Aufgabe 2: Linear rückgekoppelte Schieberegister, ChipScope und Logikanalysator

G. Kemnitz*, TU Clausthal, Institut für Informatik

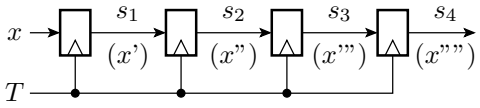
4. Oktober 2012

Zusammenfassung

Ein linearer Automat ist die einfachste sequenzielle Schaltung zur Erzeugung einer periodischen pseudo-zufälligen Bitfolge und ist in der Übungsaufgabe das Testobjekt. Die erzeugte Bitfolge soll als erstes durch Simulation berechnet, als zweites mit einem externen Logikanalysator und als drittes mit einem integrierten Logikanalysator aufgezeichnet, angezeigt und ausgewertet werden.

1 Linear rückgekoppelte Schieberegister

Ein Schieberegister ist eine Kette von flankengesteuerten Speicherzellen, bei dem jede Nachfolgezelle den Wert der Vorgängerzelle übernimmt. Mit jeder aktiven Taktflanke wandert der gespeicherte Bitvektor eine Position weiter. Am Eingang kommt ein neues Bit hinzu und am Ausgang geht ein Bit verloren.



x	s_1	s_2	s_3	s_4
1	0	0	1	1
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Abbildung 1: Schieberegister

Ein linear rückgekoppeltes Schieberegister (LFSR, **l**inear **f**eedback **s**hift **r**egister) ist ein Ringschieberegister, bei dem der Ausgabewert der letzten Speicherzelle zusätzlich modulo-2 zu ausgewählten Bitstellen addiert wird. Die modulo-2 Addition ist die bitweise Addition unter Vernachlässigung des Übertrags und wird mit einem EXOR-Gatter realisiert. Die besondere Eigenschaft eines linear rückgekoppelten Schieberegisters ist, dass es ausgehend von einem Anfangswert zyklisch eine Zustandsfolge durchläuft, die große Ähnlichkeit mit einer Zufallsfolge hat (Abb. 2).

Zur Erzeugung von Pseudo-Zufallsfolgen werden primitiv rückgekoppelte Schieberegister bevorzugt. Ein primitiv rückgekoppeltes Schieberegister ist ein linear rückgekoppeltes Schieberegister, das einen Maximalzyklus der Länge

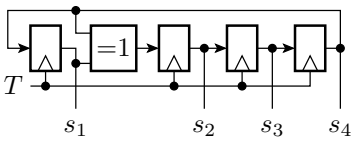
$$Z = 2^r - 1$$

(r – Länge des Schieberegisters) erzeugen kann, der alle Zustände außer »alles Nullen« enthält. Der Nullzustand geht bei jedem linear rückgekoppelten Schieberegister in sich selbst über. Das linear rückgekoppelte 4-Bit-Schieberegister in Abb. 2 hatte z.B. einen Zyklus der Länge

$$Z = 2^4 - 1 = 15$$

und ist somit primitiv rückgekoppelt. Abbildung 3 a zeigt ein primitiv rückgekoppeltes 8-Bit-Schieberegister und Abb. 3 b mögliche primitive Rückführungen für weitere Registerlängen.

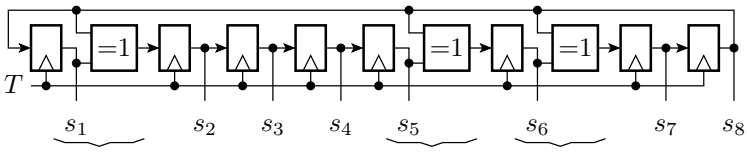
*Tel. 05323/727116



Schritt	s_1	s_2	s_3	s_4
1	1	1	0	1
2	1	0	1	0
3	0	1	0	1
4	1	1	1	0
5	0	1	1	1
6	1	1	1	1
7	1	0	1	1
8	1	0	0	1

Schritt	s_1	s_2	s_3	s_4
9	1	0	0	0
10	0	1	0	0
11	0	0	1	0
12	0	0	0	1
13	1	1	0	0
14	0	1	1	0
15	0	0	1	1
1	1	1	0	1

Abbildung 2: Linear rückgekoppeltes 4-Bit-Schieberegister



RK: 1 5 6 a)

r	RK	r	RK	r	RK	r	RK	r	RK
8	1, 5, 6	28	3	48	1, 27, 28	68	9	88	1, 71, 72
12	3, 4, 7	32	1, 27, 28	52	3	72	6, 47, 53	92	1, 12, 13
16	2, 3, 5	36	11	56	1, 21, 22	76	1, 35, 36	96	2, 47, 49
20	3	40	2, 19, 21	60	1	80	1, 37, 38	100	37
24	1, 3, 4	44	1, 26, 27	64	1, 3, 4	84	13	104	1, 10, 11

b)

Abbildung 3: a) primitiv rückgekoppeltes 8-Bit-Schieberegister b) Rückführstellen für primitiv rückgekoppelte Schieberegister anderer Registerlängen

Die Schaltung eines rückgekoppelten Schieberegisters hat einen Takt- und einen Initialisierungseingang und den Registerzustand, einen Bitvektor der Breite r, als Ausgang. Der Bitvektor mit den Rückführstellen (Bits mit Rückführung sind »1« und die anderen »0«) sei ein Parameter der Entwurfseinheit:

```
entity LFSR is
 generic (
  RK: STD_LOGIC_VECTOR:="0010");
 port (T, I: in STD_LOGIC;
  s: out STD_LOGIC_VECTOR(RK'LENGTH-1 downto 0));
end entity;
```

