XILINX®

XAPP910 (v1.0) October 27, 2005

# Doubling Counter/Timer Resolutions with CoolRunner-II

## Summary

Counter and timer modules are often incorporated in CPLD designs. But, while counters and timers are ideal for some designs, they are not so ideal for designs that need greater resolution. This application note presents a method of doubling a counter's resolution by using the CoolRunner$^{TM}$-II DualEDGE triggered registers.

This application note demonstrates how to double timer/counter resolution by using DualEDGE triggered registers in CoolRunner-II CPLDs, and includes a design trick to achieve higher frequency beyond the device's maximum specified frequency, $F_{MAX}$.

## Introduction

Timer/Counter units are found in many applications. In some cases, they are used to measure elapsed time by counting processor cycles or clock ticks. For example, to generate a Pulse Width Modulation (PWM) signal by using two different initial or terminal count values, and a one-shot timer that toggles the I/O pin on overflow, the pin could be set to 1 for a desired amount of time, then 0 for a different amount of time, then 1 again, and so on. The period of the PWM signal in this case is a function of the sum of the two timer periods, and the duty cycle is the length of time that the pin is set to 1 as a percentage of the period.

$$P_{PWM} = P_{timer1} + P_{timer2}$$

$$DutyCycle = \frac{t_1}{(t_0 + t_1)} \times 100$$

PWM waveform accuracy is dependent on the input clock frequency. A 100 MHz clock can generate resolution accuracy of 10 ns. For a PWM waveform with 100 ns period and 60% duty cycle, the terminal count for '1' is six and for '0' is four for a 100 MHz counter. For the same period with 55% duty cycle (55 ns) at 100 MHz, the counter will have problem since it cannot count to 5.5. It will require a 200 MHz clock, which has 5 ns resolution, and count to 11 for 55 ns.

The CoolRunner-II CPLD with DualEDGE triggered registers are the ideal solution for this type of application. It can achieve 200 MHz counter resolution while using a 100 MHz clock, and solve the resolution problem while also maintaining the low power and low noise operation.

## DualEDGE Registers

CoolRunner-II DualEDGE triggered registers allow designers to reach unprecedented performance levels. CoolRunner-II CPLDs can double system performance by creating DualEDGE Triggered (DET) registers. CoolRunner-II DET registers allow data to be registered on both the rising and falling edge of a clock. DET registers can be used for logic functions that include shift registers, counters, comparators, and state machines. Designers must evaluate the desired performance of the CPLD logic to determine use of DET registers.

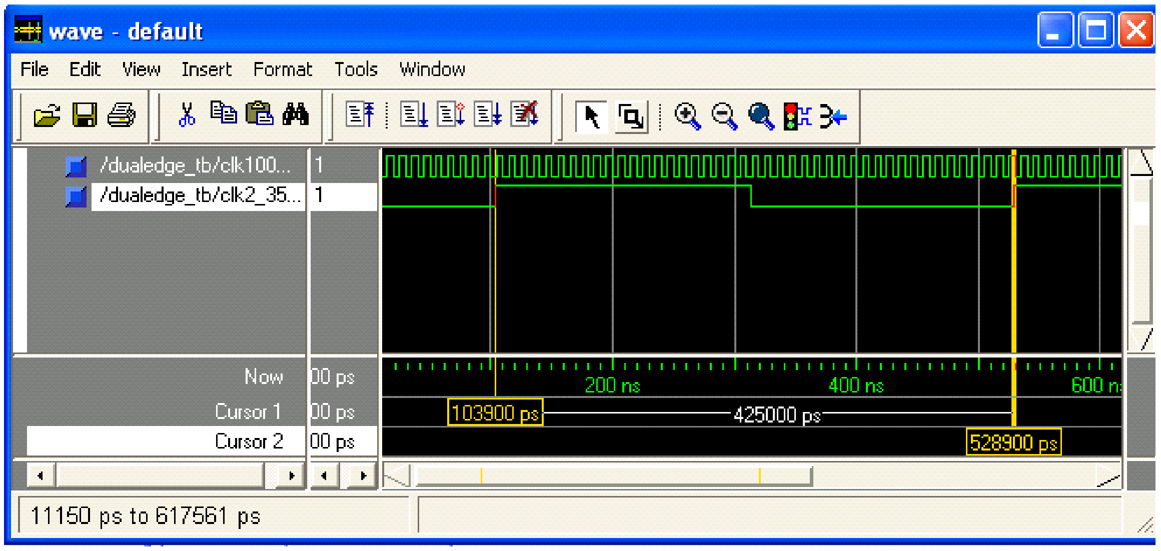
The DET register is available on all macrocells in all devices of the CoolRunner-II family.

