
A Programmable Clock Generator Based on Xilinx CoolRunner®

Gunther Zielosko, Heiko Grimm

1. Objectives

As presented in our previous application note we would like to develop applications for BASIC-Tiger® which combine its intelligence with fast hardware based on complex programmable logic devices (CPLDs). We chose the Xilinx CoolRunner® family as a basic circuitry which are available as so-called CoolRunner®2 for applications up to 300 MHz (!).

What exactly is our programmable clock generator supposed to do?

Almost every electronics workshop needs a clock generator. It would be best if it worked quartz-precise and its frequency was adjustable at the same time. This is actually contradictory, because quartz-precise means working with fixed frequencies. Anybody who has come across PLLs (Phase Lock Loop – phased frequency adjustment) knows that there are some tricks. This may be nice, but we will pursue another path, since you will only get an exact frequency by dividing the quartz-precise basic frequency. PLLs show slight frequency fluctuations (jitters) because of its type of control. Normally in digital technology frequencies are divided by flip-flops, e.g. a 10 MHz input frequency becomes 5 MHz after a flip-flop. After another flip-flop 2.5 MHz are outputted etc. This is smart but not flexible enough for our purposes. You will gain better results with a programmable divider which is able to divide the input frequency by almost every number. This is implemented by constructing a counter which counts input impulses and passes the result on to a bit comparator which compares it with a given value. When the count reaches the given value the counter is reset and restarts counting. Like this the counter can be reset exactly after e.g. 1,000 input clock cycles. The reset impulse is evaluated and is outputted as an output frequency. To make things really comfortable the counter should be quite fast (high output frequencies), boast many steps (fine frequency graduation) and contain an elegant interface (simple control using BASIC-Tiger®). Our guidelines are as follows:

- Input frequency of 0 to 100 MHz as possible,
- 24 bit divider (16.8 million steps!),
- we will use the extended EPort system as an interface for BASIC-Tiger®, i.e. the component to be developed has to contain the whole logic of 3 EPorts with 8 outputs each (Four 74HC377 and one 74HC138 in the Plug-and-Play-Lab).

2. Solution based on XCR3064XL in a 44 pin PLCC package

The objective mentioned above can be solved with a single XCR3064XL module in a 44 pin PLCC package. The pin assignment is shown in figure 1; figure 2 depicts the connection to BASIC-Tiger®, obviously no big deal.

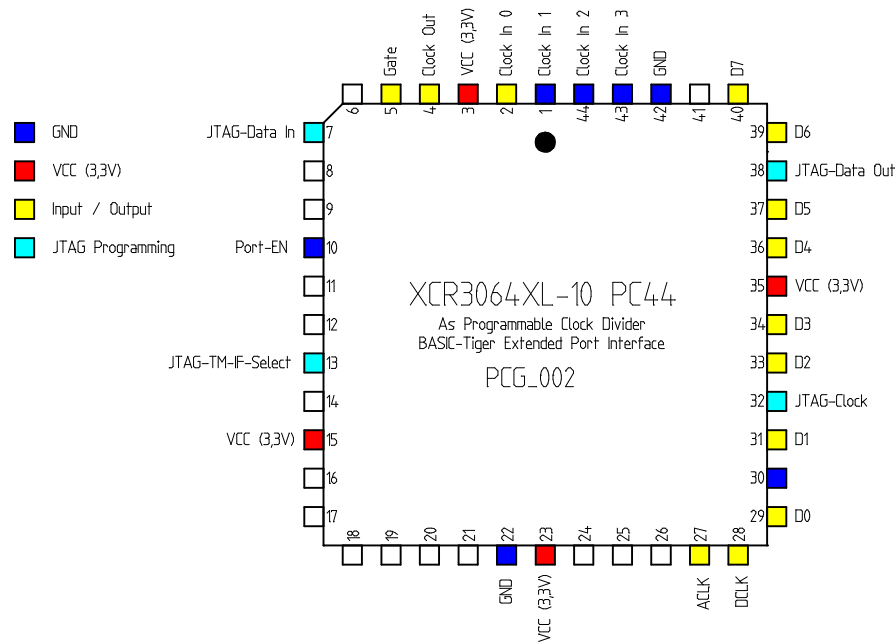


Fig. 1 XCR3064XL in a 44 pin PLCC package as a programmable divider (PCG_002)

