Technische Universität Darmstadt
FG Eingebettete Systeme und
ihre Anwendungen (ESA)

Prof. Dr. Andreas Koch

Embedded Systems & Applications

"Eingebettete Prozessorarchitekturen"
Aufgabe 3: Rekonfigurierbare Prozessoren
Abgabe bis zum 09.02.2011, 23:59 MET

1 Einleitung

Der Schwerpunkt dieses Übungsblattes sind Experimente mit rekonfigurierbaren Prozessoren am Beispiel der Stretch S5000-Serie. Um die Rekonfigurationseigenschaft tatsächlich auszunutzen, wird hier ein erweitertes Programmbeispiel bearbeitet, das sowohl Überblenden als auch Chroma-Keying beherrscht. Abhängig von der vom Benutzer via Kommandozeilenparameter eingestellten Betriebsart soll ein von Ihnen implementierter Satz von Stretch Extension Instructions (EI) bereitstehen.

2 Vorinstalliertes Material

Im Unterverzeichnis `phase3-material` Ihres HOME-Bereiches ist ein Beispiel-Projekt mit den zur Bearbeitung dieser Aufgabe erforderlichen Dateien bereits vorinstalliert.

3 Entwicklungsumgebung

Wechseln Sie nach dem Anmelden auf einem der Arbeitsplatzrechner der FG ESA durch ein `cd` Kommando in das Unterverzeichnis `phase3-material`. Die Entwicklungsumgebung, die bereits in der Vorlesung demonstriert wurde, kann nun mit dem Kommando

```
st-ide
```

gestartet werden. Die sehr umfangreiche Online-Hilfe (Menü Help bzw. Taste F1 tippen) enthält nicht nur Hinweise zur Benutzung des Werkzeugs (z.B. in Form eines Leitfadens zum `rgb2ycc`-Beispiel in der Rubrik "S5000 Development Tools Quick Start Guide"), sondern auch eine vollständige Beschreibung von Stretch C (in "SCP Architecture Reference", Abschnitt "Defining and Using Extension Instructions").

4 Beispielprojekt

Öffnen Sie nun das Projekt ChromaBlend durch File, Open Project und Auswählen der Datei `ChromaBlend.spf`. Nun finden sich im Abschnitt Source Files der Projektübersicht (Bereich links) die C-Quellen der kombinierten ChromaBlend-Anwendung. Sie lassen sich durch einen Doppelklick öffnen. Ein Kommentarkopf erklärt die neue Funktionsweise des neuen Programmes.

Das Projekt definiert vier Targets: zwei für die verschiedenen Betriebsarten Überblenden (mit Blending im Namen) und Chroma-Keying (mit ChromaKey im Namen) in je zwei Varianten. Eine für die schnelle Native-Ausführung auf dem Multi-GHz Prozessor des Arbeitsplatzrechners (gekennzeichnet durch ein

NAT im Namen) und eine weitere für die Ausführung auf dem simulierten S5000-Chip (gekennzeichnet durch ein SIM im Namen). Die Blending und ChromaKey-Varianten sind effektiv dasselbe Programm (das ja beide Operationen beherrscht) und unterscheiden sich nur durch die Kommandozeilen. Trotzdem muß jede der beiden Varianten vor Ausführen gebaut werden. Das Build-System erkennt allerdings, dass in beiden Fällen die gleiche ausführbare Datei verwendet wird und kürzt den zweiten Build ab.

5 Experimente im Native-Mode

Wählen Sie also nun nacheinander im Target-Pop-Up-Menü (über der links gelegenen Projektübersicht) nacheinander die verschiedenen Targets an und veranlassen Sie das Bauen der ausführbaren Programme durch Build, Build (oder einfaches Tippen der Taste F7). In allen vier Fällen sollte dabei nach erfolgreichem Abschluss ein `Build succeeded` angezeigt werden.

