Aufgabe 2

Aufgabenstellung. Erweitern Sie Ihre Applikation (das Assembler-Programm aus der 1. Aufgabe) derart, dass sie jetzt Datenblöcke in einer Endlosschleife verarbeiten kann. Ein Datenblock hat eine feste Länge von 16 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 und schließlich, d.h. nach dem der Datenblock vollständig empfangen worden ist, 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, verarbeitete Datenblock empfangen worden ist. Synchronisations- oder Überwachungsmaßnahmen sind nicht erforderlich.

Wenn die Anzahl der Register im PicoBlaze nicht ausreichend ist, benutzen Sie Speicherzellen im Scratch-Pad-Speicher. Der Zugriff darauf erfolgt mit den Befehlen FETCH und STORE.

Hinweise. Im Hinblick auf die 3. Aufgabe halten Sie die Datenblocklänge flexibel, so dass sie auf einen anderen Wert bis maximal 256 leicht geändert werden kann.

Programmierschnittstelle. Neben der aus der 1. Aufgabe bekannten Programmierschnittstelle wird die Programmierschnittstelle für die 2. Aufgabe um drei neue Adressen erweitert, über die der Zugriff auf den RAM-Block möglich ist. Unter den 4-Bit-Adressen $(0010)_2$ und $(0011)_2$ verbirgt sich ein 11-Bit-Adressregister, der zum adressieren des RAM-Blocks dient. 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 $(0010)_2$ die Adressbits A7 bis A0, und unter der 4-Bit-Adresse $(0011)_2$ die Adressbits A10 bis A8 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 dritte 4-Bit-Adresse $(0100)_2$ ist für den 8-Bit-Datenaustausch mit dem RAM-Block vorgesehen. Beim Schreibzugriff auf diese Adresse mit einem OUTPUT-Befehl wird ein Byte direkt in den RAM-Block gespeichert, und zwar unter derjenigen Adresse, die im 11-Bit-Adressregister steht. Beim Lesezugriff auf diese Adresse mit einem INPUT-Befehl wird ein Byte aus dem RAM-Block gelesen, und zwar aus derjenigen Speicherzelle, die mit der Adresse aus dem 11-Bit-Adressregister adressiert wird. Die untere Tabelle gibt den Überblick über das sog. Mamory-Mapping beim Zugriff auf Peripheriekomponenten und den RAM-Block.

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