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

Prof. Dr. Andreas Koch

"Eingebettete Prozessorarchitekturen"
Aufgabe 2: Konfigurierbare Prozessoren
Abgabe bis zum 22.12.2006, 23:59 MET

1 Einleitung

In diesem Übungsblatt befassen Sie sich mit konfigurierbaren Prozessoren, speziell der Technologie der Fa. Tensilica. Hilfreiche Informationen finden sich dazu in den Tensilica Lab Notes 3 und 6, die Ihnen mit der Materialsammlung zur Verfügung gestellt werden, sowie in der vollständigen Dokumentation, die im Pfad

`/opt/cad/tensilica/tensilica-2006.5/XtDevTools/downloads/RA-2006.5/docs`

installiert ist. Die beiden Lab Notes behandeln das `ByteSwap`-Beispiel, das bereits in der Vorlesung am 29.11.06 demonstriert wurde. Sie sind hilfreich für das Nachvollziehen gängiger Arbeitsschritte, die Sie auch für das Lösen der neuen Aufgaben benötigen werden. Einige Abweichungen zu der in den Lab Notes vorgestellten Windows-Version der Software werden Abschnitt 2 noch genauer erläutert.

Die Materialsammlung umfasst weiterhin einen Xtensa Workspace zum Import in die Tensilica `xplorer`-Entwicklungsumgebung. Dieser enthält in getrennten Projekten neben den Quellen des `ByteSwap`-Beispiels auch leicht modifizierte Quellen der Bildüberblendungsanwendung (Halbierung der bearbeiteten Bildgröße).

Die Materialsammlung zu diesem Übungsblatt findet sich auf der EPA-Web-Seite und ist bereits im HOME-Bereich Ihrer Arbeitsgruppe auf unseren Rechnern im Unterverzeichnis **`phase2-material/`** vorinstalliert.

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

`epa06@esa.informatik.tu-darmstadt.de`

mit dem Betreff "Gruppe N Aufgabe 2", wobei N Ihre Gruppennummer ist. Diese E-Mail hat als Anhänge die in den Teilaufgaben geforderten Komponenten.

2 Einrichten der Arbeitsumgebung

1. Starten Sie die Tensilica Entwicklungsumgebung durch das Kommando `xplorer`, legen Sie einen neuen Workspace an (Pfadname zu einem neuen Verzeichnis unterhalb Ihres HOME-Bereichs eingeben!) und wechseln Sie in die Workbench-Ansicht (Icon unten auf dem Startbildschirm klicken).

2. Importieren Sie in den neuen Workspace (Menu File, Punkt Import) nun den Xtensa Xplorer Workspace `phase2.xws` aus der Materialsammlung (mit Browse den Pfadnamen im `phase2-material` Verzeichnis auswählen). Selektieren Sie alle enthaltenen Projekte (ByteSwap und Blending) und ignorieren Sie die Warnung, dass Sie keinen XPG Zugang haben. Nun sollten im C/C++ Projects-Bereich der Workbench (links mittig) die `ByteSwap` und `Blending` Projekte auftauchen.

3. Die Quelldateien finden sich innerhalb der Projektordner. Für das Blending-Beispiel existiert auch noch ein Unterordner `Images`, in dem die Ein- und Ausgabebilder für die Tests liegen. Für dieses Beispiel ist die Kommandozeile mit den Argumenten (Dateinamen, Überblendfaktor) für die Funktionen Run, Profile und Debug bereits fertig im Workspace eingetragen, sie können also direkt benutzt werden (den Pfeil anklicken um das Pop-Up-Menu zu öffnen, dann die gewünschte Ausführung auswählen). Die anderen Unterordner im Projects-Bereich (`bin`, `Binaries`, etc.) sind für Sie nicht relevant.

4. Für Ihre Experimente müssen Sie nun die zu verwendende Prozessorkonfiguration anwählen. Klicken Sie dazu mit der rechten Maustaste im System Overview-Bereich (links unten) den Ordner Configurations an und wählen Sie das Kommando New Configuration aus.

5. Im nun erscheinenden Dialog wählen Sie die dritte Option von oben aus ("... based on an already installed build"), klicken Next, Auto-Discover Builds, und wählen dann den nun erscheinenden einzigen vorhandenen Prozessor durch Anklicken aus.

6. Tragen Sie als neuen Namen für die Konfiguration in das Textfeld oben `epa06` ein und klicken dann auf Finish. Erlauben Sie, falls nötig, das Überschreiben von Daten in der Registry und ignorieren evtl. auftauchende Fehlermeldungen.

7. Wählen Sie dann die neue Konfiguration im Reiter "C:" (oben mittig, steht anfangs vermutlich auf "No Active Config") durch Klick auf den Pfeil nach unten aus.

Nun sollte Ihre Umgebung für die neuen Experimente mit dem `Blending`-Projekt bzw. für das Nachvollziehen des `ByteSwap`-Beispiels eingerichtet sein. Im Reiter "P:" (steht anfangs vermutlich auf "No Active Project", oben mittig) können Sie zwischen den beiden Projekten umschalten.