Testen Sie nun zunächst die Native-Version des Programms auf Funktionsfähigkeit. Wählen Sie dazu aus dem Target-Pop-Up-Menü das gewünschte NAT-Target aus und selektieren dann Debug, Run und klicken den Run-Knopf. Hinweis: Falls der Menüpunkt Run inaktiv grau dargestellt sein sollte, haben Sie vergessen, das entsprechende Target zu bauen. Nun sollte das Programm in der gewünschten Betriebsart ablaufen (siehe die oben angezeigte Kommandozeile). Nach erfolgreicher Ausführung sollte in diesem Run-Dialog dann die Meldung `Done.` auftauchen.

Im Unterverzeichnis `Images` des Projektverzeichnisses sollte nun eine der Betriebsart entsprechende Bilddatei geschrieben worden sein: `blendedimage.ppm` beim Überblenden, `keyedimage.ppm` beim Chroma-Key-Verfahren. Beurteilen Sie mit den üblichen Werkzeugen (siehe letztes Aufgabenblatt), ob die Ausgabebilder korrekt sind.

6 Messungen an SIM-Versionen (2 Punkte)

Führen Sie nun nach dem gleichen Schema die SIM-Targets aus. Überprüfen Sie, ob die Ausgabebilder neu geschrieben wurden (Zeitstempel mit Unix-Kommando `ls -l` anzeigen lassen) und ob sie die korrekten Daten enthalten.

Untersuchen Sie nun die Ausführungszeiten, die Ihnen in bekannter Manier des Tensilica-Simulators, der ja auch von Stretch verwendet wird, nach der Programmausführung angezeigt werden: Wieviele Takte braucht das Programm in jedem der beiden Betriebsmodi insgesamt? In beiden Fällen werden ja dieselbe Anzahl von Pixeln verarbeitet (120000), aber warum ist Betriebsart Chroma-Keying deutlich langsamer?

Erstellen Sie nun Profile der beiden SIM-Targets. Dazu klicken Sie zu jedem der Targets einmal auf das Profile-Icon (ein in der Icon-Leiste recht mittig gelegener grüner Kreis). Nach einer knappen Minute können Sie den vorgeschlagenen Namen für das fertige Profil bestätigen und die Daten, die im Unterordner Profile der Projektübersicht dargestellt werden, sichten.

Beantworten Sie anhand der beiden Profile nun die Fragen, wieviele Takte absolut jeweils in den Funktionen `blend` und `chromakey` verbracht werden und welchen relativen Anteil an der jeweiligen Gesamtausführungszeit dies ausmacht.

7 Beschleunigung der Ausführung mit Stretch-C EIs (6 Punkte)

Beschleunigen Sie unter Ausnutzung der in der Vorlesung vorgestellten Konzepte das Programm mit seinen *beiden* Betriebsarten nun durch das Entwickeln mehrerer EIs (Sie brauchen mindestens zwei!). Dazu können Sie im Unterordner Stretch C Files der Projektübersicht durch Rechtsklick neue `.xc`-Dateien mittels Add New File ... anlegen.

Sie sollten je Betriebsart eine getrennte Stretch-C-Datei anlegen. Jede dieser Dateien kann aber durchaus mehrere Instruktionen definieren. Modifizieren Sie dann auch das C-Programm der Anwendung so, dass Ihre EIs korrekt benutzt werden.

Tipp: Vergessen Sie nicht, in Ihren C und Stretch-C Dateien die entsprechenden Header-Dateien `libei.h` und `stretch.h` mittels `#include` einzulesen. Hier ggf. einen Blick in die Vorlesungsfolien werfen!

Testen Sie die Funktionsfähigkeit Ihres ISEF-unterstützten Programmes zunächst in der schnellen NAT-Betriebsart: Wenn Sie nun das entsprechende Target bauen, werden die neu angelegten Stretch-C-Dateien jetzt mitübersetzt. Hinweis: Falls sich der Compiler über fehlende Include-Dateien beklagt, wählen Sie *vor* dem eigentlichen Build jede der Stretch-C Dateien *einzeln* aus und lösen durch Rechtsklick auf Compile eine getrennte Kompilierung aus.

