Entwurf digitaler Systeme

Labor

Ausgabe 1.0, 09.01.2017
Autor: Klaus Gosger

Inhaltsverzeichnis

Versuch 1 - Schachuhr 4
1.1 Konzeption der Schaltung 5
1.2 Schaltungsentwurf 6
1.3. Funktionale Verifikation 6
1.4. Schaltungssynthese und Implementierung 7
Versuch 2 - freies Thema 8
2.1. Funktionale Beschreibung 8
2.2. Schaltungsentwurf 8
2.3. Funktionale Verifikation 8
2.4. Schaltungssynthese 8
3. Zielsystem 9
Anhang 1: Lösungen zu den Fragen und Beispiele für VHDL 11
A.1 Schachuhr 11
Anhang 2: Literatur 17

Versuch 1 - Schachuhr

Mit Hilfe einer Baugruppe zur Evaluierung soll eine Schachuhr auf einem FPGA (Field Programmable Gate Array) realisiert werden. Folgende Abbildung zeigt die Baugruppe, sowie die Beschaltung des FPGAs. Zum Einsatz kommt das Basys-2-Board der Firma Digilent mit dem FPGA Spartan3E-100 CP132 der Firma Xilinx.

- 100,000-gate Xilinx Spartan 3E FPGA
- Atmel AT90USB2 Full-speed USB2 port providing board power and programming/data transfer interface
- Xilinx Platform Flash ROM to store FPGA configurations
- 8 LEDs, 4-digit 7-segment display, 4 buttons, 8 slide switches
- PS/2 port and 8-bit VGA port
- User-settable clock (25/50/100MHz), plus socket for 2nd clock
- Four 6-pin header expansion connectors
- ESD and short-circuit protection on all I/O signals.

Bild 1: Basys 2 Board: Blockdiagram und Leistungsmerkmale

Die Schachuhr soll folgende Funktionen implementieren:

(1) Umschalten von Spieler A auf Spieler B bzw. zurück nach jedem Schachzug (durch den am Zug befindlichen Spieler),

(2) Anzeige der insgesamt benötigten Zeit für den am Zug befindlichen Spieler in Minuten und Sekunden,

(3) Anhalten und Fortsetzen der Uhr bei Spielunterbrechungen (Start/Stopp-Funktion),

(4) Reset-Funktion.

Zur Anzeige sollen die auf der Baugruppe befindlichen Bauteile verwendet werden (7-Segmentanzeige mit 4 Stellen, Schalter bzw. Taster, 50 MHz Taktsignal).

1.1 Konzeption der Schaltung

Aufgabe 1.1: Zeichnen Sie ein Blockdiagramm der Schachuhr, das in einer HDL-Beschreibung die Entity darstellt.

Aufgabe 1.2 Erstellen Sie ein Konzept bzw. grobes Blockschaltbild für das Innenleben: Welche Funktionen bzw. Komponenten benötigen Sie (z.B. Sekundentakt, Speicher bzw. Zähler für die Spielzeiten, Steuerung durch Tasten, Anzeige etc)?
Hinweis: Arbeiten Sie vom Groben ins Feine (erst die Struktur, die Details später). Ignorieren Sie zunächst bitte Reset und Start/Stopp. Abstrahieren Sie die Ansteuerung der 7-Segmentanzeige zu einem Funktionsblock Anzeige, der später detailliert werden kann.

Aufgabe 1.3: Ergänzen Sie das Blockdiagramm um die Funktionen Reset und Start/Stopp. Begründen Sie Ihren Entwurf.

Aufgabe 1.4: Detaillieren Sie die Speicher bzw. Zähler für die Spielstände.

Aufgabe 1.5: Detaillieren Sie den Funktionsblock für die 7-Segmentanzeige so, dass hervorgeht, wie Sie die 4 Ziffern der Sieben-Segment-Anzeigen ansteuern.

Hinweise über die Verwendung der 7-Segmentanzeige finden Sie im Reference Manual zum Basys 2 Board. Folgendes Prinzip kommt zur Anwendung:

Bild 2: Basys 2 Board: Verschaltung 7-Segment -Anzeige

Die 4 Anzeigen (Ziffern) teilen sich die Kathoden-Anschlüsse, die Anoden des jeweiligen Segmentes sind zusammen geschaltet und dienen somit quasi zur Auswahl des jeweiligen Segments. Das folgenden Timing-Diagramm gibt genaueren Einblick in die Funktionsweise:

Bild 3: Basys 2 Board: Timing für gemultiplexete 7-Segment-Anzeigen

1.2 Schaltungsentwurf

Die in der Konzeption festgelegten Funktionsblöcke sind nun als Schaltung zu entwerfen, d.h. in VHDL zu beschreiben. Für den Entwurf können sie die Muster aus den angehängten Lösungen als Basis verwenden.