Mit dem Standardwert »0010« wird das 4-Bit-Register in Abb. 2 beschrieben. Um das 8-Bit-Register in Abb. 3 a nachzubilden, ist bei der Instanziierung der Standardwert des Parameters **RK** wie folgt zu überschreiben:

```
... generic map(RK => "01100010") ...
```

Für das abgetastete Initialisierungssignal und den Zustand sind interne Signale zu vereinbaren. Die Abtastung des Initialisierungssignals erfolgt in einem Abtastprozess, der bei jeder steigenden Taktflanke dem abgetasteten Initialisierungssignal den Wert des unabgetasteten Signals zuweist:

```
architecture a of LFSR is
 signal I_del: STD_LOGIC;
 signal z: STD_LOGIC_VECTOR(RK'LENGTH-1 downto 0);
begin
 process(T)
 begin
```

```
 if RISING_EDGE(T) then I_del <= I; end if;
end process;
```

Die eigentliche Funktion wird durch einen Prozess mit dem abgetasteten Initialisierungssignal und dem Takt in der Weckliste beschrieben, der, wenn das abgetastete Initialisierungssignal »1« ist, an das Zustandssignal **z** den Anfangswert »alles Eins« und sonst bei einer steigenden Taktflanke den um eine Position rotierten Ist-Zustand zuweist. Wenn das höchstwertige Bit im Register eins ist, wird zusätzlich der Rückführvektor **RK** bitweise modulo-2-addiert. Die Zuweisung des Zustands- an das Ausgabesignal erfolgt nach dem Prozess in einer nebenläufigen Signalzuweisung:

```
process(I_del, T)
begin
 if I_del='1' then
  z <= (others =>'1');
 elsif RISING_EDGE(T) then
  if z(z'HIGH)='0' then
   z <= z(z'HIGH-1 downto 0) & z(z'HIGH);
  else
   z <= (z(z'HIGH-1 downto 0) & z(z'HIGH)) xor RK;
  end if;
 end if;
end process;
s <= z;
end architecture;
```

2 Simulation

Die Simulation erfordert einen Testrahmen, der die beiden Eingabesignale T und I erzeugt und das linear rückgekoppelte Schieberegister als Instanz enthält. Dazu werden eine Konstante für die Taktperiode und Signale für die Anschlüsse des Testobjekts vereinbart. Im nachfolgenden Beispiel wird der Standardwert des Parameters **RK** beibehalten, d.h., es wird eine Instanz des rückgekoppelten Schieberegisters in Abb. 2 erzeugt:

```
constant tP: DELAY_LENGTH := 20 ns;
signal I, T: STD_LOGIC;
signal y: STD_LOGIC_VECTOR(3 downto 0);
...
TestObj: entity WORK.LFSR port map(T=>T, I=>I, s=>y);
```

Der nachfolgende Testprozess erzeugt zum Simulationsbeginn einen Initialisierungsimpuls der Breite $2{,}7 \cdot t_\mathrm{P}$ und anschließend bis zum Simulationsende nach 10 µs einen Takt mit der Periode t_P.

```
Testprozess: process
begin
 T<='0'; I<='1', '0' after 2.7*tP;
 loop
  wait for tP/2; T <= not T;
  if NOW > 0.5 us then wait; end if;
 end loop;
end process;
```

Die Beschreibung des Testobjekts, des Testrahmens, einer Kommandodatei zu Simulation mit GHDL und eine sav-Datei sind vorgegeben. Wie in der vorherigen Aufgabe ist das zugehörige Archiv »PrVHDL-A2.zip« in das Arbeitsverzeichnis für das Praktikum zu entpacken. Auch die übrigen Schritte bis zur Durchführung der Simulation und zur Visualisierung der Simulationsergebnisse stimmen mit denen in der Voraufgabe überein.

2.1 Schaltungsentwurf mit ISE

Die Schaltungsbeschreibung »LFSR.vhdl« ist synthesefähig[1]. Damit sie sich besser ausprobieren lässt, soll sie in die Gesamtschaltung in Abb. 4 a eingebettet werden. Als Grundtakt wird der 50-MHz Takt GCLK0 von dem Oszillator auf der Rückseite der Baugruppe genutzt. Der Schalter »SW0« dient als Umschalter zwischen einem schnellen Takt – dem halbierten Grundtakt – und einem langsamen Takt von etwa einem Hertz – dem durch 2^{27} geteilten Grundtakt. Ein Taktsignal muss alle angeschlossen Speicherzellen möglichst zeitgleich erreichen. Dazu gibt es in dem programmierbaren Logikschaltkeis spezielle Taktleitungen, die jeweils von einem BUFG-Treiber angesteuert werden. In der VHDL-Beschreibung ist hinter dem Taktteiler die Schaltungsinstanz für den BUFG-Treiber einzufügen[2]. Das Initialisierungssignal wird von »BTN0« erzeugt. Das Ausgangssignal des rückgekoppelten Schieberegisters wird zum einen auf die Leuchtdioden und zum anderen auf Kontakte der Steckerleiste A2 (gegenüber den Schaltern) geführt. An die Steckkontakte werden die Eingänge des Logikanalysators angeschlossen (Abb. 4).