Wenn Sie das Target erfolgreich bauen konnten, führen Sie es mit Debug, Run aus und prüfen wieder die neu geschriebenen (Zeitstempel!) Bilddateien in beiden Modi auf Korrektheit.

Falls Ihre EIs nicht auf Anhieb zufriedenstellend funktionieren sollten, können Sie (wie in der Vorlesung gezeigt) im Native-Mode auch Ihre Stretch-C-Programmteile durch Einzelschritte debuggen. Setzen Sie dazu einfach einen Breakpoint an den Anfang der zu untersuchenden EI durch Klicken in die ganz linke Spalte des Stretch-C-Quellcodes (hier taucht dann ein kleines Stop-Icon auf) und starten Sie den Debugger durch Klick auf das Debug-Icon (grüner Play-Pfeil der Icon-Leiste). Die Programmausführung sollte nun beginnen, aber an dem gesetzten Breakpoint anhalten. Jetzt können Sie durch Klicken von Step Over (mittleres der drei Step-Icons in der Icon-Leiste, zwischen Debug und Profile-Bereich gelegen) die einzelnen EI-Schritte durchgehen. Hinweis: Falls Sie versehentlich Step Into angeklickt haben sollten (Ihnen also als Quelltext ein unbekanntes C++-Ungetüm entgegenspringt), sollten Sie durch einen beherzten Klick auf Step Out wieder in heimische Stretch-C-Gefilde zurückfinden. Die Werte von Variablen können Sie sich während des Einzelschrittbetriebs durch Debug, Windows, Watch, Watch 1 anzeigen lassen: Tippen Sie dazu in dem nun rechts erscheinenden Watch-Bereich einfach die Namen der Sie interessierenden Variablen ein. Dabei sind auch Arrays möglich, deren Elemente dann durch Klicken des +-Icons neben dem Variablennamen aufgeklappt werden können. Von Schritt zu Schritt geänderte Werte werden in Rot hervorgehoben. Hinweis: Die Schrittgeschwindigkeit des Debuggers wird anscheinend langsamer exponentiell abhängig von der Anzahl der im Watch-Bereich dargestellten Elemente. Es empfiehlt sich also, die Anzahl der dargestellten Arrays, deren Elemente jeweils *einzeln* zählen, so gering wie möglich zu halten. Weitere Informationen zum Debugger entnehmen Sie falls nötig bitte der Online-Dokumentation (angefangen mit dem oben bereits erwähnten "Quick Start" aber auch z.B. "Stretch IDE 2009.06", Abschnitt "Building, Debugging, and Profiling", Unterabschnitt "Debugging").

Wenn Ihre EIs jetzt im Native-Mode korrekt übersetzt werden und auch funktionieren, wählen Sie im Target-Pop-Up-Menü ein SIM-Target, um nun auch die genaue Simulation auf dem "echten" Stretch S5000-Chip zu ermöglichen. Bauen Sie dann mit diesem neuen Target das Projekt. Dieser Lauf dauert etwas länger als der im Native-Mode, da nun die Hardware auf der ISEF auch platziert und verdrahtet wird (im Native-Mode wurde nur ein Verhaltensmodell der ISEF in C++ erstellt, aber ohne die ISEF-Konfiguration detailliert zu berechnen).

Hier sollte eigentlich nichts mehr schief gehen. Ausnahmen: Sie sind mit der Hardware zu großzügig umgegangen und Ihre Instruktionen passen nicht mehr auf je eine ISEF. In diesem Fall sollten Sie sich Ihren Stretch-C-Code nocheinmal anschauen. Dafür hilfreich ist ein Blick in die Berichtsdatei des Stretch-C-Compilers, die Sie sich zur Lösung der Aufgaben ohnehin ansehen müssen (siehe unten).

Jetzt können Sie Ihre EI-versehene und (hoffentlich) beschleunigte Anwendung genauer untersuchen. Lassen Sie zunächst jede der beiden Betriebsarten im SIM-Modus laufen und geben Sie nun die zur gesamten Ausführung benötigten Takte an (aus der Konsolen-Ausgabe des Tensilica-Simulators). Ermitteln Sie nun mittels des Profilers (ebenfalls wieder in beiden Betriebsarten) die absoluten Taktanzahlen und den relativen Anteil an der Ausführungszeit der beiden Funktionen `blend` und `chromakey` und geben Sie diese Messwerte an.

