**1.1 Aufgabe**

Studieren Sie den Aufbau des DSPs und des DSP-Kerns (Core) in den Kapiteln 1 „Introduction“ und 2 „Computational Units“ im Manual „ADSP-BF561 Blackfin® Processor Hardware Reference“.

a) Wieviele DSP-Cores besitzt der Blackfin BF-561 (im Folgenden kurz DSP)?

2 Kerne

b) Welche Schnittstellen besitzt der DSP?

UART, SPI, SPORT0 (serieller PORT), SPORT1, GPIO

c) Welche Recheneinheiten (computational units) besitzt jeder DSP-Core?

Jeder Kern besitzt 4 Addierer (8Bit) und 2 MACs (Multiply and Accumulate). Diese führen in jedem Takt eine 16Bit x 16Bit Multiplikation mit Akkumulation zu einem 40Bit Ergebnis durch. (mit 8Bit extended precision)

d) Welche Rechnerarchitektur besitzt der DSP? Wie unterscheidet sie sich von einer klassischen von-Neumann- oder Harvard-Architektur?

Der DSP besitzt eine modifizierte Harvard Architektur. Die beiden L1 Speicher sind als Harvard Architektur aufgebaut. Die über den Bus erreichbaren Speicher L2 und ggf. externer Speicher sind wie bei einer von Neumann Architektur erreichbar.

e) Welche Befehlswortlängen besitzt der DSP?

Das Blackfin Befehlsset ist optimiert für 16Bit Befehle, komplexe Befehle sind zu 32Bit Befehlen kodiert.

f) Wie groß ist der Addressraum des DSPs?

4Gb Adressraum, 32Bit

g) Über welche internen Speicherressourcen verfügt der DSP?

16K Cache Memory

16K SRAM

4 Bänke 16K L1 Data memory

4K Scratchpad SRAM

h) Wieviele programmable Flags (PF) besitzt der DSP? Nennen Sie 4 Register, die zur Konfiguration der PFs dienen.

Er verfügt über 48 programmierbare general purpose I/O Flags.

* Flag Direction Control Register
* Flag Control and Status Registers
* Flag Interrupt Mask Registers
* Flag Interrupt Sensitivity Registers

i) In welchem Datenformat werden vorzeichenbehaftete Zahlen vom DSP verarbeitet?

2er Komplement

j) Welche Datenregister (Data Registers) besitzt der DSP zur Verarbeitung von Daten in der DAU (Data Arithmetic Unit)? Welche Länge haben sie?

R0-R7 mit jeweils 32Bit

k) Welche Zeigerregister (Pointer Registers) besitzt der DSP zur Adressierung von Daten in der AAU (Address Arithmetic Unit)? Welche Länge haben sie?

Zeigerregister P0 bis P5, FP, SP mit 32Bit Länge

l) Welche Operationen lassen sich mit der ALU (Arithmetic Logic Unit) ausführen? Welche Wortbreite können die Operanden dabei haben?

Multplikation und Addition

8, 16, 32 Bit

m) Welche Operationen lassen sich mit dem Barrel Shifter ausführen? Welche Wortbreite können die Operanden dabei haben?

Arithmetische Shifts, Logische Shifts, Bit Rotation

16, 32, oder 40Bit

**1.2 Aufgabe**

Studieren Sie die Kontrollregister der programmable Flags des DSP im Kapitel 14 „Programmable Flags“ im Manual „ADSP-BF561 Blackfin® Processor Hardware Reference“.

a) Wozu dienen die Register FI0n\_DIR, FI0n\_INEN und FI0n\_FLAG\_D und wie müssen Sie gesetzt werden, um die PFs zu konfigurieren und zu benutzen?

Diese dienen dazu um das Verhalten der PFs zu konfigurieren.

Mit dem **Flag Direction Register** kann jedes einzelne PF als Eingang oder Ausgang konfiguriert werden.

Mit dem **Flag Input Enable Register** schaltet man den Input Buffer für Eingangspins, diese sind Standardmäßig ausgeschaltet.

