APPLICATION NOTE

Xilinx has acquired the entire Philips CoolRunner Low Power CPLD Product Family. For more technical or sales information, please see: www.xilinx.com

XAPP 309

Clock modulation: How to synthesize additional clocks for counters and state machines

Frank Wirtz
CPLD Applications Specialist

1998 Jul 13

Philips
Semiconductors

PHILIPS

Clock modulation: How to synthesize additional clocks for counters and state machines

XAPP 309

Xilinx has acquired the entire Philips CoolRunner Low Power CPLD Product Family. For more technical or sales information, please see: www.xilinx.com

INTRODUCTION

The Philips CoolRunner™ devices allow for both synchronous and asynchronous (product term) clocking within its architecture. The 32 macrocell device allows for the use of two clocks, while the 64 and 128 macrocell devices provide the user with a total of four clocks. In some instances, designs require the use of more asynchronous clocks than are currently provided by the XPLA™ architecture. While the CoolRunner family will be enhancing its current flexibility by adding additional clocking resources shortly, this note deals with a technique to synthesize an additional asynchronous clock term as long as a higher frequency global clock is available. An excellent prerequisite applications note to read is 'Understanding CoolRunner Clocking Options'.

WHO CARES?

Some of our competitors have the ability to easily implement asynchronous clocks throughout their devices. This capability is afforded at the cost of other things, such as output enable control, flexible routing, and speed. Many designers, having implemented their designs with multiple clocks, wish to 'trade up' and enjoy the performance, power, and flexible routing capabilities of the CoolRunner family. There can be a porting issue if their designs exceed the number of clocks available in the XPLA architecture. This situation can be easily remedied if a higher speed (Nyquist) clock is available.

IN THE BEGINNING . . .

Many of the designs that require additional clocking have register elements that are used only as one bit state machines or as 'flags' to indicate events that have transpired. Typically a one or a zero is clocked into a register by this 'asynchronous' clock, and then reset or preset by a later event. These registered bits can easily be synthesized using 'soft' latches or registers (refer to 'Understanding CoolRunner Clocking Options'). However, if a higher speed clock is available in the design, then a clock enable technique can be used to gate the clock to a register. The higher speed clock may be used only if it is greater than two times the frequency of the 'asynchronous' clock, and only if the high speed clock is continuous or at least present when gating is required. Gating a clock in this fashion is called 'Clock Enabling', and is not provided for by hardware in the XPLA architecture. This technique was originally implemented manually using the schematic in Figure 1. The software tool XPLA Designer provides for a clock enable syntax by automatically synthesizing this logic. The syntax to implement a clock enable in a *.phd file is as follows:

```
equations
   data_out.clk = clk;
   data_out.d = data;
   data_out.ce = en;
end
```