JTAG pins depicted in light blue are used for programming the XCR3064XL-10 PC44 and remain not connected in the user's circuitry, just as the white labelled pins. Clock inputs 1, 2 and 3 as well as the pin Port_EN have no pull-up resistor and have to be connected to GND. VCC (3.3V) pins are supplied by a voltage regulator (e.g. LM34801M3_3V) and additional ceramic capacitors are connected near the VCC pins. After all our divider is supposed to work with maximum 100 MHz, so even a "CoolRunner[®]" needs dynamically quite some current.

Although CoolRunner[®] operates at 3.3 V, it also tolerates 5 V input signals, so nothing is opposing a direct connection to BASIC-Tiger[®]. Vice versa most logic families also accept the 3.3 V levels of CoolRunner[®].

Example: Input frequency $F_i = 4.915200$ MHz

Factor T	Output frequency F_o
2	2.457600 MHz
4	1.228800 MHz
6	819200 MHz
20	245760 kHz
1,000	4915,2 kHz
1,000,000	4.9152 Hz

Keep in mind that a factor of up to $2 \cdot 16.8$ million is possible. Therefore, and due to the theoretical input frequency of up to 100 MHz, virtually arbitrary frequencies can be set – up to years, if you have the time... Something else concerning the really achievable input frequency: The authors achieved up to 40 MHz using the 10ns CoolRunner[®] version, although the module, being equipped with a 24-bit counter, a bit comparator (real time) and a not quite simple interface, is almost “full”. This means compromising on internal “wiring” concerning the signal layout and as result restrictions concerning speed. Using a 6ns version or a version with more macro cells and registers 100 MHz are realistic. The output frequency always has a pulse-duty factor of 1:1, i.e. oscillation is symmetric.

The gate input (pin 5) is used for switching the output signal on and off externally. In the case of an open input or logic “1” the set frequency is outputted, in the case of logic “0” or in the case of connection with ground the output is grounded. This is one option to determine the output frequency, i.e. switch the output on and off in synch with external information. Such a function is used e.g. in remote controls where infrared LEDs are switched on and off at about 35 kHz using coded commands. Information source can be any external device and, of course, the BASIC-Tiger[®] - great opportunities for using our programmable frequency generator.

The encoder connected to pins L70 and L71 with its button at L72 is used for setting the divisor in the sample device and can be left out when setting the frequency otherwise. In the program PCG_002.TIG we set the frequency with this encoder in wide ranges.

3. BASIC-Tiger[®] program PCG_002.TIG

PCG_002.TIG stands for “Programmable Clock Generator”. The program implements a comfortably programmable clock generator using the presented hardware. The quartz respectively input frequency is set fix in the source code as variable F_i . It is also possible to implement this with an input function (e.g. with the encoder!). This is followed by a program part for dynamical querying the encoder, i.e. in the case of fast rotation very large “steps” are achieved whereas in the case of slow rotation only single steps are taken. The result is multiplied with 2 and used as divisor T (Range $2 \dots 2 \cdot 16.7$ Mio.). From this the program calculates the output frequency and the period (cycle duration) and then outputs the following values:

Input frequency (F_i)
Divisor (T)
Output frequency (F_o)
Period (P)

The input frequency F_i is displayed in Hertz and so is the output frequency F_o . The latter, however, has decimal places (comma), since the output frequency can become very low. Similar applies to the period P which lasts starting from ns up to many seconds. The “multi display” with F_i , T, F_o and P is helpful when setting, because you can concentrate on the desired value and set the target value as exact as possible.

More advice concerning setting the frequency: To avoid undefined counter sequence states it makes sense to suspend the counter when setting. We achieve the latter by issuing a reset with a “1” via EPort address 13h in the BASIC-Tiger[®] program PCG_002.TIG. Only afterwards the 24-bit word’s three bytes are outputted via addresses 10h, 11h and 12h. Then the reset state is cancelled.

