Entwurf und Synthese von Eingebetteten Systemen

Sommersemester 2017

Übungsblatt 8

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Lehrstuhl Eingebettete Systeme
Prof. Dr. O. Bringmann

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Montag, den 3. Juli 2017, 12:00

Die Abgabe der Lösungen erfolgt über das ILIAS-System als gepacktes Archiv (ZIP/TAR.GZ).
Dieses ZIP enthält die schriftlichen Antworten auf Fragen, die Quellcodes in Dateiform (nicht als Listing im PDF), und die Simulationsergebnisse als VCD-Dateien.

Abgabefrist: Montag, den 10. Juli 2017, 12:00 Uhr

Aufgabe 1: High Level Synthese: Ablaufplanung nach ASAP und ALAP [9 Punkte]

Gegeben seien die folgende C-Funktion:

```
int calculate(int a, int b, int c, int d, int e, int f, int g) {
  Y = ((a+b*d-f)*g)*(c - f)+(f-g-(d+c)+(a+b));
  return Y*f;
}
```

Die arithmetischen Operationen werden wie folgt auf Ressourcentypen abgebildet:

Operation	Ressourcentyp	Ausführungszyklen
+, -	ALU	1 Takt
*	Multiplizierer	1 Takt

(a) Stellen sie den Datenflussgraphen der Funktion calculate() grafisch dar.

(b) Stellen sie den Problemgraphen (Sequenzgraphen) grafisch dar.
Geben sie zudem das algebraische Modell $G(V, E)$ an.

(c) Erstellen sie für G den Ablaufplan nach dem ASAP-Verfahren.

(d) Erstellen sie für G den Ablaufplan nach dem ALAP-Verfahren.

(e) Bestimmen sie die Mobilität $\mu(v)$ aller Operationen $v \in V$.
Was kann über Operationen mit Mobilität 0 bzgl. des Freiheitsgrades bei der Optimierung gesagt werden?

Aufgabe 2: High-Level-Synthese mit Xilinx Vivado HLS [11 Punkte]

Im ILIAS-Downloadbereich finden sie die Implementierung einer abgespeckten ALU in C[1]. Die ALU hat zwei 16 Bit breite Operanden sowie einen Operator-Eingang (Opcode) zur Selektion der jeweiligen Operation. Die ALU unterstützt die folgenden Operationen:

- Multiplikation (Opcode 0)
- Subtraktion (Opcode 1)
- Addition (Opcode 2)
- Ganzzahl-Division (Opcode 3)

(a) Führen sie eine High-Level-Synthese mit dem Tool **Xilinx Vivado HLS** durch! Das Target-Device sei hierbei das XC7VX330TFFG1761-3 aus der Virtex-7-Reihe. Spezifizieren sie eine Taktperiode von 20ns.

Erwartete Abgabe: Generierte VHDL-Dateien, Report alu_csynth.rpt

(b) Beschreiben sie kurz die Funktion der Ports der generierten Entität `alu`[2].

(c) Wie hoch ist der gesamte geschätzte Flächenverbrauch (Anzahl Lookup-Tables, Anzahl Flip-Flops)?

(d) Wie groß ist die geschätzte Taktperiode des Designs?

(e) Wie hoch ist die Anzahl der Latenzzyklen der einzelnen Operationen (Multiplikation, Subtraktion, Addition, Division)?

(f) Führen sie nun den kompletten FPGA-Implementierungsflow inklusive der RT- und Logik-Synthese sowie der Implementierung mit dem Tool **Xilinx Vivado** durch! Das Target-Device sei hierbei das XC7VX330TFFG1761-3 aus der Virtex-7-Reihe. Spezifizieren sie eine Taktperiode von 20ns.

Erwartete Abgaben: Post-Implementation Utilization Report *alu_utilization_placed.rpt*, Post-Implementation Timing Report *alu_timing_summary_routed.rpt*)

(g) Wie hoch ist der gesamte Flächenverbrauch (Anzahl Look-up-Tables, Anzahl Flip-Flops) der FPGA-Implementierung? Warum unterscheiden sich die Werte gegenüber den Werten aus Aufgabe (c)?

(h) Erfüllt ihre FPGA-Implementierung das spezifizierte Timing-Constraint? Wie hoch dürfte die Taktfrequenz maximal sein? Vergleichen sie die Taktfrequenz mit der in der HLS abgeschätzten Taktperiode!

Anleitungen zur Durchführung der High-Level-Synthese und der FPGA-Implementation mit Vivado (HLS) finden sie im Anhang dieses Übungsblatts.

[1] Sie finden diese Implementierung auch im AFS unter /afs/wsi/es/lehre/ESES

[2] Die nicht durch die C-Funktion definierten generierten Ports sind hier dokumentiert: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_1/ug871-vivado-high-level-synthesis-tutorial.pdf

Tutorial: Durchführung einer High-Level-Synthese mit Xilinx Vivado HLS

Die Xilinx Vivado Suite ist in zwei grobe Einheiten unterteilt: Vivado HLS ist zuständig für die High-Level-Synthese, Vivado ist zuständig für den weiteren FPGA-spezifischen Ablauf, d.h. umfasst die Logiksynthese sowie die Implementierungsphase (Mapping, Placing, Routing) und die Bitstream-Generierung.
Um mit Vivado HLS eine High-Level-Synthese durchzuführen, d.h. ein in C beschriebenes Programm in die Register-Transfer-Ebene zu synthetisieren, müssen folgende Schritte durchgeführt werden:

