
An “afterburner” for the Tiger

Gunther Zielosko, Heiko Grimm

1. Objectives

Anybody who deals with the BASIC-Tiger® cannot complain about the programming convenience (even with complex tasks) and the great options for data input and output. But this kind of “intelligence” has its price, that is to say the limited speed of completing tasks. E.g. a programmable clock generator, a logic analyser or a pattern generator connected to a BASIC-Tiger® would be hardly faster than a few kHz. If an exact frequency is vital we face another problem. The BASIC-Tiger® evinces slight clock deviations among each other in every clocked application. If you need something faster and more precise, it takes either a lot of money (finished devices) or the developer solves the problem with elaborate hardware, where most of the time comfort falls by the wayside. Is it not possible to combine the BASIC-Tiger® system’s convenience with a fast hardware? It is...

In this application note we will discuss solutions which allow a relatively easy implementation of hardware based programmable clock generators, logic analysers and pattern generators which can be conveniently programmed using the BASIC-Tiger® system and will then operate independently. The first tangible example, which is going to be dealt with in the next application note, is a programmable frequency divider which divides an input frequency (e.g. from a quartz oscillator) by an adjustable number.

2. Basics

The basis of our hybrid solution is a fast logic part which can consist of either single modules in TTL, HC, HCT or ordinary CMOS technology or an integrated PLD solution, depending on the conditions. The integrated PLD solution is of course very smart and much faster than a version using standard modules, but it requires the user to have some skills in programming and testing such modern elements. We will need a fast counter respectively divider as a basic element for the applications (clock generator, logic analyser or pattern generator) presented in the following sections. Depending on the application a fast static RAM will be used later.

- The **programmable clock generator** has only one counter which uses a fast external clock. The counter’s outputs are compared with statically fed information via a bit comparator. If these values (actual counter, static information) are identical the counter is reset. If static information is adjustable an arbitrary counter divider ratio can be set – i.e. the high input frequency can be reduced in many steps. In the case of choosing an input frequency of e.g. 10 MHz, the original clock length is 100 ns. All clock lengths of 200 ns, 300 ns, 400 ns up to seconds, minutes and hours can be derived from this – it only depends on the divisor sequence length. Here BASIC-Tiger® would only take over the part

of the static comparative word. Like this it can serve as a very simple frequency generator due to its computing power, its display and some operating elements.

- In case of a **logic analyser** the addresses of a fast static RAM are counted up in write mode via our fast-clocked counter, while the information is read in via the 8 inputs (8 bits). At the end of the writing procedure the fast clock for the counter is switched off. After resetting the counter the BASIC-Tiger® takes over clocking the counter – now the RAM is in the read mode and the saved information is read into the BASIC-Tiger® and evaluated via the I/O data pins now set to output. Here the clock can be arbitrarily slow or even stop at an actual count. This allows elegantly evaluating single data or small areas.
- The **pattern generator** works similar. First of all pattern (8-bit words) of our choice are written to the RAM using a slow BASIC-Tiger® clocking. This happens by feeding information from the BASIC-Tiger® to the RAM data I/Os set to input. After a RESET the clock is switched to fast and all 8 bit words are read in a very fast sequence, at which RAM data pins now work as outputs.
- We mustn't forget further solutions on this basis – fast AD respectively DA converters can be upgraded to an oscilloscope or function generator in co-operation with an also fast RAM and its address counter. Basically, it also works with two clocks in succession – one for the real time application and one for evaluation.

3. The counter's/ divider's hardware

3.1. The conventional method – standard modules

In contrast to most previous application notes there are no standard solutions for this case – just animations for your own developments. As mentioned before we will need a fast counter/divider with many stages. There are no such counter components as finished standard modules, but at least some with several stages per module. Table 1 shows a selection of standard counters respectively dividers which are more or less suitable for this purpose.

Version	Description	Max. frequency
74LS192	Sync. Up/Down Dual Clock 1 decade BCD Counter	ca. 10 MHz
74LS193	Sync. Up/Down Dual Clock 4 bit Binary Counter	ca. 10 MHz
CD4029	Presetable 4 bit Up/Down Counter (binary)	ca. 10 MHz
CD4520	Dual Binary 4 bit Up Counter	ca. 10 MHz
74LS292	31 Stage Programmable Frequency Divider	30 MHz
74LS294	15 Stage Programmable Frequency Divider	30 MHz
74LS668	Synchronous 4 bit Up/Down Counter (decimal)	ca. 30 MHz
74LS669	Synchronous 4 bit Up/Down Counter (binary)	ca. 30 MHz
74HCT4020	14 Stage Binary Ripple Counter	

Tab. 1 Standard counter/divider modules in different technologies

As you can see from table 1, the achievable frequencies as well as modules and wiring required for an extensive counter slightly damp our enthusiasm. Standard modules reach some MHz maximum. If you plan to construct long chains, you will have to take care as well that delay times of the single steps do not become longer than the clock – not every module is suitable for such an application. Fig. 1 shows one possible solution for a programmable divider – we used the decimal counter version 74LS192. Programming takes place via decade pre-selector switches which set 4 bits each in BCD code. For this we will use the setting inputs of the 74LS192; they set the counter to the given decimal number and then countdown starts. Reaching zero, the complete counter is set again etc. Setting could also be done by BASIC-Tiger, simply using e.g. 24 outputs of the extended I/O system at the Plug-and-Play-Lab. At the divider’s output we connected a 74LS123 monostable multivibrator which converts the very short set impulses to an impulse of defined length.

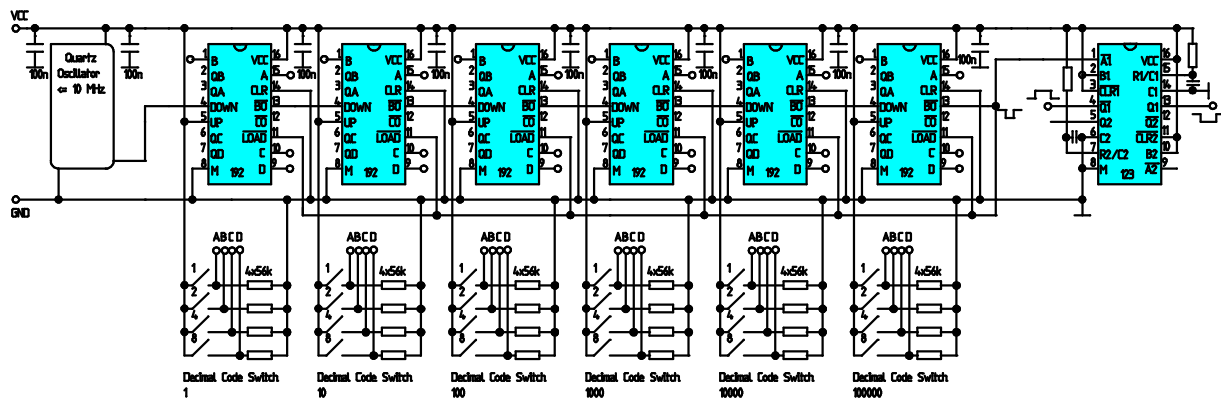


Fig. 1 Programmable clock generator implemented by “discrete” decimal 74LS192 counter elements

3.2. An elegant alternative: CPLDs

Figure 1 depicts impressively the effort which has to be taken for such a relatively simple task. Solutions which do not use a standard module but so-called CPLDs (Complex Programmable Logic Device) are much handier. Such CPLDs are modules where individual, sometimes quite extensive logic functions can be created by a programming procedure. This programming procedure is done by the user who, of course, has to have software and hardware for programming these otherwise useless modules at his disposal. In the following sections we will deal with a very modern solution based on this: The XILINX[®] CoolRunner[®] family.

3.2.1. XILINX[®] CoolRunner[®] family

The name of these programmable logic devices indicates one of their essential advantages: They require only little power (they remain “cool”) despite their high speed (“runner”). There is a whole range of devices which differ from each other concerning their number of macro cells, their gate functions and registers. The achievable speed is in a range above 100 MHz (!) depending on the version (there are devices with gate switching times of about 6.7 to 10 ns).

Therefore these universal logic devices are especially suitable as fast supplementary system for the BASIC-Tiger[®], also due to their price of less than 5€ We cannot provide an extensive description of this technology in this application note, but we will present some basics.

As a logic system these devices contain fast RAM cells which are connected by interconnect cells as required by the user. This connection structure is saved in the internal EEPROM and is loaded to the faster RAM when operating voltage is supplied, which by the way is 3.3 V here (!).

A comprehensive software development system (for programming via the PC and its parallel interface) is available for free download on the Xilinx website:

<http://www.xilinx.com/products/software/webpowered.htm>

The Xilinx programming adapter's circuitry is available on:

http://user.cs.tu-berlin.de/~sirhenri/sides/pld_xilinx/jtag.pdf

Table 2 shows the currently available versions of the CoolRunner[®] family. We chose version XCR3064XL in the PLCC44 package for our experiments.

	XCR3032XL	XCR3064XL	XCR3128XL	XCR3256XL	XCR3384XL	XCR3512XL
Macro cells	32	64	128	256	512	384
Available gates	800	1600	3200	6400	12800	9600
Registers	32	64	128	256	512	384
System speed (MHz)	175	145	145	140	127	127
44 pin PLCC	36*	36*				
44 pin 1mm VQFP	36*	36*				
48 in 0.8mm CSP	36*	40*				
56 pin 0.5mm CSP		48*				
100 pin 1mm VQF		68*				
144 pin 0.8mm CSP			84*			
144 pin 1.4mm VQFP			108*	120*		
208 pin PQFP			108*	164*	172*	180*
25 pin FINELINE BGA				164*	212*	212*
280 pin 0.8mm CSP				164*		
324 pin FINELINE BGA					220*	260*

* Available I/O pins

Tab. 2 XILINX CoolRunner[®] versions

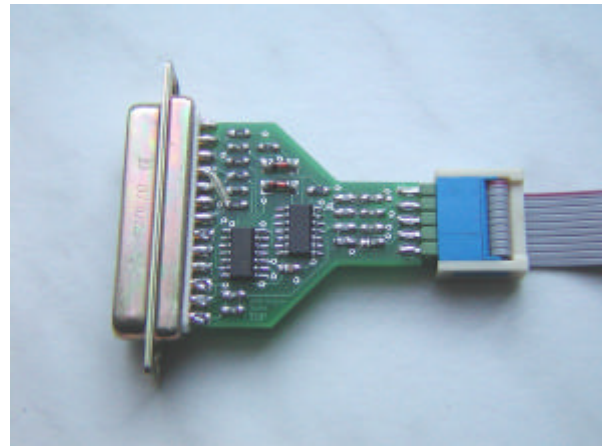


Fig. 2 The XCR3064XL in a 44 pin PLCC package Fig. 3 A self-made programming adapter package

We learn from table 2 that our XCR3064XL has 36 available I/O pins in its PLCC package which can arbitrary be used (with certain restrictions) as inputs, outputs or clocks. The residual of the 44 pins are used as GND, VCC or programming pins.

Besides, although the CoolRunner[®] works with 3.3 V, it also tolerates 5 V input signals, so it can be connected directly to BASIC-Tiger[®]. Vice versa most logic families accept the CoolRunner[®] 3.3 V levels.

For further applications an additional fast static RAM is required. You can start off with e.g. a 6164. After all a pattern generator with 8 Kbytes can be implemented this way. 8 Kbytes (exactly 8,192 or 2^{13}) require 13 addresses, i.e. our divider should have at least 13 stages.

In the following application notes we will implement concrete projects based on the CoolRunner[®] modules. Besides the applications with counters mentioned above we can also think of pure logic extensions, e.g. expansion modules for additional freely programmable I/O ports (PIO principle). For those applications we will provide both the respective data file for the “DIY programmer” and in certain cases completely programmed modules. Like this the handicraft enthusiast can resort to latest technology and develop efficient boards.