## Doubling Resolution using DualEDGE

DualEDGE registers allow CoolRunner-II CPLDs to effectively double the resolution of any clocked operation. Let's examine a system that needs 5 ns of resolution. Given a 100 MHz clock in a conventional system, this would be impossible, as 10 ns is the best achievable resolution. However, if a counter is implemented with DualEDGE, it can actually accept a 100 MHz external clock, and count at an internal frequency of 200 MHz giving us the needed 5 ns of accuracy.

## Putting DualEDGE to Use

Let's use a more realistic example, rather than using perfect, round integer values. Imagine a system that requires a 2.35 MHz (425 ns period) clock output from the CoolRunner-II device, given a 100 MHz (10 ns) clock input. There is insufficient resolution given a 100 MHz input, and hence we need to use DualEDGE registers to complete the task. A 100 MHz clocked Dual Edge register would provide the equivalent of a 5 ns period. So, the DualEDGE counter would need to count 85 clock cycles in order to create a 425 ns period output clock.



/dualedge_tb/clk100... 1
/dualedge_tb/clk2_35... 1
Now 00 ps
Cursor 1 00 ps
Cursor 2 00 ps
200 ns 400 ns 600 ns
103900 ps
425000 ps
528900 ps
11150 ps to 617561 ps

*Figure 1:* **DualEDGE Clocking**

It should be noted that although the DET registers can double the system performance with a lower frequency clock input, the DET registers can not exceed the device maximum operating frequency ($F_{MAX}$). For example, the $F_{MAX}$ for XC2C64A is 263 MHz, so the incoming clock $F_{MAX}$ for the DET register must be 131.5 MHz or less.

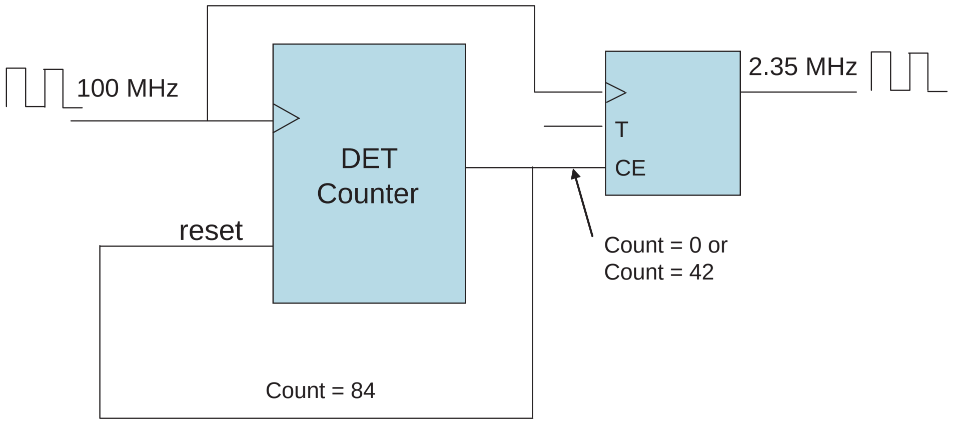


*Figure 2:* **Macrocell Clock Chain with DualEDGE Option Shown**

## DualEDGE Example Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity dualedgeclk is
  Port ( clk100Mhz : in std_logic;
    clk2_35Mhz : out std_logic);
end dualedgeclk;

architecture Behavioral of dualedgeclk is

signal counter: std_logic_vector(6 downto 0);
signal fn: std_logic;

begin

process(clk100Mhz)
begin
    if(clk100Mhz'event) then
      if(counter = 84) then
        counter <= (others=>'0');
      else
        counter <= counter + 1;
    end if;
end if;
end process;

process(clk100Mhz)
begin
    if(clk100Mhz'event) then
      if(counter = 0) or (counter = 42) then
        fn <= not fn;
      end if;
    end if;
end process;

clk2_35Mhz <= fn;