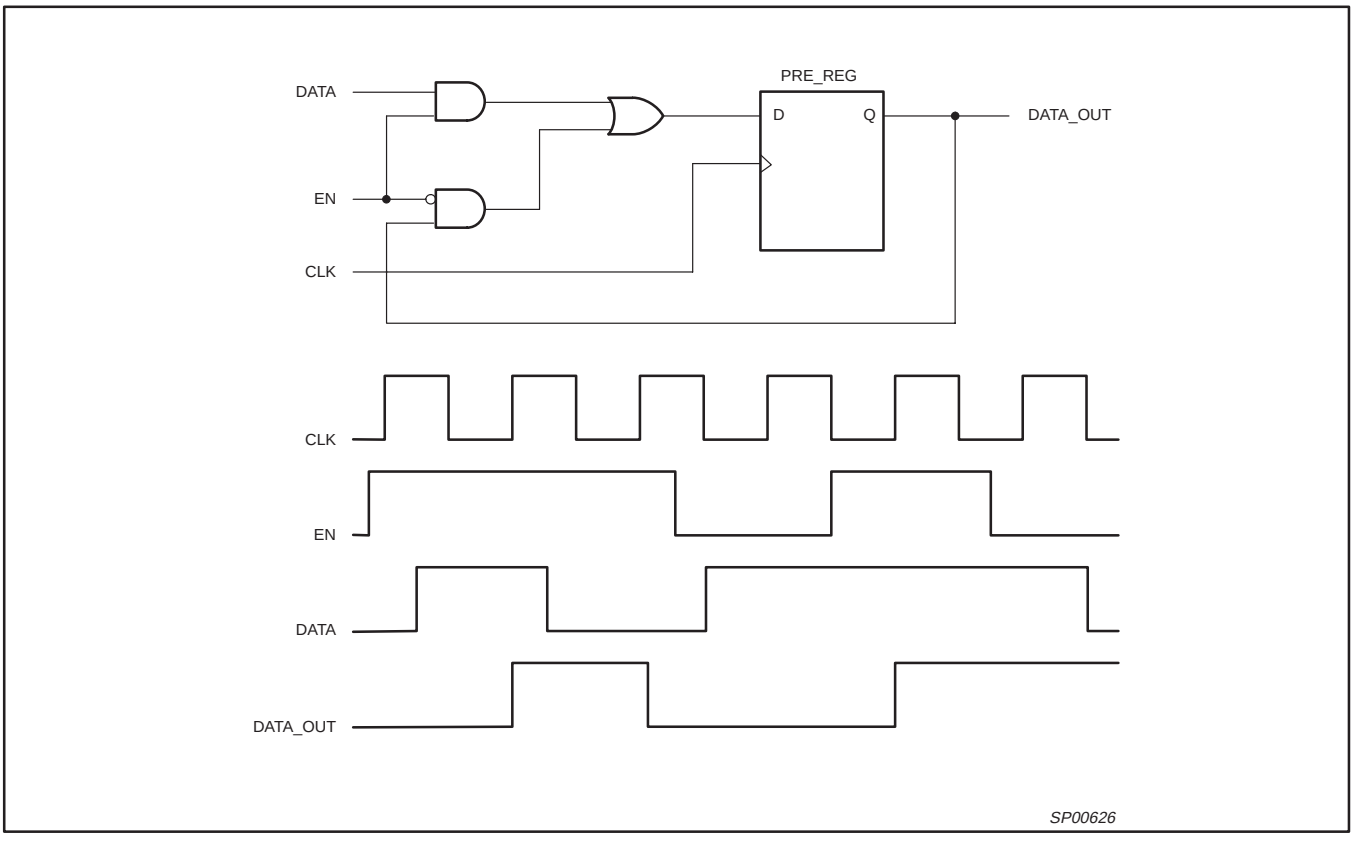

Figure 1.

This technique works well for setting or clearing a registered bit, and it is implemented without requiring an additional macrocell. However, it does not lend itself easily to counters or applications where the input data is not stable during enable. Clock enables favor the implementation of latches more so than registers. Additionally, metastability situations can occur, which will be discussed later. Edge triggered flip flops can be synthesized in 'soft' logic, however they require the use of two macrocells per bit of registered information.

SO WHAT NEXT?

If a designer wants to use additional clock terms beyond those provided by the XPLA architecture, and a faster clock is available in the design, then the clock can be modulated easily by adding a small amount of additional logic. Examine Figure 2.

In this instance, the designer is out of clock resources, but wants to implement a four bit counter to be clocked by the signal 'async_clk'. Note that the portion of the schematic contained in the dashed line is the clock modulation circuit, and it consists of only a 'D' type clock enabled flip flop, a 1n_2and gate, and an input register. The other portion of the circuit is a 'T' type counter, enabled with the 'enabler' signal. The input register (pre_reg) is used to delay the async_clk signal by one clock period and therefore lessening (but not removing entirely) the risk of metastability affecting the MTBF of the design.

At power up, all registers are initialized as reset. pre_reg and en_reg are cleared at this time, or cleared one clock period (plus T_{CO}) after async_clk is low. The output of en_reg is low, which readies the 'enabler' gate(1n_2and) for the high going edge of the registered async_clk. At this point (reset condition), 'enabler' is low, so none of the registers (other than the input register pre_reg) will clock.