Hinweis: Wenn Sie in zu zweit bzw. in Gruppen organisiert sind, können Sie die Aufgaben folgendermaßen aufteilen und **parallel** starten:

(1) Schaltungsentwurf und funktionale Verifikation: machen Sie sich mit der Funktion der Schaltung und der Simulation der Schaltung vertraut, damit Sie auf Tests und mögliche Fehler bei der Implementierung vorbereitet sind.

(2) Schaltungssynthese: Machen Sie sich mit den Synthesewerkzeugen vertraut. Verwenden Sie hierzu die vorgegebenen Muster als Dateivorlagen für den Umgang mit den Werkzeugen zur Synthese und Implementierung

Aufgabe 1.6: Entwerfen Sie die Schaltung in VHDL. Als Vorlage können Sie die HDL-Beschreibung im Skript verwenden. Analysieren Sie die Vorlage und **ändern** Sie gegebenenfalls nach Bedarf.

1.3. Funktionale Verifikation

Übernehmen Sie die Schaltungsentwürfe für den Generator in den HDL-Simulator und übersetzen Sie den Schaltungsentwurf. Übersetzen Sie die Dateien zunächst einzeln. Arbeiten Sie bitte direkt auf dem Verzeichnis, auf dem Sie die VHDL Dateien abgelegt haben, so dass bei der Fehlerkorrektur dort der fehlerfreie Quellcode abgelegt bleibt.

Aufgabe 1.7: Übersetzen Sie den Schaltungsentwurf und beheben Sie alle ggf. auftretenden Fehler, die der Compiler meldet.

Überlegen Sie sich eine Methode, um den Schaltungsentwurf zu verifizieren. Hierzu benötigen Sie eine Testumgebung, die alle Eingangssignale Ihrer Schaltung stimuliert und die Ausgangssignale aufnimmt. Hinweis: eventuell ist es hilfreich, die Signale im Prüfling zum Test hinzuzuziehen.

Aufgabe 1.8: Entwerfen Sie eine Testumgebung für ihre Schaltung.

Aufgabe 1.9: Testen Sie das Modul im Simulator. Analysieren Sie die Ausgangssignale des Prüflings (DUT). Prüfen Sie die Ergebnisse auf Plausibilität. Machen Sie notwendige Änderungen und wiederholen die Aufgaben 1.6 bis 1.9. bis sie einen verifizierten Schaltungsentwurf haben.

Aufgabe 1.10: Dokumentieren Sie die Ergebnisse Ihrer Tests. Legen Sie diese Dokumentation Ihrem Laborbericht bei.

1.4. Schaltungssynthese und Implementierung

Setzen Sie sich mit den Werkzeugen zur Synthese und Implementierung auseinander. Stellen Sie den Zusammenhang mit dem in der Abbildung gezeigten grundsätzlichen Ablauf her.

Bild 4: Synthese und Implementierung des Schaltungsentwurfs

Aufgabe 1.11: Machen Sie sich mit den Synthesewerkzeugen (ISE Webpack und den dazugehörigen „Unterprogrammen“) vertraut. Verwenden Sie hierzu die vorgegebenen Muster als Dateivorlagen für den Umgang mit den Werkzeugen zur Synthese und Implementierung.

Aufgabe 1.12: Überprüfen Sie die Ergebnisse der Verifikation des Schaltungsentwurfs (Dokumentation zu Aufgabe 1.9). Synthetisieren und implementieren Sie die den Schaltungsentwurf und testen Sie seine Funktionen auf dem Basys 2 Board. Wiederholen Sie gegebenfalls Schritte, bis die Schaltung das gewünschte Verhalten zeigt.

Aufgabe 1.13: Dokumentieren Sie die Ergebnisse. Legen Sie diese Dokumentation Ihrem Laborbericht bei.

