Aufgabe 3

Aufgabenstellung. Diese Aufgabe setzt sich aus zwei Teilaufgaben zusammen: aus der Implementierung eines applikationsspezifischen Co-Prozessors und aus einer dazu passenden Erweiterung der Assembler-Applikation.

1) Implementieren Sie in der Hardwarebeschreibungssprache VHDL einen applikationsspezifischen Co-Prozessor, der den PicoBlaze beim Sortieren von Daten nach dem Insertion-Sort-Algorithmus unterstützt, wobei der Algorithmus vollständig in Hardware zu realisieren ist. Der Algorithmus ist im folgenden Listing zu sehen.

```
PROCEDURE sort(a: INOUT string; n: positive) IS
   VARIABLE key: character;
   VARIABLE j: integer;
BEGIN
   FOR i IN 1 TO n-1 LOOP
      key := a(i);
      j := i-1;
      WHILE j >= 0 LOOP
         IF a(j) <= key THEN
            EXIT;
         END IF;
         a(j+1) := a(j);
         j := j - 1;
      END LOOP;
      a(j+1) := key;
   END LOOP;
END PROCEDURE;
```

Die zu sortierenden Daten werden über die serielle Schnittstelle empfangen (vgl. 2. Aufgabe) und durch den PicoBlaze in einem RAM-Block bereitgestellt. Der Co-Prozessor muss geeignete Schnittstellen haben, über die er einerseits auf Daten im RAM-Block lesend und schreibend zugreifen kann, und andererseits mit dem PicoBlaze kommunizieren kann. Die Schnittstelle zum RAM-Block resultiert aus den Eigenschaften der primitiven Komponente RAMB16_S9_S9 und ist in der bereitgestellten Literatur von Xilinx beschrieben. Die Schnittstelle zwischen dem PicoBlaze und dem Co-Prozessor besteht aus einigen Registern, und die Kommunikation erfolgt nach dem Master-Slave-Prinzip, wobei der PicoBlaze der Master im System ist. Zu den Registern gehören ein 11-Bit-Adressregister (PTR), ein 8-Bit-Längenregister (LEN) und ein 1-Bit-Status-/Steuerregister mit den beiden Bits STRT und DONE. Der PicoBlaze übergibt über das Register PTR die Anfangsadresse des zu sortierenden Datenfeldes, wobei das Datenfeld nur alphanumerische ASCII-Zeichen enthält. Über das Register LEN übergibt er die Länge des Datenfeldes (Anzahl der Zeichen), wobei die Länge maximal 255 betragen kann. Nach der Initialisierung der Register PTR und LEN kann der Co-Prozessor gestartet werden. Dazu muss der PicoBlaze das Bit STRT im Status-/Steuerregister auf 1 setzen. Von diesem Zeitpunkt an führt der Co-Prozessor den Sortieralgorithmus selbständig aus. Ist der Co-Prozessor mit dem Sortiervorgang fertig, so setzt er seinerseits das Bit DONE im Statusregister auf 1, womit er dem PicoBlaze die Fertigstellung der Daten anzeigt. Damit der PicoBlaze diesen Zustand feststellen kann, muss er das Bit DONE regelmäßig abfragen. Anschließend sendet der PicoBlaze das sortierte Datenfeld über die serielle Schnittstelle zurück.

Der Co-Prozessor ist so zu implementieren, dass seine Beschreibung in einer Testbench simulierbar ist, und für den FPGA-Baustein Spartan3 auf dem Entwicklungsboard synthetisierbar ist. Zur Bewertung der Lösung werden drei Werte herangezogen: die Schaltungskomplexität (Anzahl der verbrauchten Slices), die erreichte Taktfrequenz und der Durchsatz. Die Schaltungskomplexität und die Taktfrequenz resultieren aus der Synthese. Der Durchsatz, also die Anzahl der durch den Co-Prozessor benötigten Takte bezogen auf die Länge des zu sortierenden Datenfeldes, wird in einer Testumgebung gemessen.

2) Erweitern Sie Ihre Applikation (das Assembler-Programm aus der 2. Aufgabe) derart, dass sie jetzt Datenblöcke in einer Endlosschleife verarbeiten kann. Ein Datenblock hat eine feste Länge von 255 Zeichen und kann beliebige ASCII-Zeichen enthalten. Die von Ihnen zu entwickelnde Applikation soll solche ASCII-Zeichen nach dem asynchron-seriellen Übertragungsprotokoll empfangen, nicht-alphanumerische ASCII-Zeichen herausfiltern, alphanummerische ASCII-Zeichen in einem Puffer in einem RAM-Block ablegen. Nach dem der Datenblock vollständig empfangen worden ist, soll sie den Co-Prozessor starten, dann solange warten, bis der Co-Prozessor mit dem Sortiervorgang fertig ist, und schließlich, die alphanummerischen ASCII-Zeichen aus dem RAM-Block zurück senden. Der Datenblock wird von einem Laborrechner aus mit Hilfe eines Terminalprogramms an das FPGA-Entwicklungsboard mit Ihrer Applikation gesendet. Derselbe Laborrechner mit dem Terminalprogramm dient als Empfänger des verarbeiteten Datenblocks vom FPGA-Entwicklungsboard mit Ihrer Applikation. Es ist davon auszugehen, dass der Datenblock mindestens ein alphanummerisches ASCII-Zeichen enthält, und dass der Datenblocktransfer zwischen Laborrechner und FPGA-Entwicklungsboard nur im Halb-Duplex-Modus erfolgt, d.h. ein neuer Datenblock wird vom Laborrechner aus erst dann gesendet, nachdem der vorherige, sortierte Datenblock empfangen worden ist. Synchronisations- oder Überwachungsmaßnahmen sind nicht erforderlich.