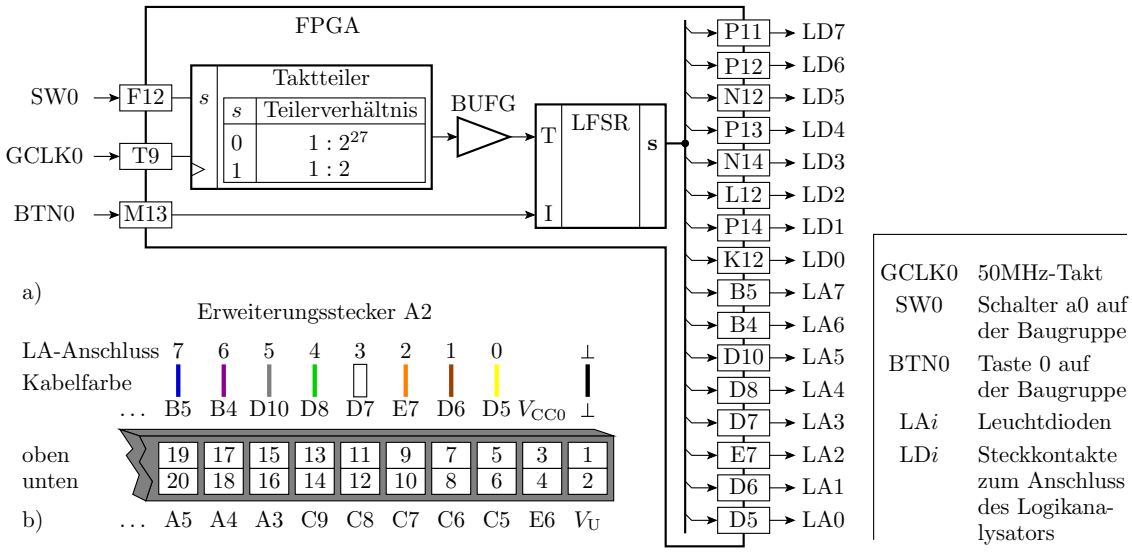


Abbildung 4: a) Gesamtschaltung b) Erweiterungsstecker A2 zum Anschluss des Logikanalysators

Die Schnittstelle der Gesamtschaltung lautet:

```
entity Gesamtschaltung is
 port(GCLK0, SW0, BTN0: in STD_LOGIC;
      LD, LA: out STD_LOGIC_VECTOR(7 downto 0));
end entity;
```

In der Beschreibung sind für das verzweigende Ausgabesignal und den heruntergeteilten Takt vor und nach dem »BUFG« interne Signale zu vereinbaren. Der Taktteiler ist ein Prozess, der bei jeder steigenden Taktflanke eine Zählvariable erhöht. Bei der Schalterstellung »SW0=0« wird der generierte Takt bei jeder steigenden Eingabetaktflanke und sonst nur bei jeder 25.000.000-sten steigenden Eingabetaktflanke invertiert.

```
architecture a of Gesamtschaltung is
 signal y: STD_LOGIC_VECTOR(7 downto 0);
```

[1] Sie enthält keine Verzögerungszeiten, Textausgaben oder andere von der Synthese nicht unterstützten Beschreibungselemente.

[2] Wenn Sie später selbst Schaltungen entwerfen, sollten Sie intern erzeugte Takte auch immer über einen BUFG-Treiber in ein Taktnetz einspeisen. Anderenfalls können durch die sonst größeren Taktversätze schwer zu lokalisierende Fehlfunktionen auftreten.

```
 signal T, TBufG: STD_LOGIC:='0';
begin
 Taktteiler: process(GCLK0)
  variable Ct: NATURAL range 0 to 25000000;
 begin
  Ct := Ct + 1;
  if Ct = Ct'HIGH then
   T <= not T;
  end if;
 end process;
```

Ein wichtiges Detail der Beschreibung des Taktteilers ist, dass der heruntergeteilte Takt ein Bitsignal mit einer Wertezuweisung in einem Abtastprozess ist. Das stellt sicher, dass der Takt in der synthetisierten Schaltung am Ausgang einer Speicherzelle abgegriffen wird, was den Taktversatz zum Basistakt minimiert und Glitches auf dem Taktsignal zuverlässig ausschließt. Auch dieses Beschreibungsdetail sollten Sie später in Ihren eigenen Entwürfen von Takktteilern immer so übernehmen. Der BUFG-Treiber – ein Grundbaustein – ist als »component« zu instanziieren. Die Komponentendefinition befindet sich im Package »UNISIM.VComponents«, das im Vorspann zu importieren ist:[3]

```
Taktnetztreiber: BUFG port map(I=>T, O=>TBufG);
```

Das Testobjekt ist hier das 4-Bit rückgekoppelte Schieberegister aus Abb. 2. Das Ausgabesignal des Schieberegieters ist entsprechend nur 4 Bit breit. Diese werden in der nachfolgenden Beschreibung den Ausgabebits 0 bis 3 zugeordnet. Ausgabebit 7 wird mit dem Takt verbunden und auf den drei übrigen Bits 4 bis 6 wird null ausgegeben.

```
 TestObj: entity WORK.LFSR port map(T=>TBufG, I=>BTN0, s=>y(3 downto 0));
 y(7 downto 4) <= T & "000";
 LD <= y;
 LA <= y;
end architecture;
```

Die VHDL-Beschreibung der Gesamtschaltung und die Projektdatei sind vorgegeben und werden beim Auspacken des Archivs in das Verzeichnis »Aufg2/ISE« kopiert. Die UCF-Datei »Aufg2.ucf_« ist unvollständig. Sie ist vor dem Start von ISE in »Aufg2.ucf« umzubenennen und für einen Teil der Schaltungsanschlüsse sind die Gehäuseanschlüsse entsprechend Abb. 4 a zu ergänzen. Nach dem Start von ISE ist in das Verzeichnis »Aufg2/ISE« zu wechseln, das Projekt »Aufg2.xise« zu öffnen, die Schaltung wie in der ersten Praktikumsanleitung beschrieben zu synthetisieren und die erzeugte Bit-Datei in die Versuchsbaugruppe zu laden.