3 Experiment: Untersuchung der Bildüberblendung (4 Punkte)

Für diese Aufgaben muss die Einstellung der Umgebung (oben mittig) beginnen mit

P: Blending C: epa06 T: Debug

Falls dem nicht so sein sollte, nehmen Sie bitte in den einzelnen Reitern durch Klicken in den Pfeil nach unten die entsprechenden Auswahlen vor. Wenn Sie diese Einträge nicht sehen, haben Sie die Anweisungen des vorhergehenden Abschnitts vermutlich nicht ganz genau befolgt.

Übersetzen Sie das Blending-Programm mit einem Klick auf Build Active. Zum anschließenden Ausführen öffnen Sie aus dem Reiter Run durch Klick auf den Pfeil das Pop-Up-Menu und wählen die Ausführung von Run Blending – epa06 – Debug aus. Im Unterverzeichnis `Blending/Images` Ihres neu angelegten Workspaces ist nun die Datei `myoutimage.ppm` geschrieben worden, die Sie mit den aus der ersten Aufgaben bekannten Möglichkeiten ansehen können. Beachten Sie: Aus Zeitgründen (langsamer Simulator) ist die Auflösung der Bilder hier nur 400 x 300 Pixel.

Wichtig: Klicken Sie *nicht* einfach auf den Reiter Run! Zwar wird das Programm dann auch gestartet (als Auto – Blending – epa06 – Debug), allerdings *ohne* die erforderlichen Kommandozeilenparameter. Es wird also keine Blending-Operation ausgeführt und keine Ausgabedatei geschrieben!

Führen Sie nun durch Klick auf Profile ein Profiling aus. Auch hier klicken Sie auf den Pfeil und wählen aus dem Pop-Up-Menu die Operation Profile Blending – epa06 – Debug aus. Wie vorher führt ein einfacher Klick direkt auf den Reiter zu einem *fehlerhaften* Profil! Beantworten Sie nun anhand der Profiling-Ergebnisse folgende Fragen:

1. Wieviele Zyklen braucht das Programm insgesamt? Wieviele werden dabei durch Source Interlock und Taken Branches-Zyklen verschwendet? Diese Informationen finden Sie im Console-Reiter (unten mittig beim Profiling).

2. Welchen Anteil hat die Funktion `blend` an der gesamten Ausführungszeit (siehe Reiter Profile)?

3. Wählen Sie im Profile-Bereich die Funktion `blend` an und untersuchen Sie im Profile Disassembly-Bereich (rechts mittig) den kompilierten Assemblerquelltext. Geben Sie je ein Beispiel für viele verschwendete Interlock-Zyklen und Branch Delay (abgekürzt bdelay)-Zyklen an. Sie können sich diese Daten zu jeder mit dem Mauszeiger überfahrenen Instruktion anzeigen lassen. Geben Sie Ihre Antwort als Auszug aus dem Assembler-Text an! Tipp: In diesem Bereich können durch Shift-Click mehrere Instruktionen selektiert werden, und dann durch Edit, Copy und Paste in ein anderes Fenster kopiert werden. Auf diese Weise lassen sich Assembler-Bereiche in Ihr Lösungsdokument einbinden.

Schalten Sie nun den Übersetzungsmodus im "T:" Reiter von Debug auf Release um und wählen Sie aus dem Build Active-Reiter die Funktion Rebuild Active aus, die das Programm mit Optimierung (Stufe -O2) neu übersetzt. Hinweis: Ein einfacher Klick auf Build Active übersetzt das Programm *nicht* neu, da sich der Quelltext nicht geändert hat.

Erstellen Sie nun ein Profil der optimierten Lösung, indem Sie im Reiter Profile nun die Operation Profile Blending – epa06 – Release selektieren (beachten Sie das Release am Ende!). Beantworten Sie dann diese Fragen:

1. Wie haben sich die Gesamt- sowie die Interlock und Branch Taken-Zyklenanzahlen entwickelt?
2. Welchen Anteil hat `blend` nun an der Gesamtlaufzeit?
3. Gibt es immer noch soviele Instruktionen mit Interlock-Zyklen in `blend`? Wie hat sich die Anzahl der Branch Delay-Zyklen entwickelt? Wie erklären Sie diese Änderungen? Tipp: Nach Anklicken des Profile-Reiters tauchen rechts davon fünf kleine, fast unlesbare Icons auf, mit denen man die im Profil angezeigten Daten ändern kann. Die rechten beiden davon befassen sich mit Interlock und Branch Delay-Zyklen.

Alle diese Ergebnisse sollen in einem kurzen PDF-Dokument in Prosa beschrieben und kommentiert werden.

4 Experiment: Beschleunigung durch eigene TIE (4 Punkte)

Schreiben Sie nun ein oder mehrere TIE-Instruktionen, die den Ablauf des Programmes beschleunigen.