end Behavioral;
```

## Beyond $F_{MAX}$ with a Simple Design Trick

Let's say we need to generate a 2.34 MHz clock from a 200 MHz internal clock. Using the same trick as above, a 2.34 MHz frequency corresponds to a 427.5 ns clock period, and a 200 MHz frequency corresponds to a 5 ns clock period. In theory, using 200 MHz DualEDGE registers would allow a terminal counter to be clocked at an internal frequency of 400 MHz (2.5 ns resolution). Given this 2.5 ns resolution, we would simply need to count to a terminal value of 171 in order to generate a 2.34 MHz clock (427.5 ns period). But alas, as stated above, DualEDGE Triggered registers can only be clocked at an external frequency of $F_{MAX}$ divided by two, or less.

$$f_{EXT} \le \frac{f_{MAX}}{2}$$

Hence, we need another solution.

In order to generate a 2.34 MHz (427.5 ns period) clock from an input frequency of 200 MHz (5 ns resolution), we would need to have a terminal counter count to a value of 85.5. Notice that there is not enough resolution! Here is a trick-–we internally invert the incoming clock to the counter immediately after the counter counts to an integer value of 84. Doing this ensures that the next count will occur on the falling edge of the 85th clock cycle. This gives us the ½ cycle worth of resolution needed to count to 85.5. Figure 3 shows a block diagram of this circuit.

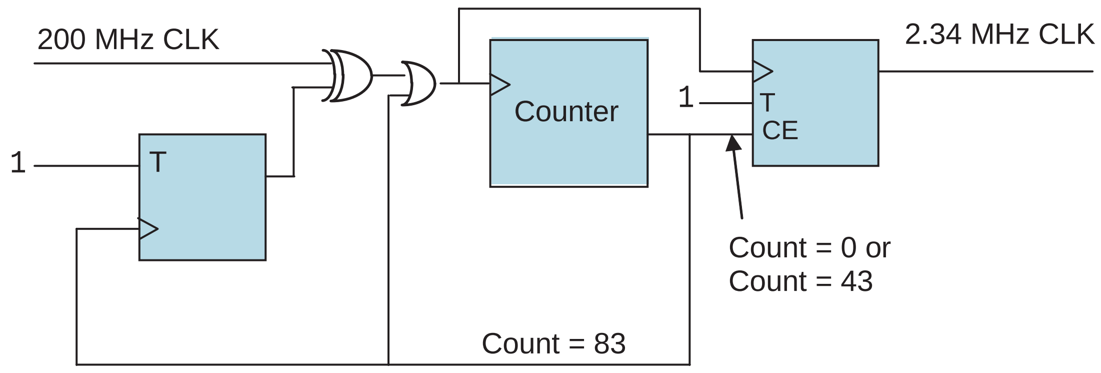


*Figure 3:* **Circuit Block Diagram**

From Figure 3, the counter counts to 84 (0 to 83) and toggles the left T register. This inverts the 200 MHz clock such that the counter actually counts on the next falling edge of the 200 MHz clock. The OR gate before the counter ensures that there is no glitch during the clock inversion process.

## Design Code Example

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library UNISIM;
use UNISIM.VComponents.all;

entity clkgen is
  Port ( clk200Mhz : in std_logic;
    clk2_34Mhz : out std_logic
        );
        attribute KEEP: string;
end clkgen;

architecture Behavioral of clkgen is

signal clka : std_logic;
signal ca1: std_logic;
signal ca2: std_logic;
signal qa : std_logic_vector(6 downto 0);
signal ta: std_logic;
signal taclk: std_logic;
signal fa: std_logic;
attribute KEEP of clka: signal is "TRUE";

begin

ca1 <= clk200Mhz xor ta;
ca2 <= '1' when (qa = 83) else '0';
clka <= ca1 or ca2;

process(clka)
begin
    if(clka'event and clka = '1') then
```

```
            if(qa = 84) then
                qa <= (others => '0');
            else
                qa <= qa + 1;
            end if;
        end if;
    end process;

    taclk <= '1' when (qa = 83) else '0';

    process(taclk)
    begin
        if(taclk'event and taclk = '1') then
            ta <= not ta;
        end if;
    end process;

    process(clka)
    begin
        if(clka'event and clka = '1') then
            if(qa = 0) or (qa = 43) then
                  fa <= not fa;
            end if;
        end if;
    end process;

    clk2_34Mhz <= fa;


end Behavioral;
```

Figure 4 is a simulation waveform for this design. It shows that the CoolRunner-II device is capable of generating a 2.34 MHz clock from a 200 MHz input signal.