2.2 Test

Zum Testen der Beispielschaltung ist Schalter »SW0« auf »0« zu stellen (1Hz-Takt). Bei Betätigung von »BTN0« müssen die niederwertigen vier Leuchtdioden angehen und die höchstwertigste Leuchtdiode im Taktrythmus blinken. Nach dem Loslassen der Rücksetztaste wird zyklisch von den vier niederwertigsten Leuchtdioden die erzeugte Pseudo-Zufallsfolge angezeigt.

2.3 Logikanalyse

Bei Umschaltung auf den schnellen Takt arbeitet die Schaltung so schnell, dass nur noch ein gleichmäßiges Leuchten angezeigt wird. Ein Logikanalysator ist ein Gerät, das die logischen Datenfolgen an seinen Eingängen mit einer hohen Geschwindigkeit aufzeichnet. Unser Logikanalysator ist zuerst entsprechend Abb. 4 b an den Erweiterungsstecker A2 anzuschließen.

[3] Die Beschreibungsschablone der Instanziierungsanweisungen findet man unter »Edit ▷ Language ▷ Templates ▷ VHDL ▷ Device Primitive Instantiation ▷ FPGA ▷ Clock Components ▷ Clock Buffers«.

2.3.1 Konfiguration des Logikanalysators

Vor dem Test ist der Logikanalysator mit einer xml-Datei zur konfiguriereren. Die Sample-Rate beschreibt die Anzahl der Abtastwerte pro Sekunde. Zulässig sind Werte zwischen 1 und 150.000.000 sowie 600.000.000. Für die Signalaufzeichnung mit dem schnellen Takt sind 600 Millionen Abtastwerte pro Sekunde ein geeigneter Wert:

```
<la>
 <samplerate>600000000</samplerate>
```

Der zu verwendende Logikanalysator zeichnet 4048 Werte auf. Die gesamte Aufzeichnungsdauer ist entsprechend:

$$t_{\text{Aufzeichnung}} = \frac{4048}{600.000.000\,\text{s}^{-1}} \approx 6{,}7\,\mu\text{s}$$

Bei 25 Takten pro Mikrosekunde werden knapp 170 Taktperioden aufgezeichnet. Um die Signale für das richtige Zeitfenster aufzuzeichnen, müssen Trigger und Pre-Trigger-Wert richtig eingestellt sein. Der Trigger beschreibt eine Signalbedingung, an der der Bezugszeitpunkt für die Signalaufzeichnung erkannt wird. Für die Triggereinstellungen hat der zu verwendende Logikanalysators zwei Hilfsvariablen »A« und »B«, die aus einer UND-Verknüpfung von Bitbedingungen gebildet werden. Bitbedingung kann Signalwert »0« oder »1«, steigende oder fallende Flanke sein. Für den Gesamt-Trigger lässt sich einstellen, das er ausgelöst wird, wenn »A=1«, »B=1«, »A∨B=1« etc. ist. Im nachfolgenden Beispiel muss »A=1« erfüllt sein, wobei »A« die UND-Verknüpfung der Bedingungen »1 an Dateneingang 0« und »0 an den Dateneingängen 1 bis 3« ist (weitere Details siehe Kurzdokumentation zum USB-LOGI-500 auf der Web-Seite):

```
 <trigger when="A">
  <A>
   <ch when="high">0</ch>
   <ch when="low" >1</ch>
   <ch when="low" >2</ch>
   <ch when="low" >3</ch>
  </A>
 </trigger>
</la>
```

Der Pre-Trigger-Wert legt den relativen Zeitanteil des dargestellten Zeitfensters vor dem Triggerzeitpunkt fest (Abb. 5). Zulässige Werte sind 1 bis 7 für 1/8 bis 7/8 des Darstellungsfensters. Im Beispiel ist 1/8 eingestellt:

```
<pretrigger>1</pretrigger>
```

Nach Start des Logikanalysators (siehe Folgeabschnitt) wird gewartet, bis die erforderliche Mindestanzahl von Pre-Trigger-Werten aufgezeichnet ist. Dann wird der Aufzeichnungsspeicher weiter zirkular gefüllt, bis das Trigger-Ereignis eintritt. Abschließend werden die nach dem Trigger-Ereignis darzustellenden Werte aufgezeichnet und die aufgezeichneten Signalwert angezeigt.

Die Signaldefinitionen legen die Namen und Kanalnummern der anzuzeigenden Signalwerte fest. Signale mit mehreren Kanälen, in der nachfolgenden Beschreibung das Signal »y« mit den Kanälen »0« bis »3«, beschreiben Bitvektoren:

```
<signals>
 <signal name="Takt">
  <ch>2</ch>
 </signal>
 <signal name="y">
  <ch>0</ch>
  <ch>1</ch>
  <ch>2</ch>
  <ch>3</ch>
 </signal>
</signals>
```

Die Kanalnummern stehen auf dem Logikanalysatorgehäuse und auf den Isolierschläuchen an den Kabelenden.