Versuch 2 - freies Thema

Als weiterer Versuch ist Ihnen ein Thema freigestellt. Ausgehend von Versuch 1 (Schachuhr), kämen beispielsweise folgende Abwandlungen / Ergänzungen in Frage: (1) Stoppuhr / Eieruhr oder (2) ein Metronom (Anzeige „beats per minute“) usw...

Ihr Thema kann aber auch in eine völlig andere Richtung gehen, z.B. ein Signalgenerator (siehe Vorlesungsmanuskript) oder ein VGA Schnittstellen -Treiber oder ein Lauflicht / Laufschriftleiste...

Die Art des Themas richtet sich nach Ihren eigenen Interessen, sowie nach den Möglichkeiten der verwendeten Entwicklungsumgebung und des Zielsystems. Beispielsweise können Sie sich nach frei verfügbaren Bausteinen (IP-Cores umschauen), bzw. ein komplett eigenes Design realisieren.

Gehen Sie bitte wie folgt vor:

- Wenn möglich, wählen Sie sich einen Sparringspartner in Ihrem Kurs, mit dem Sie Ihr Thema gemeinsam realisieren. Auf diese Weise können Sie Arbeiten aufteilen und den Fortschritt und das weitere Vorgehen miteinander diskutieren. Da die Synthesewerkzeuge recht komplex sind, ist beispielsweise eine Arbeitsaufteilung in (1) Entwurf und Test der Schaltung, sowie (2) Synthese und Implementierung denkbar.
- Besprechen Sie Ihren Themenvorschlag mit Ihrem Betreuer. Bei dieser Gelegenheit ist auch zu klären, ob Sie zusätzliche Hardware benötigen und anschaffen möchten.
- Realisieren Sie Ihren Systementwurf in den in den folgenden Kapiteln angegebenen Schritten.
- Dokumentieren Sie die wichtigsten Ergebnisse in jedem der Schritte.
- Präsentieren Sie Ihren Systementwurf in der Abschlusspräsentation vor den anderen Teilnehmern des Kurses.

Das Testat für das Labor erhalten Sie nach erfolgreicher Abschlusspräsentation und Abgabe der Dokumentation (Laborbericht) bei Ihrem Betreuer.

2.1. Funktionale Beschreibung

Beschreiben Sie Ihr Vorhaben und das Funktionsprinzip. Als Muster für die Dokumentation können Sie die Pflichtaufgabe verwenden, bzw. auf Literatur oder Fundstellen im Web verweisen. Beschreiben Sie Ihre geplante Vorgehensweise und ggf. die Aufteilung der Aufgaben.

2.2. Schaltungsentwurf

Entwerfen Sie Ihre Schaltung mit VHDL. Dokumentieren Sie Ihre Ergebnisse.

2.3. Funktionale Verifikation

Schreiben Sie eine Testumgebung zu Ihrem Schaltungsentwurf und testen Sie die Schaltung. Dokumentieren Sie Ihre Ergebnisse.

2.4. Schaltungssynthese

Implementieren Sie Ihren Schaltungsentwurf auf der Ziel-Hardware mit Hilfe der Entwicklungsumgebung. Dokumentieren Sie Ihre Ergebnisse.

3. Zielsystem

Als Zielsystem kommt ein Evaluations-Board von Digilent zum Einsatz. Dieses Board bietet bereits umfangreiche Schnittstellen auf der Leiterkarte, die Sie in ihrem Versuch verwenden können.

- 100,000-gate Xilinx Spartan 3E FPGA
- Atmel AT90USB2 Full-speed USB2 port providing board power and programming/data transfer interface
- Xilinx Platform Flash ROM to store FPGA configurations
- 8 LEDs, 4-digit 7-segment display, 4 buttons, 8 slide switches
- PS/2 port and 8-bit VGA port
- User-settable clock (25/50/100MHz), plus socket for 2nd clock
- Four 6-pin header expansion connectors
- ESD and short-circuit protection on all I/O signals.

Bild 5: Basys 2 Board: Blockdiagram und Leistungsmerkmale

Außer dem FPGA-Baustein (Xilinx Spartan-3E-100) und der zugehörigen fest eingebauten Peripherie (Arbeitsspeicher, Flash-Speicher, LED- und 7-Segment-Anzeigen, Mikroschalter, Taster und Taktgenerator) sind folgende Schnittstellen zur Erweiterung bzw. zur Kommunikation vorhanden:

- Steckplätze für vier I/O Module (Expansion I/O für Peripherie-Module PMOD)
- USB-Port
- PS/2 Port und VGA Port

Der USB-Port dient der Kommunikation mit einem Rechner. Hierunter fällt auch die Programmierschnittstelle für die Entwicklungswerkzeuge. Ebenfalls über den USB-Port erfolgt die Stromversorgung. Für Netzwerkverbindungen steht ein Ethernet-Port zur Verfügung.

