notation as the sequence {0,1,2,0,1} where the element 0 of this sequence signifies block B_0, the element 1 block B_1 and the element 2 block B_2. The above example is shown in Table 1. Similarly holds for each of the remaining partitions. All encoded partitions are sorted in ascending order in order to define an index for each one from 0 to 51. This encoded format of a partition is presented at the output of the EFP5 circuit.

```
library IEEE;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_1164.all;

entity EFP5 is
    port (CLK , RESET : in std_logic;
          CTL : in std_logic_vector(5 downto 0);
          RSTFLAG : out std_logic;
          FPCLK : out std_logic_vector(4 downto 0) );
end EFP5;

architecture behavioral of EFP5 is
      type RG_string is array(0 to 4) of integer;
      type validpartitions is array(0 to 51) of RG_string;
      type validcodewords is array(1 to 20) of std_logic_vector(10 downto 1);
      constant phased_output : validcodewords :=
      (  ('0','0','0','0','0','0','0','0','0','0'), ('0','0','0','0','0','0','0','0','0','1'),
         ('0','0','0','0','0','0','0','0','1','1'), ('0','0','0','0','0','0','0','0','1','1','1'),
         ('0','0','0','0','0','0','0','1','1','1'), ('0','0','0','0','0','0','0','1','1','1','1'),
         ('0','0','0','0','1','1','1','1','1','1'), ('0','0','0','1','1','1','1','1','1','1'),
         ('0','0','1','1','1','1','1','1','1','1'), ('0','1','1','1','1','1','1','1','1','1'),
         ('1','1','1','1','1','1','1','1','1','1'), ('1','1','1','1','1','1','1','1','1','0'),
         ('1','1','1','1','1','1','1','1','0','0'), ('1','1','1','1','1','1','1','1','0','0'),
         ('1','1','1','1','1','1','0','0','0','0'), ('1','1','1','1','1','0','0','0','0','0'),
         ('1','1','1','1','0','0','0','0','0','0'), ('1','1','1','1','0','0','0','0','0','0'),
         ('1','1','0','0','0','0','0','0','0','0'), ('1','0','0','0','0','0','0','0','0','0') );
       constant partition : validpartitions :=
       ( (0,0,0,0,0), (0,0,0,0,1), (0,0,0,1,0), (0,0,0,1,1), (0,0,0,1,2), (0,0,1,0,0),
         (0,0,1,0,1), (0,0,1,0,2), (0,0,1,1,0), (0,0,1,1,1), (0,0,1,1,2), (0,0,1,2,0),
         (0,0,1,2,1), (0,0,1,2,2), (0,0,1,2,3), (0,1,0,0,0), (0,1,0,0,1), (0,1,0,0,2),
         (0,1,0,1,0), (0,1,0,1,1), (0,1,0,1,2), (0,1,0,2,0), (0,1,0,2,1), (0,1,0,2,2),
         (0,1,0,2,3), (0,1,1,0,0), (0,1,1,0,1), (0,1,1,0,2), (0,1,1,1,0), (0,1,1,1,1),
         (0,1,1,1,2), (0,1,1,2,0), (0,1,1,2,1), (0,1,1,2,2), (0,1,1,2,3), (0,1,2,0,0),
         (0,1,2,0,1), (0,1,2,0,2), (0,1,2,0,3), (0,1,2,1,0), (0,1,2,1,1), (0,1,2,1,2),
         (0,1,2,1,3), (0,1,2,2,0), (0,1,2,2,1), (0,1,2,2,2), (0,1,2,2,3), (0,1,2,3,0),
         (0,1,2,3,1), (0,1,2,3,2), (0,1,2,3,3), (0,1,2,3,4) );
        signal index : integer := 0;
        signal present_state1, present_state2, next_state, pclk: std_logic_vector(10 downto 1);
        signal invalidcode_flag : std_logic := '0';
      begin
       reg1 : process (CLK, RESET)
          begin
             if  RESET = '1' then  present_state1 <= phased_output(1);
             elsif (CLK='1' and CLK'event )  then present_state1 <= next_state ;
             end if;
       end process;
       reg2 : process (CLK, RESET)
          begin
             if  RESET = '1' then  present_state2 <= phased_output(11);
             elsif (CLK='0' and CLK'event )  then present_state2 <= next_state ;
             end if;
       end process;
       next_state_logic : process (CLK, RESET)
          begin
             case CLK is
              when '1' =>  if RESET = '1' then index <= 1;  else index <= index + 1; end if;
              when '0' =>  if RESET = '1' then index <= 11; else index <= index + 1; end if;
              when others   => null;
             end case;
             if index < 20 then next_state <= phased_output(index + 1);
             else next_state <=  phased_output(1); index <= 1;  end if;
             for i in 1 to 20 loop
               if next_state = phased_output(i) then invalidcode_flag <= '0'; exit;
               else invalidcode_flag <= '1'; end if;
             end loop;
        end process;
        output_logic : process (index, present_state1, present_state2)
          variable ptn : RG_string;
          variable pindex : integer;
          begin
            pindex := CONV_INTEGER(CTL);
            if pindex >= 0 and pindex <= 51 then ptn := partition(pindex);
            else ptn := partition(0); end if;
            case CLK is
              when '1' =>  pclk <= present_state1;
                           if RESET = '1' then RSTFLAG <= '1'; else
                           RSTFLAG <= invalidcode_flag;  end if;
              when '0' =>  pclk <= present_state2;
                           if RESET = '1' then RSTFLAG <= '1'; else
                           RSTFLAG <= invalidcode_flag;  end if;
              when others   => null;
            end case;
            for n in 0 to 4 loop
             case ptn(n) is
              when 0 => FPCLK(n) <= pclk(1) xor pclk(2);
              when 1 => FPCLK(n) <= pclk(3) xor pclk(4);
              when 2 => FPCLK(n) <= pclk(5) xor pclk(6);
              when 3 => FPCLK(n) <= pclk(7) xor pclk(8);
              when 4 => FPCLK(n) <= pclk(9) xor pclk(10);
              when others => FPCLK(n) <= '0';
             end case;
            end loop;
           end process;

end behavioral;
```