2.3.2 Datenaufzeichnung und Darstellung

Die im Vorabschnitt beschriebene Konfigurationsdatei wird beim Entpacken der Zip-Datei in das Verzeichnis »Aufg2/LA« kopiert und heißt »ConfigLA_Aufg2.xml«. In einem Terminal ist

- in dieses Verzeichnis zu wechseln
- auf der Baugruppe mit »SW0=1« der schnelle Takt auszuwählen und
- die Aufzeichnung mit

  ```
  usb-logi ConfigLA_Aufg2.xml
  ```

  zu starten.

Der Kommandoaufruf erzeugt eine lxt- und eine sav-Datei, mit denen automatisch GTKWAVE zur Darstellung der Messwerte aufgerufen wird. Abbildung 5 zeigt die Messwerte mit den Beispieldaten. Zur Wiederholung der Messung mit dem langsamen Takt sind mit dem Schalter »SW0« der Takt umzuschalten, die Anzahl der Abtastwerte pro Sekunde auf etwa 100 zu verringern und die Messung neu zu starten.

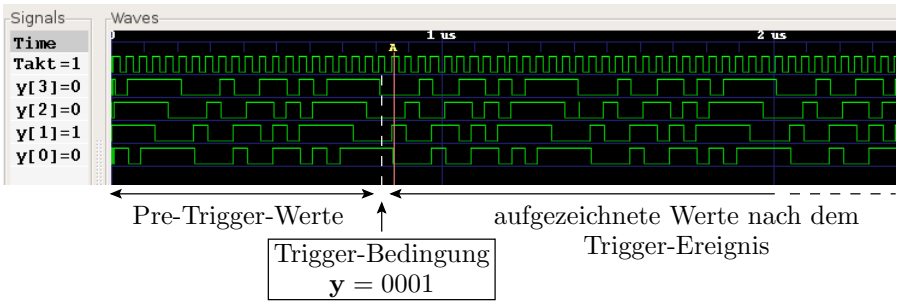


Abbildung 5: Aufzeichnungsergebnis des Logikanalysators für den schnellen Takt

3 ChipScope

Alternativ zum externen Logikanalysator kann der Logikanalysator auch mit in den FPGA programmiert werden. Für die Erzeugung einer Logikanalysatorschaltung besitzt das Entwurfssystem ISE einen Schaltungsgenerator, der aus einer Parameterbeschreibung die Schaltung generiert. Im Beispiel sei der Aufzeichnungstakt des integrierten Logikanalysators (ILA) der 50-MHz-Eingabetakt »GCLK«. Er soll fünf Dateneingänge haben, die den Zustand des rückgekoppelten Schieberegisters $\mathbf{z}$ und den intern erzeugten Takt T mit der steigenden Flanke von »GCLK« aufzeichnen. Alle fünf Signale sollen auch für den Trigger auswertbar sein, wobei die einfachste Form der Triggerung, der Vergleich mit vorgegebenen Signalwerten, genügt (Abb. 6). Die Teststeuerung und das Auslesen der aufgezeichneten Daten erfolgt über das Programmierkabel und ein spezielles auf dem PC laufendes Programm (ChipScope).

Zur Konfiguration des integrierten Logikanalysators ist in ISE in »Hierarchy« eine neue Quelle vom Typ »ChipScope Configuration file« wie folgt zu erzeugen:

- »rechter Mausklick« ▷ »New Source« ▷ Dateiname: »ChipScope«, Source Type: »Chip Scope Definition and Connection File«

