Aufgabe 3 (10 + 40 Punkte)

Lernziele: Der von uns eingesetzte FPGA-Baustein XC3S200 ist mit mehreren sog. primitiven Komponenten ausgestattet. Beispielsweise verfügt er über zwölf konfigurierbare RAM-Blöcke und zwölf 18x18-Bit-Multiplizierer. Im Rahmen dieser Aufgabe sollen Sie u.a. die Funktionsweise und die Anwendung solcher Komponenten kennen lernen. Als Vorbereitung zur Lösung dieser Aufgabe machen Sie sich mit den Applikationsberichten *Using Block RAM in Spartan-3 Generation FPGAs* (XAPP463.pdf) und *Using Embedded Multipliers in Spartan-3 FPGAs* (XAPP467.pdf) vertraut.

Außerdem vertiefen Sie Ihre Kenntnisse aus den vorherigen Aufgaben und erweitern Ihr Wissen, in dem Sie sich mit folgenden Punkten befassen:

a) Entwurf und Implementierung von VHDL-Komponenten mit selbst definierten Schnittstellen,
b) synthesegerechte Verhaltensbeschreibung komplexerer Schaltwerke mit Hilfe von PROCESS-, IF- und CASE-Anweisungen,
c) Strukturbeschreibung im hierarchischen Entwurf durch Instanzierung von bereitgestellten, selbst entwickelten und primitiven Komponenten.

Aus der 1. Aufgabe übernehmen Sie die Komponente zur Darstellung von Zahlen auf der 4-stelligen 7-Segmentanzeige, und aus der 2. Aufgabe verwenden Sie die Schaltung zur Entprellung von Tasten und ggf. den Universalzähler.

Hinweise:

1. Vorgegebene Schnittstellen oder bereitgestellte Komponenten dürfen weder geändert noch durch andere ersetzt werden.
2. Der Takt in synchronen Systemen ist eine „unantastbare“ Größe und darf auf keinen Fall über Gattern mit anderen Signalen verknüpft werden. So etwas ist ein schlechter Programmierstil und Hinweis auf mangelnde Kenntnisse der Digitaltechnik. Der Takt darf nur direkt an die Clock-Eingänge von Flipflops angeschlossen werden.

Aufgabenstellung.

Im Rahmen dieser Aufgabe ist eine *durchsatzoptimierte* und *fließbandorganisierte* Schaltung in VHDL zu beschreiben und für den FPGA-Baustein XC3S200 auf dem Entwicklungs-Board zu synthetisieren. Diese Schaltung soll das Skalarprodukt zweier Vektoren berechnen, und zwar durch Multiplizieren der einzelnen Elemente der Vektoren und Aufsummieren der Zwischenprodukte.

Der Kern der Schaltung soll ein modular aufgebautes Schaltwerk sein, das aus einem Steuerwerk und aus einem fließbandorganisierten Rechenwerk besteht. Das Steuerwerk und das Rechenwerk sollen zusammen in einer Komponente namens *core* integriert werden. Die Schnittstelle dieser Komponente ist in der Datei *core.vhd* vorgegeben und in Bild 1 zu sehen. Der Kern ist in einem übergeordneten System (in der Datei *Aufgabe3.vhd*) zusammen mit der Schaltung zur Tastenentprellung und dem BCD-Transcoder zu einem vollständigen System integriert.

```
entity core is
   generic(RSTDEF: std_logic := '0');
   port(rst:   in  std_logic;                        -- reset,          RSTDEF active
        clk:   in  std_logic;                        -- clock,          rising edge
        swrst: in  std_logic;                        -- software reset, RSTDEF active
        strt:  in  std_logic;                        -- start,          high active
        sw:    in  std_logic_vector( 7 DOWNTO 0);    -- length counter, input
        res:   out std_logic_vector(43 DOWNTO 0);    -- result
        done:  out std_logic);                       -- done,           high active
end core;
```

Bild 1: Schnittstelle der core-Komponente.

In Bild 2 ist der Aufbau des gesamten Systems schematisch dargestellt, wobei der FPGA samt Inhalt grau hervorgehoben ist. In der Einsatzumgebung des FPGA-Bausteins befinden sich ein 50-MHz-Taktgenerator, acht Schiebeschalter $sw_7..sw_0$, zwei Tasten btn_0 und btn_3, eine Leuchtdiode ld_0 sowie eine 4-stellige 7-Segmentanzeige. Der Taster btn_3 liefert ein zentrales Rücksetzsignal, mit dem alle Speicherelemente in der Schaltung zurückgesetzt werden sollen.