1. Legen Sie dazu in System Overview (falls nicht unten links sichtbar: auf das Symbol rechts über dem Profile Disassembly-Bereich klicken, das hat als Tool-Tip den Text C/C++ Perspective) durch Rechtsklick auf TIE Source mit New TIE File eine neue Datei an, die sie `blending` nennen (ohne `.tie`).
2. Sie werden dann gefragt, an welche Prozessorkonfiguration Sie die neuen Instruktionen binden möchten. Wählen Sie hier none aus, sonst machen Sie sich Ihren Ursprungsprozessor kaputt! Ignorieren Sie die Warnung (Sie wissen ja, was Sie tun).
3. Stattdessen geben Sie mit Rechtsklick auf epa06_1-params in der System Overview das Kommando Clone Config from Build with TIE, mit dem Unterpunkt blending.tie. So legen Sie eine *Kopie* namens `epa06_tie` (Namen in Textfeld eintippen!) des Prozessors an und erweitern diese mit dem TIE. Wählen Sie im C: Reiter (steht vermutlich noch auf epa06) nun den erweiterten Prozessor als Ziel aus (also C: epa06_tie).
4. Nun hat sich im Hauptfeld die TIE-Ansicht geöffnet. Da Sie noch keinen Quelltext eingegeben haben, sind die Flächenabschätzungsdaten noch leer. Unter dem Bereich klicken Sie nun auf den Reiter TIE source und geben Ihren Quelltext ein. Speichern Sie diesen ab (z.B. mit Control-S), und wechseln Sie dann wieder mit dem unteren Reiter TIE overview in die Flächenabschätzung. Nach einem kurzen Übersetzungsvorgang bekommen Sie hier die aktuellen Flächenabschätzungen zu sehen.
5. Sollte Ihre TIE-Beschreibung fehlerhaft sein, wird der unten gelegene Reiter Problems fettgedruckt. Wenn Sie ihn anwählen, bekommen Sie dann eine Liste von Fehlern. Durch Doppelklick auf die Fehlermeldung springen Sie an die entsprechende Stelle im TIE-Quelltext und können den Fehler korrigieren.

Wenn die TIE korrekt übersetzt wird, können Sie daran gehen, den C-Quelltext von `blend` so umzuschreiben, dass die TIE benutzt wird. Speichern Sie die Datei ab (Control-S) und übersetzen Sie sie wieder mit Build Active oder Rebuild Active (macht hier keinen Unterschied). Wenn Sie dabei Fehler bekommen, dass Ihre neuen TIE-Instruktionen unbekannt sind (genauer: `undefined reference`), haben Sie vermutlich vergessen, die ausgewählte Prozessorkonfiguration von "C: epa06" auf "C: epa06_tie" umzustellen, und/oder vergessen am Anfang der von Ihnen nun auf TIEs umgestellen C-Datei die C-Definitionen der TIEs einzubinden:

```
#include <xtensa/tie/blending.h>
```

Wenn Sie diesen ganzen Prozess im Griff haben, entwickeln Sie nun die TIEs zur Beschleunigung der `blend` Funktion. Beachten Sie dabei, dass der `epa06`-Prozessor (und damit auch Ihre Kopie `epa06_tie`) nur *eine* Load-Store-Unit hat.

Testen Sie Ihren beschleunigten Prozessor und die modifizierte Software durch Run, wobei Sie aus dem Pop-Up-Menu nun Run Blending – epa06_tie – Debug auswählen (beachten Sie: hier epa06_tie statt epa06). Tipp: Vergleichen Sie das von Ihrem TIE-Prozessor erzeugte Bild mit den Referenzdaten durch das Unix-Kommando

```
cmp myoutimage.ppm outimage.ppm
```

Falls beide Bilder gleich sind, kommt dieses Kommando ohne weitere Meldungen zurück. Falls es Unterschiede gibt (Ihr Prozessor also nicht richtig rechnet), wird eine Fehlermeldung der Art

```
myoutimage.ppm outimage.ppm differ: char 69, line 5
```

ausgegeben.

Nach erfolgreichen Tests untersuchen Sie das Programm wieder mittels Profiling. Auch hier stellen Sie bitte sicher, dass Sie aus dem Pop-Up-Menu des Reiters Profile nun das korrekte Profile Blending – epa06_tie – Release ausführen!

Zusammenfassend geben Sie für diesen Aufgabenteil ab:

1. Die kommentierten Quellcodes Ihrer TIE-Erweiterung und des darauf umgestellten C-Programms (also `blending.tie` und `blending.c`)
2. Den Flächenbedarf Ihrer TIE-Erweiterung als Anzahl Gatter je Instruktion bzw. Register.
3. Die nun erreichte Gesamtausführungszeit in Zyklen sowie die Anzahl von Interlock und Branch Delay-Zyklen in der `blend`-Funktion.

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

5 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`.