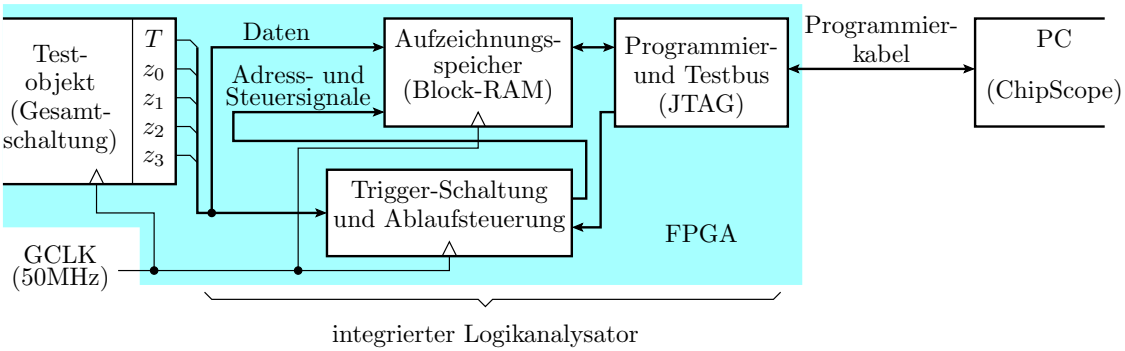


Abbildung 6: Test der Beispielschaltung mit einem integrierten Logikanalysator

- »Next« ▷ »associated to Gesamtschaltung« ▷ »Next« ▷ »Finish«

Die neue Projektdatei »ChipScope.cdc« ist mit einem Mausklick zu öffnen. In dem aufgehenden Fenster ist in »Trigger Parameters« für die Eingangsanzahl »5«, »Match Type Basic«, eine »Match Unit«, kein Zähler und kein »Trigger Sequencer« einzustellen. Links im Fenster »Core Utilization« wird jeweils die erforderliche Hardware in »Look-Up-Tabellen, Flipflops und Block-Rams angezeigt (Abb. 7). Jede Erweiterung der Trigger-Möglichkeiten, die für die Einstellung der Aufzeichnungsfensters zur Verfügung gestellt werden soll, kostet zusätzliche Hardware, die dann nicht mehr für das Testobjekt zur Verfügung steht. In unserem Beispiel ist der Schaltkreis fast leer, so dass es nicht stören würde, einen »Trigger Sequencer« für verkettete Trigger-Bedingungen, mehrere »Match Units« etc. vorzusehen. Nur werden diese Zusatzfunktionen für eine so einfach zu testende Schaltung nicht benötigt.

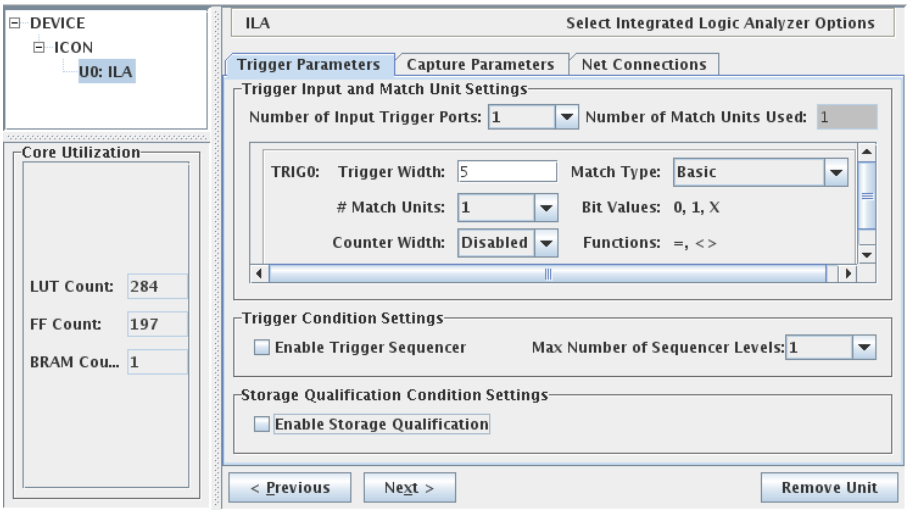


Abbildung 7: Triggerparameter für den integrierten Logikanalysator

Im Menü »Capture Parameters« soll für die Tiefe des Aufzeichnungsspeichers 1024, für die aufzuzeichnenden Datenkanäle »Data Same As Trigger« und für die Aufzeichnungstaktflanke »Rising« eingestellt werden. Im Menü »Net Connections« ist mit »Modify Connections« das Menü zur Zuordnung der Anschlusssignale für den Logikanalysator zu öffnen. Der Aufzeichnungstakt sei wie in Abb. 6 festgelegt der 50MHz-Eingabetakt »GCLK«. Intern ist allerdings nur das Ausgabesignal des zugehörigen Takttreibers »GCLK_BUFGP«, den das Entwurfssystem automatisch einfügt, verfügbar (Abb. 8 a). An die Dateneingänge sind gleichfalls, wie in Abb. 6 gezeigt, der halbierte Takt T und das 4-Bit-Zustandssignal des rückgekoppelten Schieberegisters anzuschließen

(Abb. 8 b).

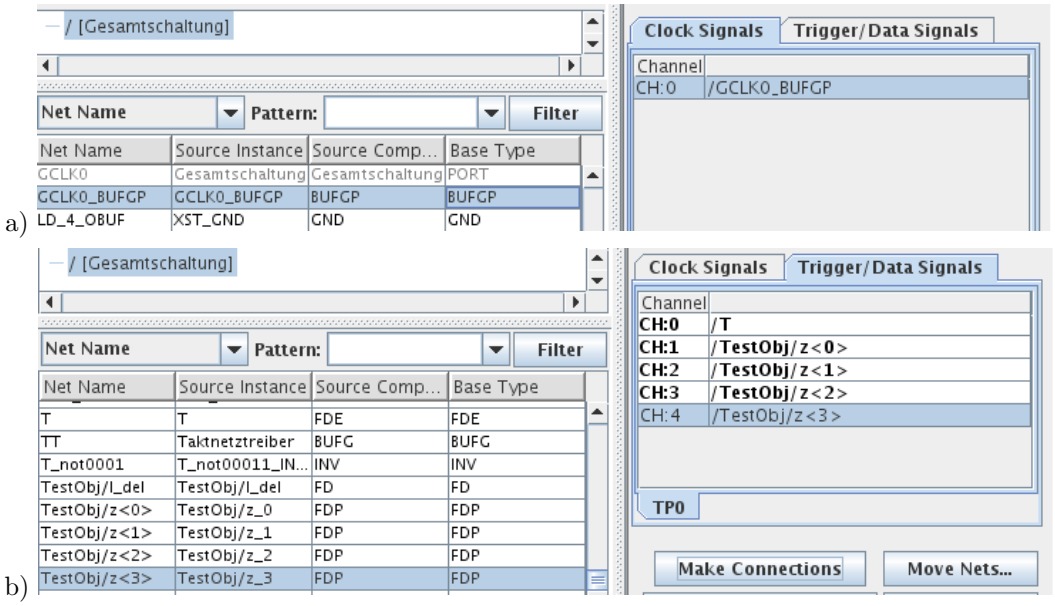


Abbildung 8: a) Zuordnung des Aufzeichnungstakts b) Zuordnung der aufzuzeichnenden Signale