Programmiert wird das Board mit der SW Adept von Digilent, siehe Literaturhinweise. Beachten sie die richtige Stellung des Jumpers (auf PC).

Bild 6: Basys 2 Board: Programmierung

Die Peripherie ist an folgende FPGA Pins angeschlossen (siehe Reference Manual):

FPGA pin definition table color key		
Grey		Not available to user
Green		User I/O devices
Yellow		Data ports
Tan		Pmod port signals
Blue		USB signals

Basys 2 Spartan-3E pin definitions											
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
C12	JD1	P11	SW0	N14	CC	B2	JA1	P8	MODE0	M7	GND
A13	JD2	M2	USB-DB1	N13	DP	C2	USB-WRITE	N7	MODE1	P5	GND
A12	NC	N2	USB-DB0	M13	AN2	C3	PS2D	N6	MODE2	P10	GND
B12	NC	M9	NC	M12	CG	D1	NC	N12	CCLK	P14	GND
B11	NC	N9	NC	L14	CA	D2	USB-WAIT	P13	DONE	A6	VDDO-3
C11	BTN1	M10	NC	L13	CF	L2	USB-DB4	A1	PROG	B10	VDDO-3
C6	JB1	N10	NC	F13	RED2	L1	USB-DB3	N8	DIN	E13	VDDO-3
B6	JB2	M11	LD1	F14	GRN0	M1	USB-DB2	N1	INIT	M14	VDDO-3
C5	JB3	N11	CD	D12	JD4	L3	SW1	P1	NC	P3	VDDO-3
B5	JA4	P12	CE	D13	RED1	E2	SW6	B3	GND	M8	VDDO-3
C4	NC	N3	SW7	C13	JD3	F3	SW5	A4	GND	E1	VDDO-3
B4	SW3	M6	UCLK	C14	RED0	F2	USB-ASTB	A8	GND	J2	VDDO-3
A3	JA2	P6	LD3	G12	BTN0	F1	USB-DSTB	C1	GND	A5	VDDO-2
A10	JC3	P7	LD2	K14	AN3	G1	LD7	C7	GND	E12	VDDO-2
C9	JC4	M4	BTN2	J12	AN1	G3	SW4	C10	GND	K1	VDDO-2
B9	JC2	N4	LD5	J13	BLU2	H1	USB-DB6	E3	GND	P9	VDDO-2
A9	JC1	M5	LD0	J14	HSYNC	H2	USB-DB5	E14	GND	A11	VDDO-1
B8	MCLK	N5	LD4	H13	BLU1	H3	USB-DB7	G2	GND	D3	VDDO-1
C8	RCCLK	G14	GRN2	H12	CB	B14	TMS	H14	GND	D14	VDDO-1
A7	BTN3	G13	GRN1	J3	JA3	B13	TCK-FPGA	J1	GND	K2	VDDO-1
B7	JB4	F12	AN0	K3	SW2	A2	TDO-USB	K12	GND	L12	VDDO-1
P4	LD6	K13	VSYNC	B1	PS2C	A14	TDO-S3	M3	GND	P2	VDDO-1

Bild 7: Basys 2 Board: Pin-Belegung FPGA

Weitergehende Information zum Board finden Sie im Reference Manual und Schaltplan im Internet bei Digilent (siehe Literatur).

Weitergehende Information zum eingesetzten FPGA finden Sie bei Xilinx im „data sheet of Spartan-3E devices (Xilinx XC3S10 0E)“, die grundlegenden Leistungsmerkmale finden sich in der folgenden Tabelle aus eben diesem Datenblatt.

Table 1: **Summary of Spartan-3E FPGA Attributes**

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM bits(1)	Block RAM bits(1)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs	Total Slices						
XC3S100E	100K	2,160	22	16	240	960	15K	72K	4	2	108	40
XC3S250E	250K	5,508	34	26	612	2,448	38K	216K	12	4	172	68
XC3S500E	500K	10,476	46	34	1,164	4,656	73K	360K	20	4	232	92
XC3S1200E	1200K	19,512	60	46	2,168	8,672	136K	504K	28	8	304	124
XC3S1600E	1600K	33,192	76	58	3,688	14,752	231K	648K	36	8	376	156

Notes:
1. By convention, one Kb is equivalent to 1,024 bits.