Figure 1. The VHDL description of the EFP5 cell

We note that each phase of the input clock signal has a duration equal to the period T of the clock with a duty cycle of 50 percent, thus having a pulse of logic-1 or logic-0 value for a half period.

Table 1. The block-encoding of the partitioning operation for the control word "100100"

index	Binary Control Word	Encoded Partition	Block index
36	100100	{0,1,2,0,1}	0,1,2

B. The algorithm aspects of cell operation

The VHDL description of the EFP5 cell is given in Figure 1. The entity section has an input port CLK on which the clock signal of frequency *f* is applied, an input port RESET on which a reset flag is applied, and an input port CTL[5..0] on which a 6-bit control word is applied. The output signal FPCLK[4..0] carry the phase information of the partition of the frame of signal CLK Each output signal can carry phase information only under the control of input CTL and only for the valid fifty-two values, that is, during a frame we have the selected partition present in the block-encoded format at the output lines of FPCLK. The above ports are shown in Figure 2 on the block diagram of EFP5.

The architecture section is of type "behavioral" and utilizes a state machine model, where two internal registers are being used, reg1 and reg2, one for the present state named "present_state1" clocked by the rising edge of the clock and the other for the present state named "present_state2" clocked by the falling edge of the clock, respectively. The next state logic block and the output logic block of the model are specified by the corresponding processes "next_state_logic" and "output_logic". The internal set of twenty valid codewords of the circuit is stored in an indexed array of size 20*10=200 bits that is represented by the constant named "phased_output". The index of the above array cycles through the integer values 1 to 20 specifying the valid codeword entry for the next state signal. The valid fifty-two partitions of the 5-phase input frame are stored in the indexed array constant "partition" of length 52 and each element of it is given by an array of integer values that signify the blocks of the partition by utilizing the Restricted Growth String notation. The partition information at each output line of FPCLK[4..0] is derived from the internal phased signals for each value of the 6-bit control word CTL by applying the EXOR operator on the signal lines of *pclk*[10..1].

C. The valid codeword sequence

Internally, the eight overlapping phased signals $pclk_1$, $pclk_2$, $pclk_3$, … , $pclk_{10}$ have frequency equal to $f/10$ (period of each $pclk_i$, i=1,2,3,…,10 equals 10·T, where T the period of CLK) with signal $pclk_1$ leading $pclk_2$, $pclk_2$ leading $pclk_3$, … , $pclk_9$ leading $pclk_{10}$ by a T/2 time difference. The logic-'1' or logic-'0' pulse width of each of the above phased signals is equal to 10·T/2. Consecutive changes of logic value at each signal $pclk_i$, for i=1,2,3,…,10 occur at the rising or falling edges of CLK at a distance of 10·T/2.