Die Bearbeitung der ChipScope-Konfigurationsparameter ist abzuschließen:

- »OK« ▷ »Return to Project Navigator« ▷ »Ja«

Dann ist im Fenster »Hierachy« die Gesamtschaltung und im zugehörigen Werkzeugfenster »Analyze Design Using Chipscope« auszuwählen. Dieser Befehl startet den Syntheseprozess für die gesamte Schaltung incl. Logikanalysator, gefolgt von der Platzierung, Verdrahtung etc. bis zum Öffnen des Programms »ChipScope«, mit dem die Bitdatei für die Schaltung in den Schaltkreis geladen, der Trigger-Wert eingestellt, die Logikanalyse gestartet, die aufgezeichneten Daten gelesen und angezeigt werden.

Wenn sich das Fenster »ChipScope Pro Analyzer« geöffnet hat, sind

- falls noch nicht erfolgt, die Versorgungsspannung und das Programmierkabel an die Baugruppe anzuschließen und
- mit einem Mausklick auf das Kettensymbol oben links im Fenster die Verbindung mit dem Programmierkabel herzustellen[4].
- Fenster mit den gefundenen Schaltkreisen mit »OK« wegklicken.

Bei erfolgreicher Herstellung der Verbindung werden die gefundenen Schaltkreise am Testbus angezeigt (Abb. 9 links oben). Dem ersten Schaltkreis in der Kette »Dev:0« ist die erzeugte Bitdatei zuzuordnen. Daraufhin wird der Schaltkreis programmiert und im Objektbaum als Unterobjekt des Schaltkreises der integierte Logikanalysator (ILA) angezeigt.

Vor der Aufzeichnung sind der Trigger- und der Pre-Trigger-Wert (Eingabefeld »Position«) einzustellen. In Abb. 10 ist als Trigger-Ereignis das erstmalige Auftreten von $T = 1$ und $\mathbf{z} =$ "0010" nach dem Aufzeichnungsstart und Anzeige von 100 aufgezeichneten Werten vor dem Trigger-Zeitpunkt eingestellt. Eine Messung wird mit einem Mausklick auf das Dreieck in der oberen Menüleiste gestartet.

[4]Bei der Fehlermeldung »Cable is locked ...« hat sich bereits ein Programm den Kabeltreiber reserviert und noch nicht freigegeben. ChipScope hat leider in der aktuellen Version den Fehler, beim Beenden den Lock-Eintrag nicht zu löschen. Bis eine bessere Lösung gefunden ist, hilft in dieser Situation nur ein Neustart von Linux. Der Fehler existiert in der neuen Version oder unter Windows möglicherweise nicht mehr.