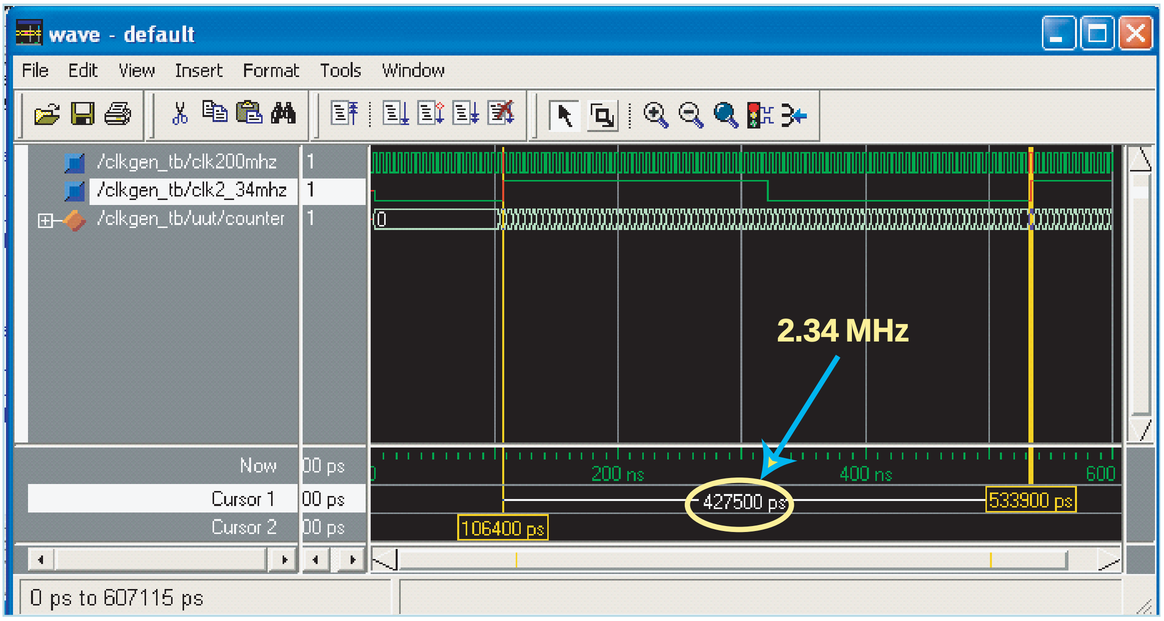


*Figure 4:* **Simuation Waveform**

Figure 5 shows how, after the 84th rising edge, the clock is internally inverted, and the counter proceeds to count on the falling edges. Note the Tco delay as this is a timing simulation.

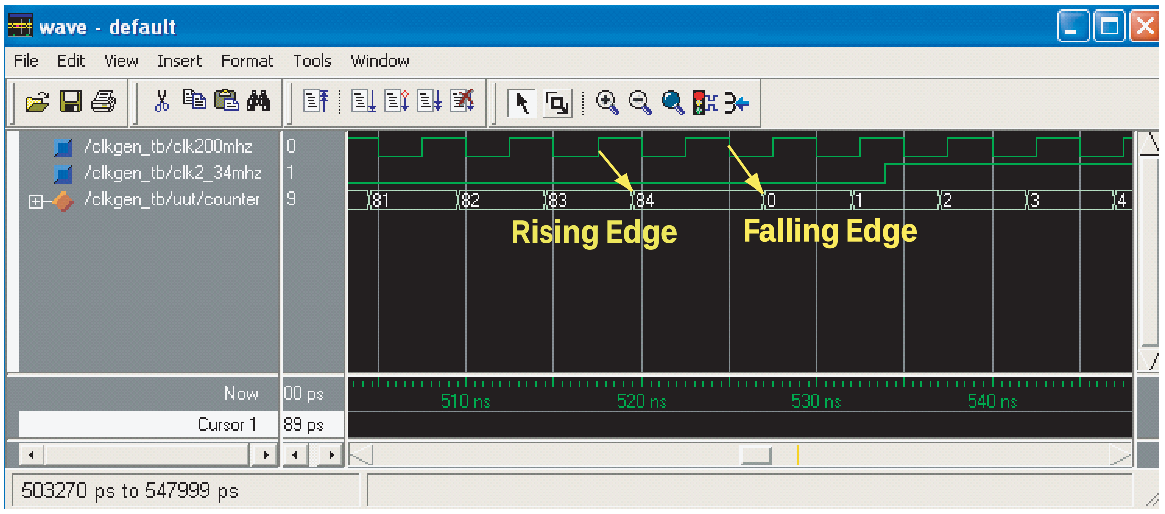


*Figure 5:* **Simuation Waveform using Different Clock Edges**

## Additional Information

CoolRunner-II Data Sheets and Application Notes

## Conclusion

CoolRunner II CPLDs are the ideal solution for providing the highest resolution for Timer/Counter applications. CoolRunner-II CPLDs are targeted for applications that require both low power consumptions and high performance.

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 10/27/05 | 1.0 | Initial Xilinx release. |