Als letztes sind noch Angaben über die auf dem ISEF realisierte Hardware Ihrer EIs interessant. Öffnen Sie dazu im Unterverzeichnis `SIM` die Ihren Stretch-C Dateien entsprechenden Berichtsdateien mit der Endung `.xr`. Hinweis: Sie müssen beim Öffnen mittels File, Open unten im Dateiauswahldialog aus dem File Type-Pop-Up-Menü die Option All files auswählen, sonst werden Ihnen die `.xr`-Dateien nicht angezeigt.

Geben Sie zu jeder Ihrer EIs die benötigte Fläche in AUs und MUs an (aus dem Abschnitt `Final resource usage report`), sowie die folgenden Charakteristika zum Zeitverhalten: Die Latenz jeder Instruktion (maximale Anzahl von Takten vom Lesen zum Schreiben von WR-Registern) und den Durchsatz (maximale Anzahl von Takten zwischen Lesen und Schreiben des gleichen ER-Registers +1). Diese Daten finden

Sie am Anfang der `.xr`-Datei zu jeder Instruktion.

8 Abgabe

Zusammenfassend geben Sie für diesen Aufgabenteil ab:

1. Die Gesamtausführungszeit und Profiling-Daten der ursprünglichen reinen C-Anwendung im SIM-Modus.
2. Die kommentierten Quellcodes Ihrer Stretch-C-EIs und des darauf umgestellten C-Programms, sowie die entstandenen `.xr`-Berichtsdateien.
3. Die nun erreichte Gesamtausführungszeit in Zyklen und die absoluten und relativen Ausführungszeiten der `blend` und `chromakey`-Funktionen.
4. Den Flächenbedarf Ihrer EIs als Anzahl AUs/MUs je Konfiguration.
5. Die Latenzen und Durchsätze der einzelnen EIs anhand der Angaben aus der Berichtsdatei.

Für den Prosa-Teil der beiden Aufgaben dieses Blattes können Sie zusammen eine PDF-Datei abgeben, die `.xc`, `.c` und `.xr` Dateien hängen Sie aber bitte getrennt an die Abgabe-Mail an!

Ihre Abgaben tätigen Sie fristgerecht in Form einer E-Mail an die Adresse

`epa@esa.informatik.tu-darmstadt.de`

mit dem Betreff "Gruppe N Aufgabe 3", wobei N Ihre Gruppennummer ist. Diese E-Mail hat als Anhänge die in den Teilaufgaben geforderten Komponenten.

Eventuelle Nacheinreichungen zur 2. Aufgabe (Tensilica) schicken Sie als *gesonderte* E-Mail an die gleiche Adresse, allerdings mit dem Betreff "Gruppe N Aufgabe 2a". Der Inhalt muß dann den Vorgaben aus dem 2. Aufgabenblatt genügen.

9 Hinweise zum Thema Plagiarismus

Im Rahmen dieser Veranstaltung wird eine vorher festgelegte Arbeitsgruppe bewertet. Fremde Code-Bibliotheken außer den vom Dozenten zur Verfügung gestellten dürfen Sie *nicht* verwenden! Zusammenarbeit über Gruppengrenzen hinweg ist in Form der Diskussion von Lösungs*ideen* erlaubt. Es dürfen aber *keine* Artefakte wie Programm-Code, Dokumentationsteile (Text, Zeichnungen, Messergebnisse) oder ähnliches ausgetauscht werden.

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Mit der Abgabe einer Lösung (Hausaufgabe, Programmierprojekt, etc.) bestätigen Sie, dass Ihre Gruppe die alleinigen Autoren des gesamten Materials sind. Weiterführende Informationen zu diesem Thema finden Sie unter `http://www.informatik.tu-darmstadt.de/Plagiarism`.