By the way, using a BASIC-Tiger[®], the divider with CoolRunner’s[®] and a suitable BASIC-Tiger[®] program you can implement an excellent frequency analyzer. By being able to set an arbitrary divisor you can measure frequencies in a very broad range.



Fig. 3 Display output at a quartz frequency of 4.915200 MHz and factor 2 (maximum frequency)

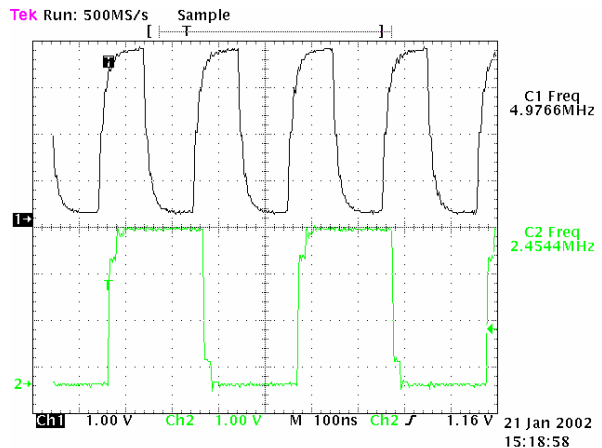


Fig. 4 Input signal (black) and output signal (green) at factor 2...



Fig. 5 ...and at factor 10

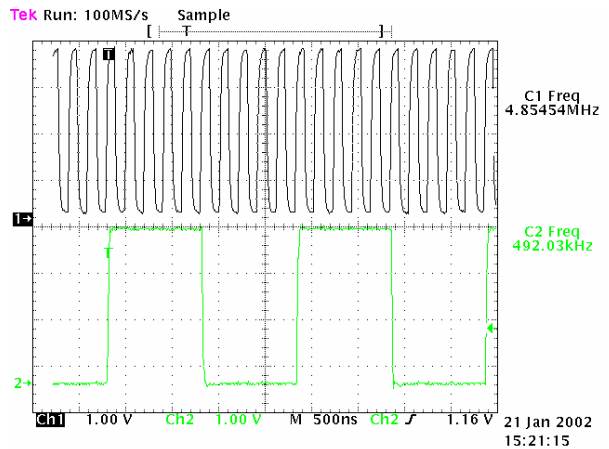


Fig. 6 Input (black) and output signal (green) at factor 10

Program PCG_002.TIG, of course, is just a simple example which can be customised to the user's purposes. Besides setting the input frequency it is also possible to measure it indirectly in a simple way. Options for frequency measurement presented in both the BASIC-Tiger[®] manual and in some application notes can be used. You only have to set the clock output so that the output frequency can be measured by BASIC-Tiger[®], i.e. some kHz to some Hz. Then the frequency is measured and the input frequency is recalculated via the divisor. The final step is adopting the value into variable Fi. Please avoid "stressing" the connection to BASIC-Tiger[®] accidentally with many Mhz.

Finally fig. 7 shows the authors' model construction for the Minilab. As you can see the hardware effort for this task is minimal. You can either input the input frequency via the quartz generator or via the HF socket. The encoder can be seen on the right.

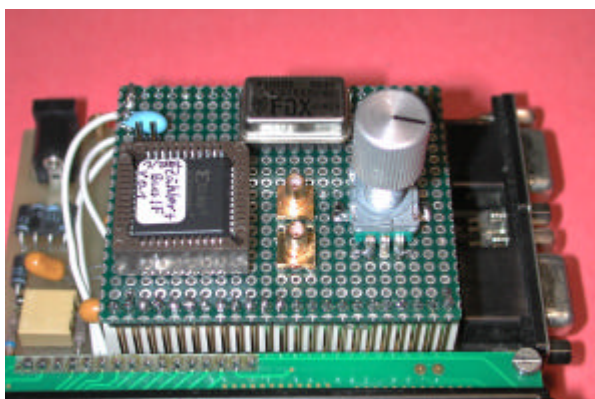


Fig. 7 Model construction with bread board for the Minilab

4. A stand-alone version of the programmable divisor

In the course of development at first a divider version was created with a direct connection of the 24-bit information, i.e. without BASIC-Tiger[®] interface. This version can be programmed e.g. via DIL switches, jumpers or another microcomputer system. Because of the simple internal wiring this version is also much faster. Here even the CoolRunners[®] 10ns version was able to reach 100 MHz. Figure 8 shows its pin allocation and figure 9 shows the authors' breadboard construction. The input frequency is put to pin 2; the divided frequency is outputted at pin 41. RESET is high active and the 24 bit for the divisor are put statically to the inputs In0 to In23. The file PCG_001.JED enclosed to this application note contains a tested solution.