Bild 8 Spartan-3E Leistungsmerkmale

Anhang 1: Lösungen und Beispiele für VHDL Code

A.1 Schachuhr

Frage 1.1 Zeichnen Sie ein Blockdiagramm der Schachuhr, das in einer HDL-Beschreibung die Entity darstellt.

Lösung:

Als Eingänge sind vorgesehen: 8 MHz Takt (Clk), Umschaltung zwischen den Spielern A und B (AB), Schalter zum Anhalten bzw. Fortsetzen der Uhr (StartStopp), sowie ein Reset (RS). Gemäß Beschaltung des CPLDs wird jeweils eine Ziffer für die 7-Segmentanzeige (DigitOut) im Zeitmultiplex ausgegeben. Der Ausgang MUXOut steuert den Multiplexer bzw. Demultiplexer.

Frage 1.2: Erstellen Sie ein Konzept bzw. grobes Blockschaltbild für das Innenleben: Welche Funktionen bzw. Komponenten benötigen Sie (z.B. Sekundentakt, Speicher bzw. Zähler für die Spielzeiten, Steuerung durch Tasten, Anzeige etc)? Hinweis: Arbeiten Sie vom Groben ins Feine (erst die Struktur, die Details später). Ignorieren Sie zunächst bitte Reset und Start/Stopp. Abstrahieren Sie die Ansteuerung der 7-Segmentanzeige zu einem Funktionsblock Anzeige, der später detailliert werden kann.

Lösung: z.B. wie folgt.

Die Schaltung benötigt zum Zählen der Sekunden für die Spieler A und B einen Sekundentakt (Sec_Clk). Zum Ausgeben der 4 Ziffern auf die 7-Segmentanzeige wird außerdem ein weiteres Taktsignal benötigt (Dis_Clk). Beide Signale werden durch Teiler aus dem Taktsignal (Clk) erzeugt.

Kern der Schaltung sind die beiden Zähler für die von den Spielern A und B jeweils benötigte Zeit (Zähler A und Zähler B). Mit Hilfe des Umschalters AB wird zwischen diesen Zählern umgeschaltet. Der jeweils aktive Zähler wird über einen Multiplexer zur Anzeige gebracht.

Frage 1.3: Ergänzen Sie das Blockdiagramm um die Funktionen Reset und Start/Stopp. Begründen Sie Ihren Entwurf.

Lösung: z.B. wie folgt. Das Signal StartStopp hält den Sekundentakt an. In der hier gewählten Lösung lässt sich die Schaltung nur bei angehaltener Uhr zurücksetzen (Signal RS). Das Zurücksetzen löscht beide Zählerstände A und B. Bemerkung: Reset sollte durch den Prozess ausgeführt werden, der die Zählerstände erzeugt (andernfalls Zugriffskonflikte).

Frage 1.4: Detaillieren Sie die Speicher bzw. Zähler für die Spielstände. Begründen Sie Ihren Entwurf.

Lösung: Die Zähler lassen sich z.B. gleich mit Hilfe jeweils einer Ziffer pro Dezimalstelle ausführen, d.h. Ziffer1 für die Sekunden (von 0 bis 9), Ziffer 2 für 10-Sekunden (con 0x bis 5x), Ziffer 3 für die Minuten (von 0 bis 9), Ziffer 4 für 10-Minuten (von 0x bis 5x). Überlauf einer Stelle löst einen Zählimpuls in der folgenden Stelle aus.

Mit dem Signal (AB) für das Umschalten zwischen den Spielern A und B wird jeweils ein Zähler für A und B aktiviert, sowie der Ausgang des Jeweils aktiven Zählers auf Register D1 bis D4 zur weiteren Verarbeitung übertragen (Funktion des Multiplexers im Blockschaltbild).

Frage 1.5: Detaillieren Sie den Funktionsblock für die 7-Segmentanzeige so, dass hervor geht, wie Sie die in der Abbildung oben gezeigten 4 Ziffern ansteuern.