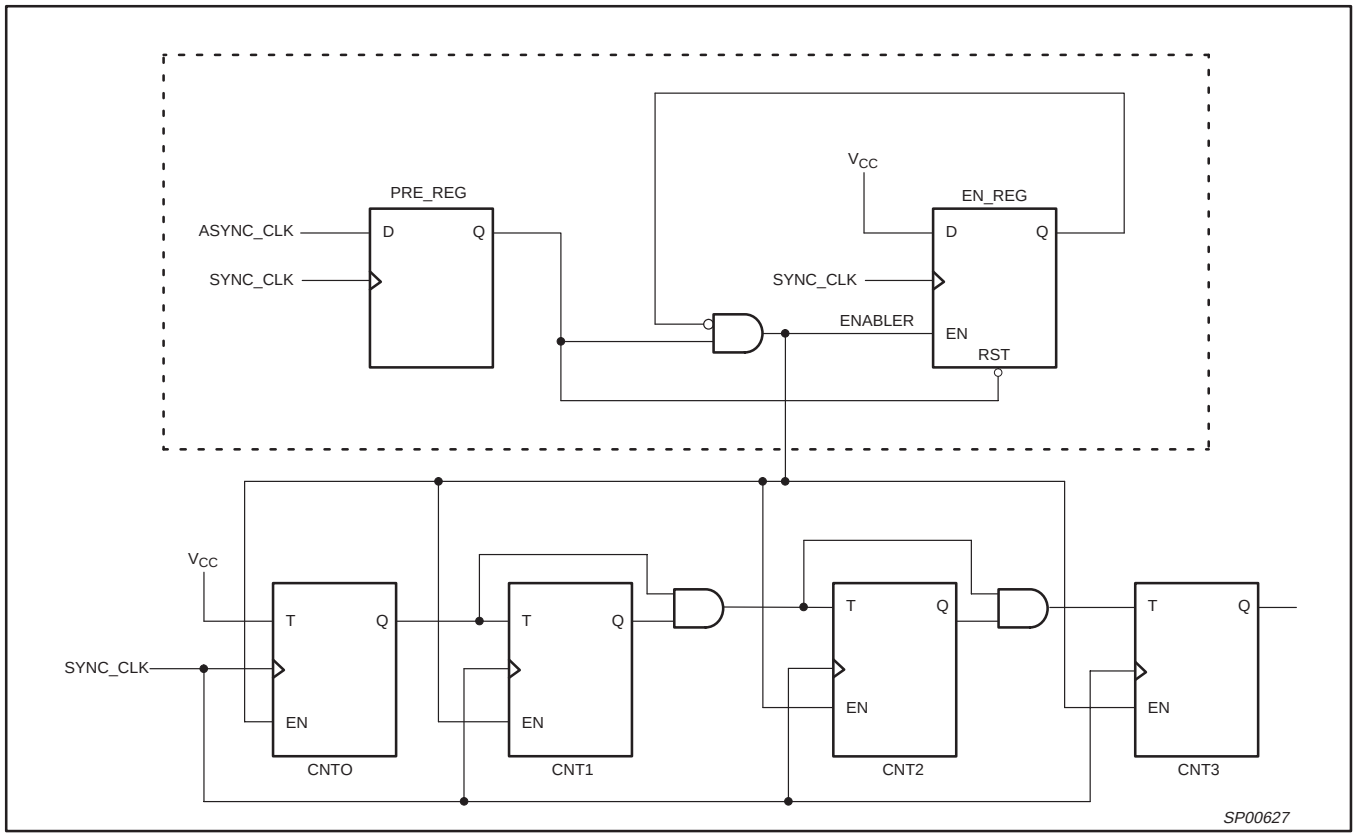

Figure 2.