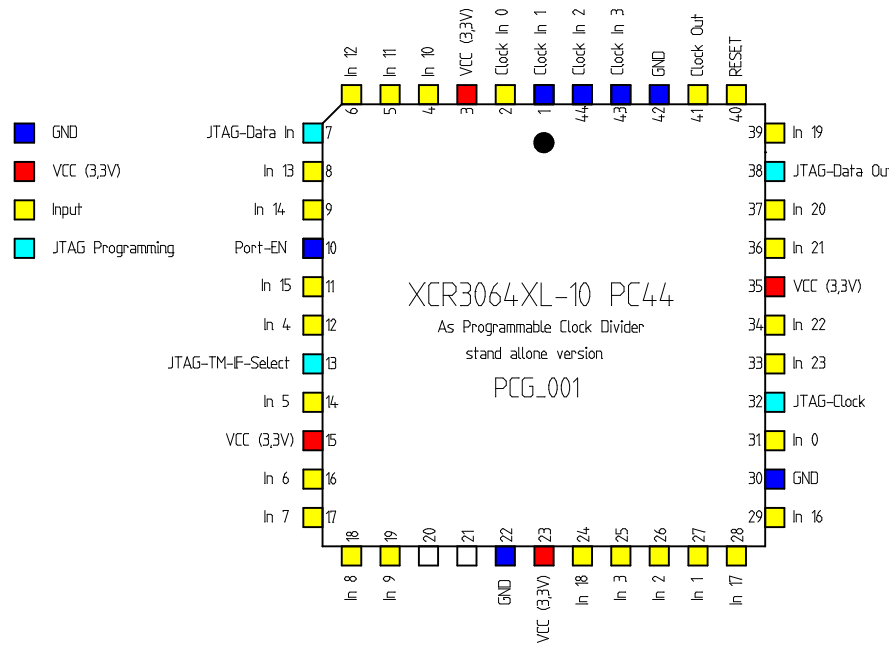


Fig. 8 Pin allocation of the stand-alone version

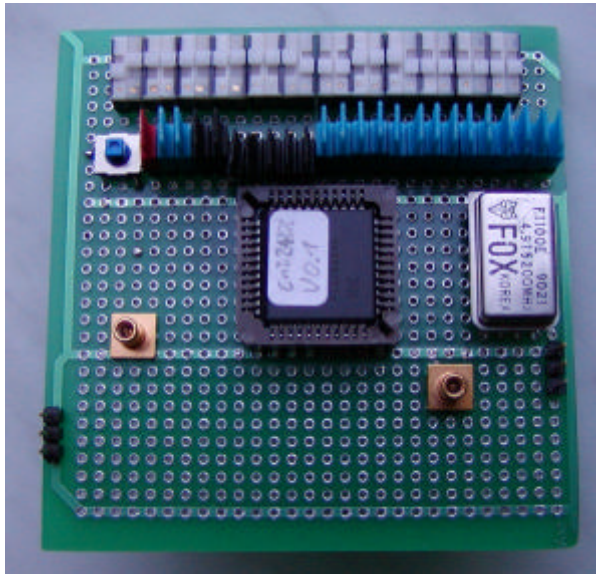


Fig. 9 Divider without interface

5. How to get the XCR3064XL as programmable dividers

The files PCG_001.JED and PCG_002.JED offer a finished solution for experts on the XILINX CoolRunner[®] system. They are enclosed to this application note. They are not suitable for novices on this terrain, which can fight their way through with information and software from the XILINX websites as well as with the programming adapter also offered there. However, it is also possible to obtain finished boards, programming adapters and/or completely programmed modules from the author with support from Wilke Technology.