Lösung: z.B. wie in folgendem Diagramm. Der Ausgang D, der den jeweils aktiven Spieler repräsentiert, wird ziffernweise ausgegeben. Der Zeitmultiplex erfolgt im Takt der Display-Clock (Dis_Clk). Für die Ausgabe auf der Beschaltung des CPLDs wird das Multiplex-Signal in 4 Bits kodiert (Ausgangssignal MUXOut), sowie die jeweils übertragene Ziffer vom BDC-Format in das Format der 7-Segmentanzeige (Ausgangssignal DigitOut).

Frage 1.6: Übersetzen Sie das Blockdiagramm in HDL-Text für den Architektur-Block (Struktur mit Prozessen, ohne Details).

Lösung: z.B. wie im folgenden HDL-Text. Zum Testen auf dem Simulator sind große Teiler für den Takt nicht praktikabel, daher wurde auf den Teiler für die Display-Clock verzichtet. Dieser kann bei einer Synthese ergänzt werden. Teiler für eine 8 MHz Taktrate: (1) 80000 für eine Display Clock (Dis_Clk) von 100 Hz, (2) 100 zur Ableitung des Sekundentaktes (Sec_Clk) aus der Display-Clock (DIS_Clk).

Struktur der Prozesse: entsprechend dem Blockschaltbild

(1) Teiler (Zähler) zur Erzeugung des Takts für Display (z.B. 100 Hz) und davon abgeleitet für die Sekunden (1 Hz)

(2) Zähler für die Spieler A und B mit Umschaltlogik und Multiplexer für den aktiven Spieler (auf die Ausgangssignale D1 bis D4). Die Zähler sind gleich als Dezimalstellen ausgeführt. Dieser Prozess wird mit dem Sekundentakt (Sec_Clk) betrieben.

(3) Zähler für den Multiplexer bzw. Demultiplexer für die Anzeige (Select Signal MUX). Dieser Prozess wird mit dem Takt für das Display betrieben (Dis_Clk).

(4) Anzeigensteuerung: Der Zähler MUX (Select-Signal) wird übersetzt in 4 einzelne Bits (Signal Display_MUX). Dieser Prozess folg dem MUX-Zähler (MUX).

(5) Dekoder für 7-Segmentanzeige: In der hier gewählten Lösung sitzt der Dekoder in der Multiplex-Leitung, d.h. die jeweils übertragene Ziffer DOut (im BCD-Format) wird übersetzt in das 7-Segment-Format DigitOut. Dieser Prozess folgt dem Signal DOut.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--use ieee.std_logic_unsigned.all;
use IEEE.numeric_std.all;

entity Chess_Clock is
 Port (RS          : in STD_Logic;      -- Reset (active low)
       Clk         : in  STD_LOGIC;     -- Serial clock 8 MHz
       AB          : in  STD_LOGIC;     -- Select A = 1, B = 0
       StartStop   : in  STD_LOGIC;     -- Stop Clock = 0, Continue = 1
       DigitOut    : out  STD_LOGIC_vector (7 downto 0); -- One Segment
      MUXOut      : out  STD_LOGIC_vector (3 downto 0));
end Chess_Clock;
```

```
architecture RTL of Chess_Clock is

-- internal signals and variables
signal Dis_Clk : std_Logic := '0'; -- display clock
signal Sec_Clk : std_Logic := '0'; -- clock in seconds
signal D1, D2, D3, D4, DOut : unsigned (3 downto 0):= (others=>'0');
-- minutes (4_3) and seconds (2_1) as output to the display
signal DCnt : unsigned (15 downto 0):= (others=>'0');
-- counter for Dis_Clk
signal SCnt : unsigned (7 downto 0):= (others=>'0');
-- counter for Sec_Clk
signal MUX : unsigned (1 downto 0):= (others=>'0');
-- Select Signal for MUX and DeMUX of Digits

begin

 Dis_Clk <= Clk; -- consider Clk as Display clock for tests at 100 Hz

 -- clock dividers for Dis_Clk and Sec_Clock
 Seconds : process (Dis_Clk, StartStop) begin
      if ((rising_edge(Dis_Clk)) and (StartStop = '1')) then
          if (SCnt < 50) then SCnt <= SCnt + 1;
          else
              SCnt <= "00000000";
              Sec_Clk <= not Sec_Clk;
          end if;
      end if;
 end process Seconds;

