Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Lehrstuhl für Technische Informatik
Prof. Dr. Manfred Schimmler
Dipl.-Inf. Lars Wienbrandt

FPGA-Entwurf mit VHDL

Sommersemester 2011

Serie 10

Abgabe: bis Fr. 08.07. um 10 Uhr im Schrein oder per Mail an lwi@informatik.uni-kiel.de.

Aufgabe 1

SHA-1 ist ein kryptologischer Hash-Algorithmus zur Ermittlung eines Prüfwertes beliebiger Daten. Seit 2006 gilt dieser allerdings als nicht mehr sicher. Dennoch wird er in vielen Anwendungen noch immer verwendet.

Recherchieren Sie nach den Eigenschaften und der genauen Durchführung der Berechnung eines SHA-1-Hashes. Erläutern Sie diese in eigenen Worten und zeigen Sie die einzelnen Berechnungsschritte auf. Beschreiben Sie auch Datenabhängigkeiten.

10 Punkte

Aufgabe 2 – Programmieraufgabe

```
entity sha1 is
port (
  clk : in std_logic;
  sync_reset : in std_logic;
  digest_in : in std_logic_vector(159 downto 0);
  digest_init_in : in std_logic;
  msg_word_in : in std_logic_vector(31 downto 0);
  msg_new_in : in std_logic;
  digest_out : out std_logic_vector(159 downto 0);
  digest_new_out : out std_logic;
  busy_out : out std_logic
);
end sha1;
```

Implementieren Sie eine VHDL-Einheit, die die Berechnung eines SHA-1-Hashes eines vollständigen 512*bit*-Datenblocks durchführt. Die Eingabe des Datenblocks soll sequentiell in 32*bit*-Datenworten durchgeführt werden. Dabei können die Datenworte in direkt aufeinanderfolgenden Takten eingespeist werden, welches allerdings nicht die Voraussetzung sein soll.

Nachdem der 512*bit*-Datenblock eingelesen wurde, soll ein `busy`-Signal signalisieren, dass der Hashwert berechnet wird. Ist der Hashwert berechnet, so wird dies ebenfalls durch ein weiteres Signal für einen Takt lang signalisiert.

Da zur Berechnung eines SHA-1-Hashes einer Nachricht bestehend aus mehreren 512*bit*-Datenblöcken der zuvor berechnete Hash rekursiv weiterverwendet wird, soll der initiale Hashwert Ihrer Einheit über einen separaten Eingang extern gesetzt werden können.

Verwenden Sie als Zielarchitektur einen Spartan xc3s5000-4fg676. Ihre Einheit sollte mit einem Systemtakt von $50MHz$ oder höher arbeiten können. Erstellen Sie ein entsprechendes Clock-Constraint. Synthetisieren und implementieren sie Ihr Design und verifizieren Sie die Taktfrequenz.

Wieviele Hashes pro Sekunde kann Ihre Einheit berechnen?

Testen Sie Ihr Design mit einer Testbench, die die Berechnung des Hashes für

```
The quick brown fox jumps over the lazy dog
```

durchführt, wobei jedes Zeichen nach $8bit$-ASCII kodiert ist. Der zu erwartende SHA-1-Hashwert ist in Hexadezimalschreibweise:

```
2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12
```

Hinweis: Die initiale Belegung des Hashwertes erfolgt über die Testbench. Genauso erwartet Ihre Einheit einen vollständigen $512bit$-Datenblock, den Sie also von Hand vorbereiten und in Ihre Testbench eingeben müssen.

Hinweis: Auf der Homepage befindet sich eine C-like C++-Implementierung der SHA-1-Hashberechnung, die unter anderem Zwischenergebnisse der einzelnen Schritte ausgibt. Sie können diese zum Debugging Ihrer VHDL-Einheit verwenden. Der Aufruf der kompilierten Binärdatei erfolgt von der Kommandozeile mit der Nachricht als Argument (in Anführungszeichen).

Die genaue Beschreibung der Schnittstelle der VHDL-Einheit finden Sie im Folgenden:

`clk`: Der Systemtakt, mind. $50MHz$.

`sync_reset`: Taktsynchrones Reset. Bricht eine laufende Berechnung ab und macht die Einheit bereit für den Empfang eines neuen Datenblocks.

`digest_in`: Eingang des initialen Hashes.

`digest_init_in`: Setzt bei "1" den initialen Hash auf den an `digest_in` anliegenden Wert. Eine evtl. laufende Berechnung wird abgebrochen.

`msg_word_in`: Sequentieller Eingang für $32bit$-Worte des $512bit$-Datenblocks.

`msg_new_in`: Signalisiert mit "1" ein neues an `msg_word_in` anliegendes Datenwort, welches im nächsten Takt übernommen werden soll. Dieses Signal wird ignoriert, nachdem ein Datenblock eingelesen wurde und während der Hash berechnet wird.

`digest_out`: Ausgang für den berechneten SHA-1-Hashwert.

`digest_new_out`: Signalisiert mit "1" für einen Takt einen neuen validen SHA-1-Hashwert an `digest_out`.

`busy_out`: Signalisiert mit "1" nach dem Einlesen des $512bit$-Datenblocks die laufende Berechnung des SHA-1-Hashwertes.

40 Punkte

Zusatzaufgabe 3

Die Auslagerung der alleinigen Berechnung eines Hashes auf einen FPGA macht oftmals wenig Sinn.

Überlegen Sie sich ein Konzept für die Berechnung möglichst vieler SHA-1-Hashes auf der parallelen FPGA-Architektur RIVYERA S3-5000, welche Ihnen in der Vorlesung vorgestellt wurde. Machen Sie eine Performance-Abschätzung und erläutern Sie, inwieweit und unter welchen Voraussetzungen (z.B. hinsichtlich der zu hashenden Nachrichten) man diese noch verbessern könnte.

Wie lange würde Ihr Konzept benötigen, um 2^{32} Datenblöcke zu hashen – mit und ohne Annahme der oben genannten Voraussetzungen?

Nützliche Informationen zur RIVYERA S3-5000 API:

FPGAs: 128× Spartan xc3s5000-4fg676

API-Systemtakt: $50MHz$

Kommunikationsregisterbreite: $64bit$

max. