„Nachbrenner“ für den Tiger

Gunther Zielosko, Heiko Grimm

1. Die Aufgabenstellung

Wer mit dem BASIC-Tiger® arbeitet, kann sich kaum über den Komfort bei der Programmierung auch komplexer Aufgaben und die tollen Möglichkeiten der Datenein- und Ausgabe beklagen. Aber diese „Intelligenz“ hat ihren Preis, nämlich die begrenzte Geschwindigkeit, mit der die Aufgaben erledigt werden. So wären z.B. ein programmierbarer Taktgenerator, ein Logikanalysator oder ein Patterngenerator mit dem BASIC-Tiger® aufgebaut, nur wenig schneller als einige Kilohertz. Kommt es auf genaue Einhaltung der Frequenz an, zeigt sich ein weiteres Problem, der BASIC-Tiger® hat systembedingt bei jeder getakteten Anwendung geringe Abweichungen der einzelnen Takte untereinander. Benötigt man etwas Schnelleres und Präziseres, kostet das entweder viel Geld (Fertiggerät) oder der Bastler löst das Problem mit aufwendiger Hardware, wobei meist der Komfort auf der Strecke bleibt.
Kann man denn nicht den Komfort des BASIC-Tiger®-Systems mit einer schnellen Hardware koppeln? Man kann...
Wir werden in diesem Applikationsbericht Lösungen diskutieren, mit denen auf relativ einfache Weise programmierbare Taktgeneratoren, Logikanalysatoren oder Pattern-generatoren auf Hardwarebasis realisiert werden, die über das BASIC-Tiger®-System komfortabel programmiert werden, dann aber allein arbeiten.
Als erstes konkretes Beispiel, das in einem der nächsten Applikationsberichte behandelt wird, dient ein programmierbarer Frequenzteiler, der eine Eingangsfrequenz (z.B. eines Quarzoszillators) durch eine voreinstellbare Zahl teilt.

2. Das Prinzip

Der Grundbaustein unserer „Zwitterlösung“ ist ein schneller Logikteil, der je nach Voraussetzungen aus Einzelbausteinen in TTL-, HC-, HCT bzw. normaler CMOS-Technologie oder aus einer integrierten PLD-Lösung bestehen kann. Letzteres ist natürlich sehr elegant und auch wesentlich schneller als eine Variante mit herkömmlichen Standardbausteinen, allerdings muß der Anwender einige Voraussetzungen zum Programmieren und Testen solcher moderner Bauelemente haben.
Für die in den nächsten Kapiteln genannten Anwendungsgebiete programmierbarer Taktgenerator, Logikanalysator oder Patterngenerator benötigen wir als Grundelement generell einen schnellen Zähler bzw. Teiler. Je nach Anwendung wird später auch noch ein schnelles statisches RAM eingesetzt.

- Bei dem **programmierbaren Taktgenerator** gibt es nur einen Zähler, der mit einem schnellen externen Takt arbeitet. Die Ausgänge des Zählers werden über einen

Bitvergleicher mit einer statisch anliegenden Information verglichen. Ist die vom Zähler kommende Information identisch mit dem Zählerstand, wird der Zähler rückgesetzt. Ist die statische Information einstellbar, kann ein beliebiges Teilerverhältnis des Zählers eingestellt werden – das bedeutet, die hohe Eingangsfrequenz kann in vielen Stufen untersetzt werden. Wählt man als Eingangsfrequenz z.B. 10 MHz, ist die Originaltaktlänge 100 ns. Daraus können dann alle Taktlängen von 200 ns, 300 ns, 400 ns.... bis zu Sekunden, Minuten und Stunden abgeleitet werden – es kommt nur auf die Länge der Teilerkette an. Der BASIC-Tiger® würde hier nur den Part des „statischen" Vergleichswortes übernehmen. Mit seiner Rechenleistung, seinem Display und einigen Bedienelementen kann er aber auf diese Weise einen sehr einfachen Frequenzgenerator darstellen.

- Im Falle eines **Logikanalysators** werden die Adressen eines schnellen statischen RAM im Schreibbetrieb über unseren Zähler mit dem schnellen Takt hochgezählt, während über die 8 Eingänge (8 Bits) die jeweils anliegende Information eingeschrieben wird. Am Ende des Schreibvorganges wird der schnelle Takt für den Zähler abgeschaltet. Nach dem RESET des Zählers übernimmt der BASIC-Tiger® die Taktung des Zählers – nun ist das RAM im Lesebetrieb und die gespeicherten Informationen werden über die jetzt als Ausgänge geschalteten I/O-Daten-Pins in den BASIC-Tiger® gelesen und ausgewertet. Dabei kann der Takt beliebig langsam werden oder gar in einem Zustand des Zählers stehenbleiben. So ist eine elegante Auswertung von Einzeldaten oder kleinen Bereichen möglich.

- Ähnlich arbeitet der **Patterngenerator**. In umgekehrter Reihenfolge wird zunächst in langsamer BASIC-Tiger®-Taktung das RAM mit Pattern (8-Bit-Worte) unserer Wahl beschrieben. Dazu werden die Informationen aus dem BASIC-Tiger® an die als Eingänge geschalteten Daten-I/Os des RAMs gelegt. Dann wird nach einem RESET der Takt auf schnell geschaltet und alle 8-Bit-Worte in sehr schneller Folge ausgelesen – wobei jetzt die Datenpins des RAMs als Ausgänge arbeiten.

- Nicht zu vergessen sind weitere Lösungen auf dieser Basis – schnelle AD- bzw. DA-Wandler können in Zusammenarbeit mit dem ebenfalls schnellen RAM und seinem Adresszähler zum Oszillografen oder Funktionsgenerator ausgebaut werden. Das Prinzip ist auch hier, nacheinander mit zwei Takten zu arbeiten – einem für die Echtzeitanwendung und einem für die Auswertung.

3. Die Hardware des Zählers / Teilers

3.1. Die herkömmliche Methode – Standardbausteine

Im Gegensatz zu den meisten Applikationsberichten bisher gibt es in diesem Falle kein Kochrezept – lediglich Anregungen für eigene Entwicklungen.

Wie schon gesagt, brauchen wir zunächst einen schnellen Zähler bzw. Teiler mit vielen Stufen. Zwar gibt es solche Zählerbausteine nicht komplett fertig als Standardbausteine, glücklicherweise aber zumindest solche mit mehreren Stufen pro Baustein. Tabelle 1 zeigt eine Auswahl von Standard-Zählern bzw. Untersetzern, die für einen solchen Zweck mehr oder weniger geeignet sind:

Typ	Beschreibung	Max. Frequenz
74LS192	Dekadischer Vor-/Rückwärtszähler 1 Dekade	Ca. 10 MHz
74LS193	Binärer Vor-/Rückwärtszähler 4 Bit	Ca. 10 MHz
CD4029	Synchr. 4-Bit Vor-/Rückwärts-Binärzähler	Ca. 10 MHz
CD4520	2 binäre 4 Bit-Vorwärtszähler	Ca. 10 MHz
74LS292	Programmierbarer Frequenzteiler	30 MHz
74LS294	Programmierbarer Frequenzteiler	30 Mhz
74LS668	Synchr. Vor-/Rückwärts Dekadenzähler	Ca. 30 MHz
74LS669	Synchr. Vor-/Rückwärts-Binärzähler	Ca. 30 MHz
74HCT4020	14-stufiger Binärzähler	

Tabelle 1 Standard-Zähler- bzw. Teilerbausteine in unterschiedlichen Technologien

Wie man aus Tabelle 1 erkennt, dämpfen die erreichbaren Frequenzen und die für einen umfangreichen Zähler erforderlichen Bausteine sowie der Verdrahtungsaufwand ein wenig unsere Euphorie. Die Standardbausteine erreichen maximal einige MHz, will man lange Ketten aufbauen, ist auch noch zu beachten, daß die Verzögerungszeiten der einzelnen Stufen nicht länger werden als der Takt – nicht alle Bausteine eignen sich für eine solche Anwendung. Bild 1 zeigt eine Lösungsvariante für einen programmierbaren Teiler – hier wurden dekadische Zähler vom Typ 74LS192 eingesetzt. Die Programmierung erfolgt über dekadische Vorwahlschalter, die jeweils 4 Bit im BCD-Code einstellen. Dafür werden hier die beim 74LS192 vorhandenen Setzeingänge benutzt, sie setzen den Zähler auf die vorgegebene Dezimalzahl und dann wird rückwärts gezählt. Beim Nulldurchgang erfolgt wieder ein Setzen des gesamten Zählers und so fort. Das Setzen könnte auch der BASIC-Tiger erledigen, dazu können z.B. einfach 24 Outputs des erweiterten I/O-Systems auf dem Plug-and-Play-Lab benutzt werden. Am Ausgang des Teilers haben wir ein Monoflop 74LS123 angeschlossen, das aus den sehr kurzen Setzimpulsen einen Impuls mit definierter Länge macht.

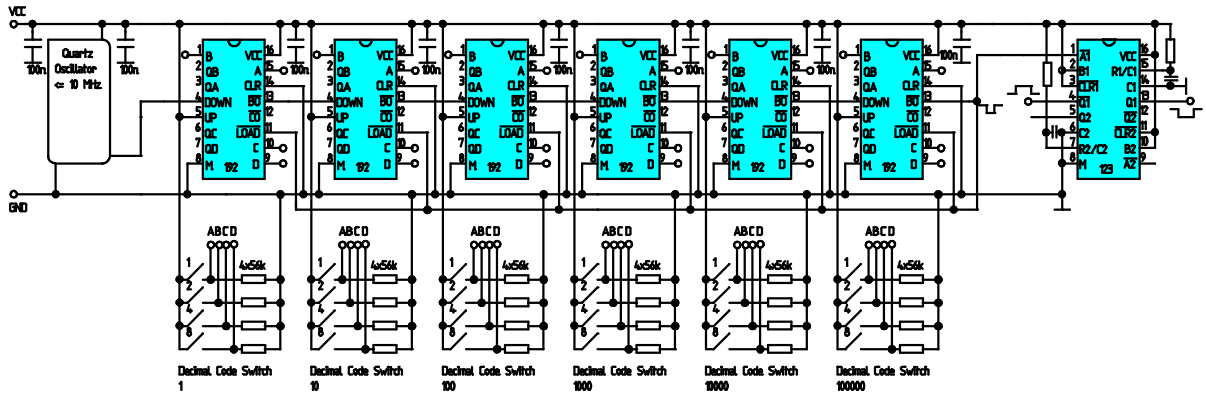

Bild 1 ein mit „diskreten“ dekadischen Zählerbausteinen 74LS192 realisierter programmierbarer Taktgenerator

3.2. Eine elegante Alternative, CPLDs

Bild 1 zeigt eindrucksvoll den Aufwand, der auch bei solchen noch recht einfachen Aufgabenstellungen erforderlich ist. Wesentlich einfacher sind Lösungen, bei denen nicht Standardbausteine benutzt werden, sondern sogenannte CPLDs. Solche CPLDs (Complex Programmable Logic Device) sind Bausteine, bei denen individuelle, teilweise recht umfangreiche Logikfunktionen über einen Programmiervorgang hergestellt werden können. Dieser Programmiervorgang erfolgt beim Anwender, der natürlich über Soft- und Hardware zur Programmierung dieser sonst „nutzlosen“ Bausteine verfügen muß. Wir wollen uns im Folgenden einmal mit einer sehr modernen Lösung auf dieser Basis beschäftigen, der CoolRunner-Familie® von XILINX®.

3.2.1. Die CoolRunner-Familie® von XILINX®

Der Name dieser programmierbaren Logikbausteine soll auf deren wesentlichen Vorzug hinweisen, sie benötigen trotz hoher erreichbarer Geschwindigkeit (Runner) nur wenig Leistung (sie bleiben „cool“). Es gibt ein ganzes Sortiment von Bausteinen, die sich in Anzahl der Makrozellen, Gatterfunktionen und Register unterscheiden. Die erreichbare Geschwindigkeit liegt je nach Ausführung (es gibt Bausteine mit Gatterverzögerungszeiten von ca. 6, 7 und 10 ns) im Bereich > 100MHz (!). Damit sind diese universellen Logikbausteine ideal geeignet als schnelle Ergänzungssysteme für den BASIC-Tiger®, zumal sie für weniger als 5 Euro zu haben sind. Eine umfassende Beschreibung dieser Technik kann natürlich nicht Gegenstand dieses Applikationsberichtes sein, hier sollen nur einige wichtige Grundprinzipien beschrieben werden.
Die Bausteine enthalten als Logiksystem schnelle RAM-Zellen, die über Verbindungszellen so verbunden werden, wie der Anwender das wünscht. Diese Verbindungsstruktur wird in einem internen EEPROM gespeichert und beim Einschalten der Betriebsspannung, die hier übrigens 3,3 V beträgt (!), in die schnelleren RAMs geladen.

Zur Programmierung, die über den PC und dessen parallele Schnittstelle erfolgt, gibt es ein umfangreiches Software-Entwicklungssystem, das übrigens kostenlos bei der Firma Xilinx im Internet heruntergeladen werden kann:
http://www.xilinx.com/products/software/webpowered.htm

Die Schaltung des Programmieradapters von Xilinx gibt es bei:
http://user.cs.tu-berlin.de/~sirhenri/sides/pld_xilinx/jtag.pdf

Tabelle 2 zeigt die derzeit verfügbaren Mitglieder der CoolRunner®-Familie. Für unsere Experimente haben wir den XCR3064XL im PLCC44-Gehäuse ausgewählt.

	XCR3032XL	XCR3064XL	XCR3128XL	XCR3256XL	XCR3384XL	XCR3512XL
Makrozellen	32	64	128	256	512	384
Verfügbare Gatter	800	1600	3200	6400	12800	9600
Register	32	64	128	256	512	384
System-Geschwindigkeit (MHz)	175	145	145	140	127	127
44 Pin PLCC	36*	36*				
44 Pin 1mm VQFP	36*	36*				
48-Pin 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*
256-Pin Fineline BGA				164*	212*	212*
280-Pin 0.8mm CSP				164*		
324-Pin Fineline BGA					220*	260*

* verfügbare I/O-Pins

Tabelle 2 Bauformen der CoolRunner-Familie® von XILINX

Bild 2 Der XCR3064XL im 44-poligen PLCC-Gehäuse

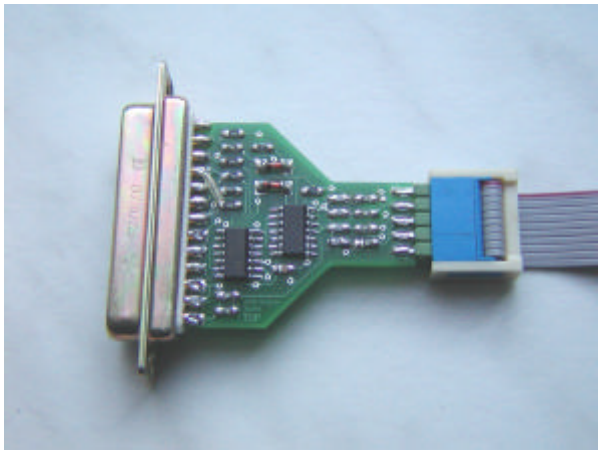

Bild 3 ein selbstgebauter Programmier-adapter

Aus Tabelle 2 entnehmen wir, daß unser XCR3064XL im 44-poligen PLCC-Gehäuse 36 verfügbare I/O-Pins hat, die mit gewissen Einschränkungen frei als Inputs, Outputs oder Clocks verwendet werden können. Der Rest der 44 Anschlüsse wird als GND-, VCC- oder Programmier-Pins benutzt.

Übrigens, obwohl der CoolRunner® mit 3,3 V arbeitet, toleriert er auch 5 V – Eingangssignale, so daß einer direkten Kopplung mit dem BASIC-Tiger® nichts entgegensteht. Umgekehrt akzeptieren die meisten Logikfamilien auch die 3,3 V – Pegel des CoolRunners®.

Für die weiteren Anwendungen ist zusätzlich ein schnelles statisches RAM erforderlich – hier kann man „zum Üben“ erst einmal klein anfangen, z.B. mit einem 6164. Immerhin kann man

so einen Patterngenerator mit 8 kByte realisieren. 8 kByte (genau 8192 oder 2^{13}) brauchen 13 Adressen, d.h. unser Teiler sollte mindestens 13 Stufen haben.

In kommenden Applikationsberichten werden konkrete Projekte mit CoolRunner®-Bausteinen realisiert. Neben den erwähnten Anwendungen mit Zählern denken wir auch an reine Logikerweiterungen, z.B. Erweiterungsbausteine für zusätzliche frei programmierbare I/O-Ports (PIO-Prinzip). Wir werden zu diesen Anwendungen sowohl das jeweilige Datenfile für den „Selbstprogrammierer“ als auch in bestimmten Fällen fertig programmierte Bausteine zur Verfügung stellen. Damit kann auch der interessierte Bastler auf modernste Techniken zurückgreifen und leistungsfähige Baugruppen entwickeln.