1. Loggen sie sich auf einem der Übungsrecher mit X11-Forwarding ein.

2. Geben sie dort folgende Befehle ein:

   ```
   setenv VIVADO_BIN "/afs/wsi/es/tools/xilinx/Vivado2016.02/Vivado/2016.2/
      bin"
   setenv VIVADO_HLS_BIN "/afs/wsi/es/tools/xilinx/Vivado2016.02/Vivado_HLS
      /2016.2/bin"
   setenv LM_LICENSE_FILE "1701@menelaos"
   set path = ($path $VIVADO_BIN $VIVADO_HLS_BIN )
   ```

   Alternative können sie die Befehle auch in einer Datei `vivado.csh` speichern und mit Hilfe des folgenden Befehls einbinden:

   ```
   source vivado.csh
   ```

3. Starten sie Vivado HLS:

   ```
   vivado_hls
   ```

4. Erstellen sie ein neues Projekt: **File** → **New Project**

   Wählen sie einen Projektnamen (beliebig, z.B. testproject) aus.

   Fügen sie die C-Dateien hinzu, die synthetisiert werden sollen und geben sie den Namen der Funktion an, die später das Top-Modul implementieren soll.

   Spezifizieren sie eine gewünschte Taktperiode in `ns`.

   Selektieren sie in **Part Selection** das gewünschte Target Device (Ziel-FPGA).

5. Prüfen sie im Projektexplorer links, ob das erstellte Projekt alle Source-Dateien erfasst hat und unter der erzeugten Solution zwei Constraint-Files `script.tcl` und `directives.tcl` erzeugt wurden.

6. Starten sie die High-Level-Synthese über das Menü mit **Solution** → **Run C Synthesis** → **[ihre Solution]**.

7. Sollte die Synthese erfolgreich gewesen sein, so erhalten ie einen Synthesereport, andernfalls werden sie im Meldungsfenster unten entsprechende Fehlermeldungen erhalten.

   Hinweis: Neben dem Synthesereport erhalten sie in der Perspektive **Analyse** weitere Informationen hinsichtlich der erzeugten FSM.

   Den Synthesereport finden sie auch in der Datei `solution1/syn/report/[projname]_csynth.xml`.

Tutorial: FPGA-Implementierung mit Xilinx Vivado

Um mit Vivado die FPGA RT-Synthese und Implementierung durchzuführen, d.h. ein zuvor modelliertes oder über die High-Level-Synthese generiertes RT-Modell in eine Netzliste zu überführen, zu mappen, platzieren und verdrahten, sind folgende Schritte notwendig.

1. Loggen sie sich auf einem der Übungsrecher mit X11-Forwarding ein.
2. Geben sie dort folgende Befehle ein:

   ```
   setenv VIVADO_BIN "/afs/wsi/es/tools/xilinx/Vivado2016.02/Vivado/2016.2/
      bin"
   setenv VIVADO_HLS_BIN "/afs/wsi/es/tools/xilinx/Vivado2016.02/Vivado_HLS
      /2016.2/bin"
   setenv LM_LICENSE_FILE "1701@menelaos"
   set path = ($path $VIVADO_BIN $VIVADO_HLS_BIN )
   ```

   Alternative können sie die Befehle auch in einer Datei `vivado.csh` speichern und mit Hilfe des folgenden Befehls einbinden:

   ```
   source vivado.csh
   ```

3. Starten sie Vivado:

   ```
   vivado
   ```

4. Erstellen sie ein neues Projekt: **File** → **New Project**

   Wählen sie einen Projektnamen (beliebig, z.B. testproject-fpga) aus.

   Selektieren sie den Projekttyp **RTL Project**.

   Fügen sie die VHDL-Dateien hinzu, die synthetisiert werden sollen.

   Selektieren sie in **Default Part** das gewünschte Target Device (Ziel-FPGA).

5. Erstellen sie ein neues Constraint-File zur Spezifikation der Taktperiode. Gehen sie dazu auf **Add Sources**. Wählen sie **Add or Create Constraints**. Erstellen sie via **Create File...** ein neues XDC-Constraint-File mit beliebigem Namen (z.B. clock.xdc).

   Öffnen sie dieses Constraints-File und fügen sie folgende Zeile hinzu:

   `create_clock -period CLKPERIOD -name system_clock [get_ports CLKPORT]`

   Ersetzen sie `CLKPORT` hierbei durch den Clock-Port ihres Designs.

   Ersetzen sie `CLKPERIOD` durch die Taktperiode in Nanosekunden.

6. Starten sie die Synthese mit **Run Synthesis**.
7. Sie sollten ein Fenster **Synthesis successfully completed** erhalten. Wählen sie **View Reports**. sie sehen nun die Post-Synthesis Reports.
8. Starten sie die Prozesse Mapping, Placing und Routing (Implementierung) mit **Run Implementation**.
9. Sie sollten ein Fenster **Implementation successfully completed** erhalten. Wählen sie **View Reports**. Sie sehen nun die Post-Implementation Reports.

Hinweise: Sie finden die Post-Synthesis Reports auch im Ordner `[projname].runs/synth_1/` sowie die Post-Implementation Reports in `[projname].runs/impl_1/`.