Programmierschnittstelle. Neben der aus den vorherigen Aufgaben bekannten Programmierschnittstelle wird die Programmierschnittstelle für die 3. Aufgabe um vier neue Adressen erweitert, über die der Zugriff auf interne Register des Co-Prozessors möglich ist.

Unter der 4-Bit-Adresse $(1000)_2$ befindet sich das Status-/Steuerregister des Co-Prozessors. Auf das Steuerregister, das nur aus dem Bit STRT auf der Position 0 besteht, kann man nur schreibend zugreifen. Das Ablegen einer Eins im Bit STRT startet automatische den in Hardware implementierten Sortieralgorithmus. Auf das Statusregister, das nur aus dem Bit DONE auf der Position 0 besteht, kann man nur lesend zugreifen. Mit einer Eins drin zeigt der Co-Prozessor an, dass er mit der Arbeit fertig ist.

Unter den 4-Bit-Adressen $(1001)_2$ und $(1010)_2$ verbirgt sich ein 11-Bit-Adressregister (PTR). Über dieses Register wird die Anfangsadresse des zu sortierenden Datenfeldes im RAM-Block übergeben. Da RAM-Block-Adressen 11 Bit lang sind, der PicoBlaze aber nur über eine 8-Bit-Ein-/Ausgabeschnittstelle verfügt, ist es notwendig, die 11-Bit-Adresse auf zwei Bytes zu verteilen; und zwar so, dass unter der 4-Bit-Adresse $(1001)_2$ die Adressbits PTR7 bis PTR0, und unter der 4-Bit-Adresse $(1010)_2$ die Adressbits PTR10 bis PTR8 verfügbar sind. Es sind sowohl Lese- als auch Schreibzugriffe auf diese Adressen möglich. Bei der Ausfügung von OUTPUT-Befehlen mit diesen Adressen werden Werte im 11-Bit-Adressregister abgelegt. Durch die Ausführung von INPUT-Befehlen mit diesen Adressen kann die zuletzt gespeicherte Adresse ermittelt werden.

Die vierte 4-Bit-Adresse $(1011)_2$ ist für ein 8-Bit-Längenregister (LEN) vorgesehen, über das die Datenlänge des zu sortierenden Datenfeldes übergeben wird. Es sind sowohl Lese- als auch Schreibzugriffe auf diese Adressen möglich. Beim Lesezugriff auf diese Adresse mit einem INPUT-Befehl wird der zuletzt gespeicherte Wert zurückgegeben.

Die untere Tabelle gibt den Überblick über das sog. Mamory-Mapping beim Zugriff auf Peripheriekomponenten, den RAM-Block und den Co-Prozessor.

Schnittstellen zwischen PicoBlaze und Peripheriekomponenten

PORT_ID[3..0]	Mode	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
Register im IO-Bereich									
0000	Read	-	-	-	-	-	BTN0	LD1	LD0
	Write	-	-	-	-	-	-	LD1	LD0
0001	Read	-	-	-	-	-	-	-	RXD
	Write	-	-	-	-	-	-	-	TXD
Register für den Zugriff auf RAM-Block									
0010	Read	A7	A6	A5	A4	A3	A2	A1	A0
	Write	A7	A6	A5	A4	A3	A2	A1	A0
0011	Read	-	-	-	-	-	A10	A9	A8
	Write	-	-	-	-	-	A10	A9	A8
0100	Read	D7	D6	D5	D4	D3	D2	D1	D0
	Write	D7	D6	D5	D4	D3	D2	D1	D0
Register für den Zugriff auf Co-Prozessor									
1000	Read	-	-	-	-	-	-	-	DONE
	Write	-	-	-	-	-	-	-	STRT
1001	Read	PTR7	PTR6	PTR5	PTR4	PTR3	PTR2	PTR1	PTR0
	Write	PTR7	PTR6	PTR5	PTR4	PTR3	PTR2	PTR1	PTR0
1010	Read	-	-	-	-	-	PTR10	PTR9	PTR8
	Write	-	-	-	-	-	PTR10	PTR9	PTR8
1011	Read	LEN7	LEN6	LEN5	LEN4	LEN3	LEN2	LEN1	LEN0
	Write	LEN7	LEN6	LEN5	LEN4	LEN3	LEN2	LEN1	LEN0