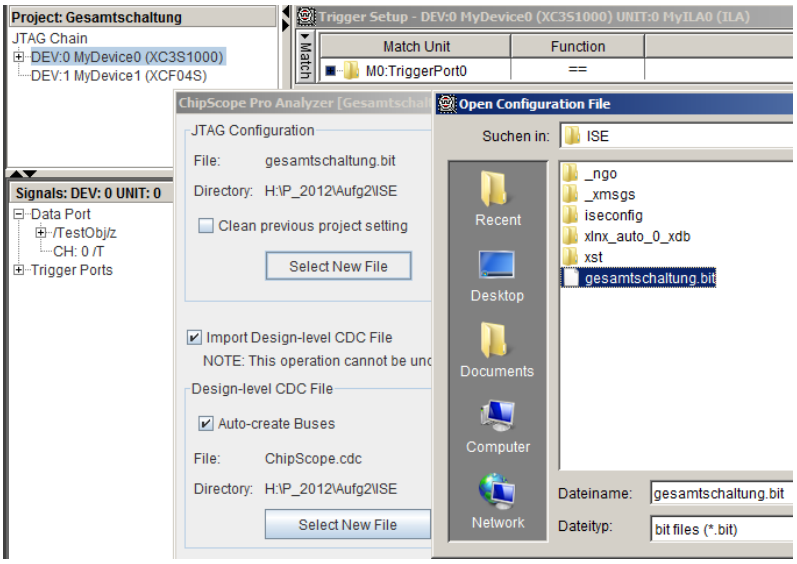


Abbildung 9: Einstellungen unter ChipScope

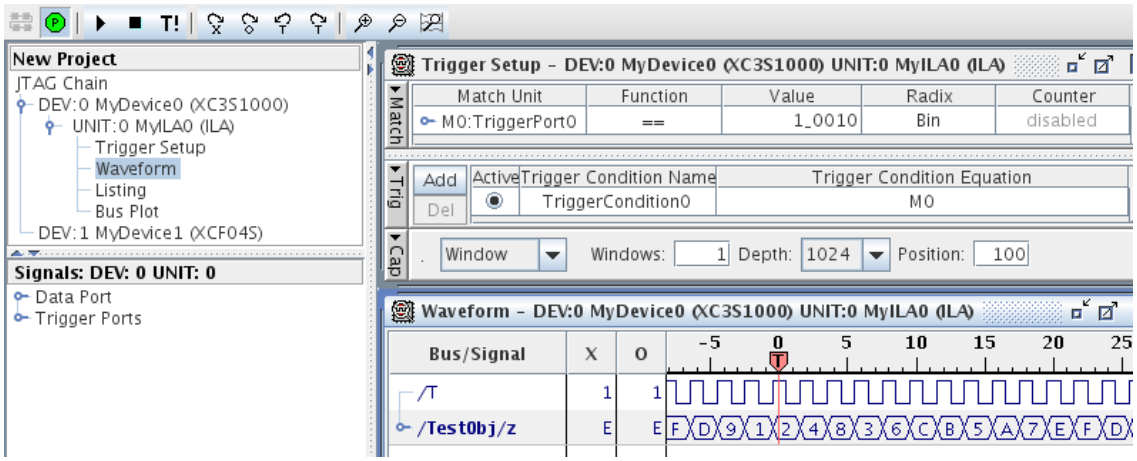


Abbildung 10: Aufzeichnungsergebnis des Logikanalysators für den schnellen Takt

4 Aufgaben

In diesem Praktikum sollen Sie vor allem die unterschiedlichen Möglichkeiten der Kontrolle der Schaltungsfunktion für den Test und die Fehlersuche ausprobieren. Denn wie bei der Software-Entwicklung kosten bei einem Hardware-Entwurf der Test und die Fehlersuche den größten Teil der Entwurfszeit.

1. Führen Sie die Simulation sowie den Test mit dem externen Logikanalysator und dem integrierten Logikanalysator aus. Verwenden Sie die vorgegebenen Beispieldateien aus der Zip-Datei auf der Web-Seite.

2. Entwerfen Sie an Anlehnung an das Beispieltestobjekt ein 5- oder 6-Bit-rückgekoppeltes Schieberegister mit selbst gewählten Rückführstellen und zeichnen Sie die Schaltung auf das Abgabeblatt.

3. Simulieren Sie die geänderte Schaltung. Bestimmen Sie die Zykluslänge der Zustandsfolge vom Anfangswert bis wieder der Anfangswert erreicht ist. Notieren Sie das Ergebnis auf dem

Abgabeblatt und bewahren Sie die GHW- und SAV-Datei zur Abnahmevorführung auf.

4. Synthetisieren Sie die geänderte Schaltung und versuchen Sie mit dem Logikanalysator dieselben Signaldarstellungen zu erzeugen wie bei der Simulation. Dazu sind auch die UCF-Datei, die Konfigurationsdatei des Logikanalysators und die Drahtverbindungen zum Logikanalysators anzupassen. Bewahren Sie auch hier die GHW- und SAV-Datei zur Abnahmevorführung auf.

5. Passen Sie als nächstes den integrierten Logikanalysator an die geänderte Schaltung an und wiederholen Sie auch den Test damit. Notieren Sie die vorgenommenen Änderungen und heben Sie ein Bildschirmfoto der Triggereinstellungen und Signalverläufe für die Abnahme auf.

Empfehlung (keine Pflichtaufgabe):

- Experimentieren Sie sowohl bei dem Logikanalysator als auch bei dem integrierten Logikanalysator mit unterschiedlichen Trigger-Einstellung.
- Schauen Sie sich wie bei der ersten Aufgabe mit

  – »Sythesize XST« ▷ »View RTL Schematic«

  das Syntheseergebnis und mit

  – »Place & Route« ▷ »Analyze Timing / Floorplan Design ...«

  die automatisch erzeugte Schaltungsanordnung an.
- Führen Sie eine »Post Place & Route«-Simulation der Schaltung durch und vergleichen Sie die berechneten Zeitverläufe für die Schaltkreisausgänge, an denen der Logikanalysator angeschlossen ist, mit den Signalverläufen, die der Logikanalysator aufzeichnet.

5 Fragen zur Selbstkontrolle

- Mit welchen Werten darf ein primitiv rückgekoppeltes Schieberegister initialisiert werden, so dass nach der Initialisierung $2^r - 1$ Zustände zyklisch durchlaufen werden?
- Welche Frequenz hat ein Takt mit einer Periode von 20 ns?
- Der externe Logikanalysator zeichnet 600 Millionen Werte pro Sekunde und der integrierte Logikanalysator 50 Millionen Werte pro Sekunde auf. Wie viel mal wird in beiden Fällen jede Periode des 25-MHz-Takts T abgetastet?

6 Abnahmekriterien

- Zeichnung des gewählten rückgekoppelten Schieberegisters
- Bestimmung der Zyklusperiode und Erklärung, wie der Wert anhand der Signalaufzeichnung mit einem Logikanalysator oder anhand des Simulationergebnisses bestimmt wurde.
- Stichprobenweise Vorführung weiterer bereitzuhaltender Signalverläufe.
- Stichprobenweise Kontrolle der von Ihnen geänderten VHDL-Dateien, der vervollständigten UCF-Datei, der geänderten Konfigurationsdatei für den externen Logikanalysator oder der geänderten Konfigurationsdatei des integrierten Logikanalysators.
- Stichprobenweise Beantwortung der Fragen zur Selbstkontrolle.

7 Aufräumen

Damit beim Ein- und Ausloggen nicht mehrere Minuten Daten transferiert werden, neu generierbaren Entwurfsdateien löschen:

≫Project≪ ▷ ≫Cleanup Project Files≪