    -- counter for players A and B

 AB_count : process (Sec_Clk, AB)

   variable A1, A2, A3, A4 : unsigned (3 downto 0):= (others=>'0');
    -- minutes (4_3) and seconds (2_1) Player A
   variable B1, B2, B3, B4 : unsigned (3 downto 0):= (others=>'0');
    -- minutes (4_3) and seconds (2_1)  Player B

  begin
    if (rising_edge(Sec_Clk)) then
       case AB is
          when '1' =>               -- counting for A
               if (A1 < 9) then
                   A1 := (A1 + 1);
                 else
                   A1 := "0000";
                   A2 := (A2 + 1);
                end if;
                if (A2 > 5) then
```

```
                  A2 := "0000";
                  A3 := (A3 + 1);
               end if;
               if (A3 > 9) then
                  A3 := "0000";
                  A4 := (A4 + 1);
               end if;
               if (A4 > 5) then
                  A4 := "0000";
               end if;
       when '0' =>                -- counting for B
               if (B1 < 9) then
                  B1 := (B1 + 1);
                else
                  B1 := "0000";
                  B2 := (B2 + 1);
               end if;
               if (B2 > 6) then
                  B2 := "0000";
                  B3 := (B3 + 1);
               end if;
               if (B3 > 9) then
                  B3 := "0000";
                  B4 := (B4 + 1);
               end if;
               if (B4 > 5) then
                  B4 := "0000";
               end if;

       when others => D4 <= "1111";
    end case;

    case AB is
         when '1' =>                -- output for A
           D1 <= A1; D2 <= A2; D3 <= A3; D4 <= A4;  -- output counter A
         when '0' =>                -- output for B
           D1 <= B1; D2 <= B2; D3 <= B3; D4 <= B4;  -- output counter B
      when others => NULL;
    end case;
   end if;
end process AB_count;

-- multiplexer select signal for 7-segment display
MUXCount : process (Dis_Clk) begin
  if (rising_edge(Dis_Clk)) then
      if (MUX < 4) then MUX <= MUX + 1;
        else MUX <= "00";
      end if;
  end if;
end process MUXCount;
```

```
  -- multiplex and decode to 7-segment display
  Display_MUX : process (MUX) begin -- generates multiplex signal
    case MUX is
      when "00" => MUXOut <= "0001"; DOut <= D1;
      when "01" => MUXOut <= "0010"; DOut <= D2;
      when "10" => MUXOut <= "0100"; DOut <= D3;
      when "11" => MUXOut <= "1000"; DOut <= D4;
      when others => MUXOut <= "0000";
    end case;
  end process Display_MUX;

  Display_Decoder : process (DOut) begin -- decodes DOut to 7-Segm.
    case DOut is
      when x"0" => DigitOut <= "01111110";
      when x"1" => DigitOut <= "00110000";
      when x"2" => DigitOut <= "01101101";
      when x"3" => DigitOut <= "01111001";
      when x"4" => DigitOut <= "00110011";
      when x"5" => DigitOut <= "01011011";
      when x"6" => DigitOut <= "01011111";
      when x"7" => DigitOut <= "01110000";
      when x"8" => DigitOut <= "01111111";
      when x"9" => DigitOut <= "01111011";
      when others => DigitOut <= "11111111"; -- E: Error
    end case;
  end process Display_Decoder;

end RTL;
```

Zur Ergänzung ein Simulatorlauf (mit SCnt < 5 statt 50):

Anhang 2: Literatur

(1) Entwurf digitaler Systeme, K. Gosger, Vorlesungsmanuskript, DHBW-Stuttgart

(2) Entwicklungswerkzeug: Xilinx ISE Design Suite (inkl. WebPACK) (bitte vorab auf ihrem Laptop installieren): http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.html

(3) Software Adept zur Programmierung des Zielsystems (bitte vorab auf ihrem Laptop installieren): https://reference.digilentinc.com/reference/software/adept/start

(4) Zielsystem: Basys 2 Board von Digilent, Reference Manual siehe https://reference.digilentinc.com/reference/programmable-logic/basys-2/start?redirect=1

(5) Tutorial von Digilent: „Getting started with FPGA“ https://reference.digilentinc.com/learn/programmable-logic/tutorials/getting-started-with-fpga/start

(7) Spartan-3E FPGA Family Data Sheet: https://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

(6) Peripherie Module zum Zielsystem, siehe z.B. http://www.digilentinc.com/