Some time later, async_clk will transition from a low to a high. pre_reg.q is asserted, and ‘enabler’ asserts, allowing all of the enabled registers to clock at the next sync_clk pulse. When sync_clk does clock all of the registers, a logic level high is clocked through to the output of en_reg, which disables the clock enable control. When ‘async_clk’ returns low, it is clocked through pre_reg, and resets the en_reg circuitry. The entire modulation circuit only allows for one ‘sync_clk’ to be gated through to other registers per ‘async_clk’ low to high to low transition. This circuitry can be implemented in XPLA Designer using the following *.phd syntax:

```
Module enabler
Title ‘Clock modulation circuitry and counter’

sync_clk     pin;                ″synchronous clock must be 2x async clock freq.
async_clk    pin;
pre_reg      pin istype ’reg’;   ″registration register... defer metastability
en_reg       pin istype ’reg’;   ″modulation register

enabler      node istype ’com’;  ″node for enabling counters, state mach, etc.

cnt3..cnt0   pin istype ’reg’;   ″four bit test counter register

count = [cnt3..cnt0];            “name the counter something unique

equations

enabler = !en_reg.q & pre_reg.q;″define enabler signal

pre_reg.clk = sync_clk;          ″clean up async timing
pre_reg.d = async_clk;

en_reg.clk = sync_clk;           ″Clock available one time only!
en_reg.ce = enabler;
en_reg.d = 1;
en_reg.ar = !pre_reg.q;

count.clk = sync_clk;            ″four bit counter stuff.
count.ce = enabler;
count = count + 1;

end
```

Notice how easily the ‘enabler’ signal can be distributed throughout the design. This technique can be used on groups of registers for counters, data buffers, and state machines. Additionally, this modulation circuitry can be implemented at a one time cost of two macrocells per asynchronous clock regardless of the amount of registers controlled.

SIMULATION OF ENABLER DESIGN

Figure 3 shows a screen capture from XPLA–Sim for the Enabler Design. Note that the counter bits (CNT3..CNT0) increment once for each rising edge of ASYNC_CLK, and that the edge of each CNTx transition is delayed by a maximum of two SYNC_CLK periods.

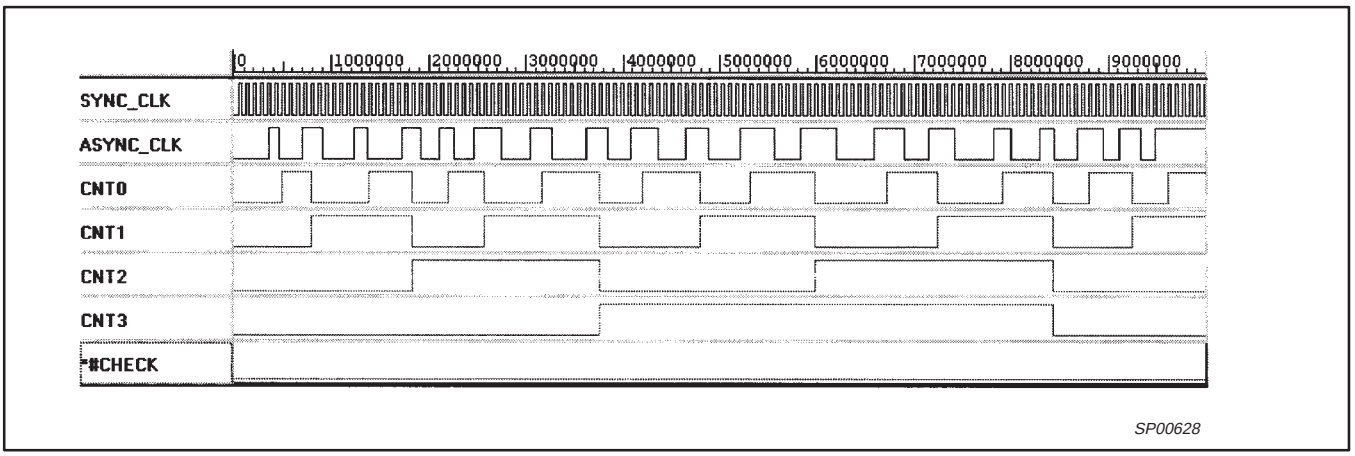

Figure 3.

Clock modulation: How to synthesize additional clocks for counters and state machines

AN075

Definitions

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Disclaimers

Life support — These products are not designed for use in life support appliances, devices or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Philips Semiconductors
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088–3409
Telephone 800-234-7381

print code Date of release: 07-98

Document order number: 9397 750 04141

Let's make things better.

Philips Semiconductors

PHILIPS