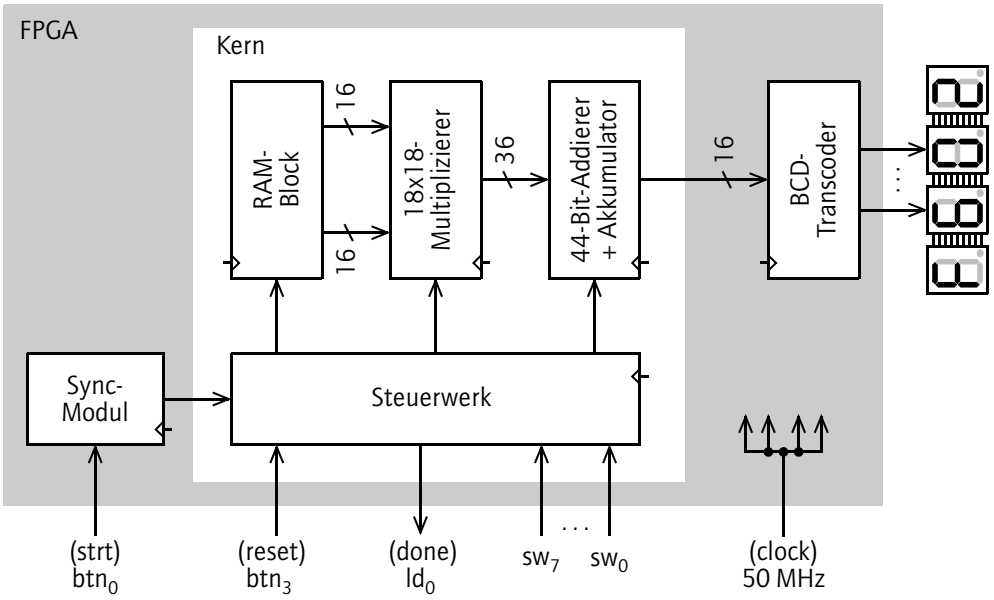

Bild 2: Schematischer Aufbau des Systems.

Das Rechenwerk der core-Komponente setzt sich aus einem RAM-Block, einem 18x18-Bit-Multiplizierer und einem 44-Bit-Addierer mit einem Akkumulator zusammen. Der RAM-Block ist als 16-Bit-Dual-Port-Speicher mit 1024 Speicherzellen konfiguriert. Er beinhaltet die beiden Vektoren und steht als Datei *ram_block.vhd* zur Verfügung. Jeder Vektor umfaßt 256 vorzeichenbehaftete 16-Bit-Dualzahlen im 2er-Komplementformat. Beide Vektoren sind im RAM-Block so angeordnet, daß der erste Vektor die ersten 256 Speicherzellen ab der Adresse 0x0000 bis 0x00FF belegt, und der zweite Vektor die darauffolgenden 256 Speicherzellen ab der Adresse 0x0100 bis 0x01FF. Weil der RAM-Block als Dual-Port-Speicher konfiguriert ist, ist es auch möglich, und im Rahmen dieser Aufgabe durchaus wünschenswert, parallel auf die Elemente beider Vektoren zuzugreifen.

Der 18x18-Bit-Multiplizierer, der 44-Bit-Addierer mit dem Akkumulator bilden die sog. MAC-Einheit, die zwei vorzeichenbehaftete Werte multipliziert und die partiellen Produkte aufsummiert. Auf dieser MAC-Einheit basiert die Berechnung des Skalarprodukts zweier Vektoren. Der 18x18-Bit-Multiplizierer ist ein vorzeichenbehafteter Multiplizierer, und er soll mit Hilfe der primitiven Komponente MULT18X18S realisiert werden. Dazu ist es erforderlich, die Bibliothek *UNISIM* mit dem Paket *VComponents* zu benutzen. Da im RAM-Block 16-Bit-Zahlen breit sind, der Multiplizierer aber 18-Bit-Zahlen verarbeiten kann, ist es vor der Multiplikation eine Vorzeichenerweiterung der beiden Operanden erforderlich. Der 44-Bit-Addierer ist ein vorzeichenbehafteter Addierer mit einem integrierten 44-Bit-Akkumulator. Weil die 4-stellige 7-Segmentanzeige nur 16-Bit-Werte darstellen kann, werden nur die 16 untersten Bitstellen des Akkumulators anzeigen.

Das Steuerwerk der core-Komponente ist die zentralle Einheit, die den gesamten Ablauf des Berechnungsalgorithmus steuert. Die Berechnung wird mit dem Signal strt (durch Niederdrücken der Taste btn_0 nach der Entprellung im Sync-Modul) gestartet, und das Fertigstellen des Ergebnisses wird mit dem Signal done (Leuchten der Leuchtdiode ld_0) angezeigt. Das Signal done soll in der Reset-Phase, beim Start und während der Ausführung des Berechnungsalgorithmus deaktiviert bleiben. Beim Start wird die Einstellung der Schiebeschalter $sw_7..sw_0$ ins Steuerwerk übernommen. Diese Einstellung ist als eine vorzeichenlose 8-Bit-Dualzahl zu interpretieren, die die Werte von 0 bis 255 darstellen kann. Der Wert dient dazu, den Berechnungsalgorithmus für Vektoren mit verschiedenen Längen auszuführen. So bedeutet z.B. der Wert 9, daß die ersten neun Wertepaare aus den beiden Vektoren miteinander multipliziert und die Zwischenresultate aufsummiert werden. Das Steuerwerk ist so zu entwickeln, daß die Ausführung des Berechnungsalgorithmus nicht nur einmalig sondern immer durch Betätigung des Tasters btn_0 möglich ist.

Für diese Aufgabe steht die Datei *core_tb.vhd* mit einer Testumgebung zur Verfügung. Mit Hilfe der dort vorbereiteten Testfälle werden bei der Abnahme der Aufgabe sowohl die Funktionalität des Schaltung geprüft als auch der Durchsatz der Berechnung gemessen. Der Berechnungsalgorithmus ist am folgenden Programmausschnitt zu sehen.

```
type int_vector is array(natural range <>) of integer;
constant ram: int_vector(0 to 1023) := ( ... );

procedure skalar(N: natural) is
   variable prod, sum: integer;
begin
   sum := 0;
   for i in 0 to N-1 loop
      prod := ram(i) * ram(256+i);s
      sum  := acc + prod;
   end loop;
   wait;
end procedure;
```

Bild 3: Algorithmus zur Berechung des Skalarproduktes.