Mit dem Flag Data Register kann der Zustand von Ausgabe PFs eingestellt, und von Eingabe PFs eingelesen werden

**1.3 Aufgabe**

Studieren Sie das Datenblatt des Audio-Codecs „AD1836 Data Sheet“ und finden Sie die folgenden Kennwerte des Codecs heraus:

a) Auflösung (Bit)

bis zu 24Bit

b) Abtastfrequenz (kHz)

bis zu 96kHz

c) Analoge Aussteuerung der Eingänge und Ausgänge (Volt)

Aussteuerbereich ADC: 2,18V Effektiv, entspricht 6,17Vss

Austeuerungsbereich DAC: 2V Effektiv, entspricht 5,65Vss

**1.4 Aufgabe**

Erstellen Sie in der Programmiersprache C eine Funktion *copyData*, welche beim Aufruf aus der o.g. ISR (d.h. mit jedem Abtasttakt) die Audiodaten des Kanals „Internal ADC R0“ (rechter Kanal des ADC 0) aus dem DMA-Lesepuffer in einen Speicherbereich *iInput* der Größe *N* schreibt, so dass in diesem Speicherbereich immer die letzten *N* eingelesenen Werte x(n) … x(n−N+1) des Audiosignals verfügbar sind, s. Abbildung 3. Die Adresse *\*pWrite,* unter der der aktuelle Signalwert abgelegt werden soll, muss von der Funktion *copyData* jeweils inkrementiert werden und beim Erreichen der Speichergrenze wieder auf die Basisadresse *iInput* zurückgesetzt werden. Der Speicher *iInput* wird dadurch zirkular adressiert.

**1.5 Aufgabe**

Erstellen Sie in der Programmiersprache C eine Funktion *genSinus*, welche beim Aufruf aus der o.g. ISR (d.h. mit jedem Abtasttakt) jeweils den Wert einer Sinusfunktion liefert, so dass bei sukzessivem Aufruf dieser Funktion im Takt der Abtastfrequenz und Ausgabe des Wertes

über den Audio-Codec am Ausgang des EVB ein Sinussignal generiert wird. Die generierte Sinusfunktion soll dabei eine normierte Amplitude *A* im Bereich 0 bis 1 besitzen. Die Frequenz soll über einen Parameter *F200* in 200-Hertz-Schritten mit *F200* = 0 … 50 einstellbar sein. Beispielsweise soll für *F200* = 10 ein Sinussignal mit der Frequenz 2 kHz erzeugt werden.

**1.6 Aufgabe**

Studieren Sie die Load-, Move- und Store-Befehle des DSP in den Kapiteln 8 und 9 des o.g. Manuals.

a) Was ist der Unterschied zwischen einem unmittelbaren (immediate) und einem indirekten (indirect) Ladebefehl (Load)? Wie wird syntaktisch dazwischen unterschieden?

Ein immediate Load lädt Konstanten direkt aus dem Programmcode in ein Register des DSP Kerns.

Indirekte Ladebefehle laden Daten aus dem Datenspeicher des DSPs in ein Register des DSP Kerns.

Syntax **immediate**:

* r3=20(z); lädt die Dezimalzahl 20 in das Datenregister 3  
  Nichtverwendete höherwertigere Datenbits werden mit Nullen aufgefüllt (zero extended)
* p1=0x1234(z); lädt Hexadezimale 0x1234 in das Zeigerregister p1, nichtbenötigte Bits werden mit Nullen aufgefüllt
* l3.h=0xbcde; Lädt Hexadezimale Zahl 0xbcde in die höherwertigen Bits des Adressregisters I3  
  Da alle 16Bit des Adressregisters verwendet werden muss kein Extension Mode angegeben werden.

Syntax **indirektes** laden:

* p2=[p1]; lädt den Inhalt der Speicherzelle auf die p1 zeigt p2

Unterschied zum direkten Laden sind die eckigen Klammern um den Zeiger

b) Erläutern Sie den Unterschied zwischen den folgenden indirekten Speicherbefehlen (Store):

[P0] = R1; W[P0] = R1; B[P0] = R1;

Mit W und B wird die Größe der Speicherstelle festgelegt. W=16Bit, B=8Bit, keine Angabe=32Bit

[P0] = R1; speichert die 32Bit von R1 in die Speicherstelle auf die P0 zeigt.

W[P0] = R1; speichert die unteren 16Bit von R1 in die Speicherstelle auf die P0 zeigt.

B[P0] = R1; speichert die untersten 8Bit von R1 in die Speicherstelle auf die P0 zeigt.

c) Was bewirkt die folgende Befehlssequenz?

P1.H = 0xFFFF; P1.L = 0x0000; R2 = [P1];

Schreibt 16 Einsen in die höheren 16Bit und 16Nullen in die unteren 16Bit von P1.

Anschließend wird der Inhalt der Speicherstelle auf die P1 zeigt in R2 geladen.

P2.H = 0xFFFF; P2.L = 0x0010; W[P2] = R2.L;

Schreibt 16 Einsen in die höheren 16Bit und 0000000000010000 in die unteren 16Bit von P2.