When the clock signal CLK is applied to the circuit, the following cyclic sequence of codewords is presented at the internal phased signals: $pclk_1$,$pclk_2$,$pclk_3$, … , $pclk_{10}$= 0000000000→1000000000→1100000000→1110000000→ 1111000000→1111100000→1111110000→1111111000→ 1111111100→1111111110→1111111111→0111111111→ 0011111111→0001111111→0000111111→0000011111→ 0000001111→0000000111→0000000011→0000000001, which is considered as being the normal internal cell operation. Each codeword remains stable for the state time of the circuit, that is T/2, and the above sequence is repeated internally throughout the operation of the cell. Thus the cycle time for the *pclk* pattern is defined by the twenty-tuple of codewords of length equal to 10·T. This duration forms the period of each internal phased signal.

The additional 1004 codewords out of the total 1024 possible codewords that are not included in the above internal cyclic sequence should be considered during the design of the circuit for achieving reliable operation of the EFP5 cell. If the circuit reaches any of these 1004 invalid codewords, then an invalid codeword flag is set. This flag maintains the proper initializing behavior until a valid codeword appears internally in the cell.

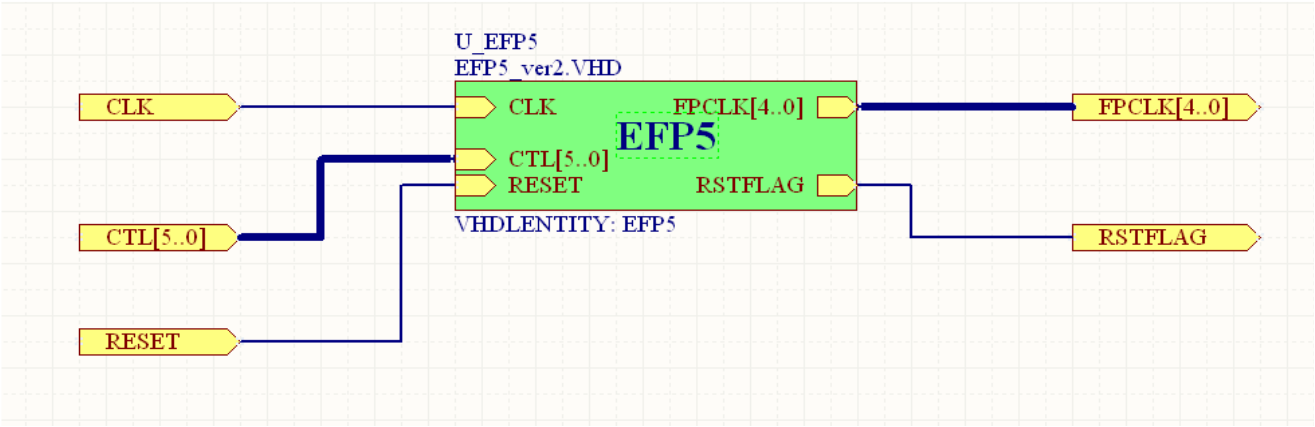

Figure 2. The EFP5 block diagram

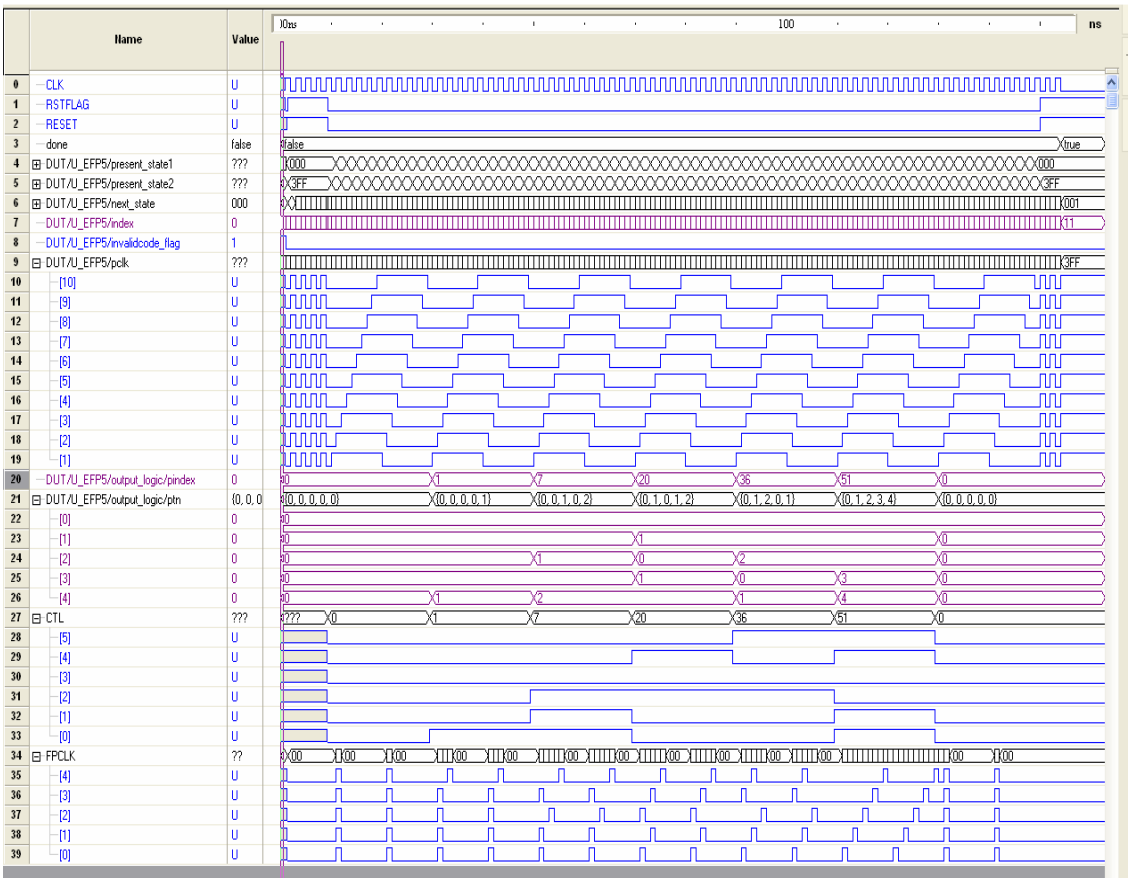

Figure 3. The EFP5 cell operation (VHDL simulation results)

D. The VHDL simulation and synthesis

The VHDL testbench simulation results for the EFP5 cell are given in Figure 3. The duration of this simulation is defined by the value of the signal "done". The signal *pclk* has a width of 10 bits, and CTL and FPCLK of 6 and 5 bits respectively. The output port FPCLK is analyzed into five individual output signals with waveforms that verify the correct operation of the circuit (for demonstration purposes only the control words "000000"=0, "000001"=1, "000111"=7, "010100"=20, "100100"=36 and "110011"=51 are shown, each for a duration of two frames). The logic value changes of the internal phased signals *pclk* occur at each rising and at each falling edge of the input signal CLK.

The synthesis of the EFP5 cell targeting an FPGA device was successfully performed giving us the following results:

- flip flops with asynchronous reset = 10
- flip flops with asynchronous preset = 10
- combinational feedback paths = 32
- combinational logic area estimate = 229 LUTs

3 Conclusion

The design aspects of the 5-Phase Block-Encoded Frame Partitioning (EFP5) circuit are being considered in this paper. The operation of this circuit is based on a set of internal phased signals that is used to generate timing patterns, which correspond to the blocks of each of the available fifty-two phase partitions under the control of a 6-bit word and for a time-frame of length 5. The VHDL description of the EFP5 cell is given. The corresponding simulation results verify the proper circuit operation while the internal phased signals *pclk*[10..1] and the output signals FPCLK[4..0] maintain the phase associations and the block encoded partitioning specification for the given frame of the input clock signal CLK. The synthesis results are given targeting an FPGA device.

References

[1] M. Abrarnovici, Y. Xiaoming, and E.M. Rudnick, *"Low-cost sequential ATPG with clock-control DFT,"* in Proceedings of the 39th Design Automation Conference, DAC 2002, New Orleans, Louisiana, USA, pp 243-248, 10-14 June 2002.

[2] C. Cao, and B. Oelmann, *"Mixed synchronous/asynchronous state memory for low power FSM design,"* in Proceedings of the Euromicro Symposium on Digital System Design, DSD 2004, pp 363-370, 31 Aug. – 3 Sept. 2004.

[3] W-K. Lee, and C-Y. Tsui, *"Finite state machine partitioning for low power,"* in Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, ISCAS'99, Orlando, FL, USA, pp 306-309, Vol. 1, 30 May – 2 June 1999.

[4] B. Oelmann, Oapos, and M. Nils, *"Asynchronous control of low-power gated-clock finite-state-machines,"* in Proceedings of the 6th IEEE International Conference on Electronics, Circuits and Systems, ICECS'99, Pafos, Cyprus, pp. 915-918, Vol. 2, 5-8 Sep. 1999.

[5] S. Poriazis, *Logic Design of Multiphase Finite State Machines*, Monograph, Phasetronic Laboratories, http://www.phasetroniclab.com Athens, Greece, 2001.

[6] S. Poriazis, *"The Two-Phase Twisted-Ring Counter Circuit,"* in Proceedings of the 2002 IEEE International Symposium on Circuits and Systems, ISCAS'02, Phoenix, Arizona, USA, vol. IV, pp. 858-861, 26-29 May 2002.