Anschließend werden die unteren 16Bit von R2 (16Nullen) in die Speichestelle auf die P2 zeigt geladen.

d) Erläutern Sie den Unterschied zwischen den folgenden indirekten Ladebefehlen:

R4 = W[P0] (Z); Lädt 16 Bit auf die P0 zeigt in die unteren 16Bit von R4 (Rest mit Nullen auffüllen)

R4 = W[P0] (X); Lädt 16 Bit auf die P0 zeigt in die unteren 16Bit von R4 (Rest mit Vorzeichenbits füllen)

R4.H = W[P0]; Lädt 16Bit auf die P0 zeigt in die oberen 16Bit von R4

R4.L = W[P0]; Lädt 16Bit auf die P0 zeigt in die unteren 16Bit von R4

Der Unterschied der Befehle liegt darin wohin beispielsweise 16Bit ein 32Bit Register geschrieben werden und wie mit den unberührten Bits verfahren wird.

e) Was bewirkt die folgende Befehlssequenz?

P1.H = 0xFFFF; P1.L = 0x0010; R7.H = W[P1];

Schreibt 16 Einsen in die höheren 16Bit und 0000000000010000 in die unteren 16Bit von P1.

Anschließend wird der Inhalt der Speicherstelle auf die P1 (16Bit) in die höheren 16Bit von R7 geschrieben.

P1.H = 0xFFFF; P1.L = 0x0000; [P1] = R7;

Schreibt 16 Einsen in die höheren 16Bit und 16 Nullen in die unteren 16Bit von P1.

Anschließend wird der Inhalt von R7 in die Speicherstelle auf die P1 zeigt geschrieben.

f) Erläutern Sie den Unterschied zwischen den folgenden indirekten Ladebefehlen:

R5 = [P0]; Schreibe Inhalt der Speicherstelle auf die P0 zeigt nach R5

R5 = [P0++]; Wie davor, danach wird P0 um 4 Inkrementiert

R5 = [P0 + 2]; Lädt 32Bit Wert der Speicherstelle die 2Byte nach der in P0 gespeicherten Adresse liegt.

**1.7 Aufgabe**

Studieren Sie die Shifter-Befehle des DSP im Kapitel 14 des o.g. Manuals.

a) Was ist der Unterschied zwischen einem arithmetischen und einem logischen Shift? Wie wird syntaktisch dazwischen unterschieden?

Bei einem arithmetischen Shift wird das ursprüngliche Vorzeichen des Registerinhalts beibehalten.

Bei einem logischen Shift wird keine Rücksicht auf das Vorzeichen des Registerinhalts genommen.

Beispiel arithmetischer Rechtsshift „>>>“ und logischer Rechtsshift „>>“.

b) Im 16-Bit-Register R0.L stehe der Wert 20dez = 0014hex = 0x00142. Durch welche Shifter-Befehle lässt sich dieser Wert verdoppeln bzw. halbieren?

Solange kein Überlauf auftritt entspricht ein Shift nach Rechts einer Halbierung des Wertes und ein Shift nach Links einer Verdopplung des Wertes.

R0.L=R0.L>>1; R0.L=R0.L<<1;

c) Im 16-Bit-Register R0.L stehe der Wert -20 = 0xFFEC. Durch welche Shifter-Befehle lässt sich dieser Wert verdoppeln bzw. halbieren?

Mit einen arithmetischen Rechts- beziehungsweise Linksshift.

R0.L=R0.L>>>1; Freiwerdende Bits werden mit Vorzeichenbits aufgefüllt (Halbieren)

R0.L=R0.L<<1; (Verdoppeln)

d) Im 16-Bit-Register R1.L stehe der Wert 10000 = 0x2710. Welcher Wert (dezimal bzw. hexadezimal) steht jeweils in dem Register R2.L nach den Befehlen

R2.L = R1.L << 2; Multiplikation mit 4. (-25536 bzw. 0x9C40) logisch

R2.L = R1.L << 2 (S); Multiplikation mit 4. (40000 bzw. 0x7FFF) arithmetisch 🡪Ursprünglich positiver Registerinhalt wird bei Überlauf auf größte positive Zahl gesetzt

e) Im 16-Bit-Register R1.L stehe der Wert -10000 = 0xD8F0. Welcher Wert (dezimal bzw. hexadezimal) steht jeweils in dem Register R2.L nach den Befehlen

R2.L = R1.L << 2; (25536 bzw. 0x63C0) logisch

R2.L = R1.L << 2 (S); (-40000 bzw. 0x8000) arithmetisch 🡪 Ursprünglich negativer Registerinhalt wird bei Überlauf auf größte negative Zahl gesetzt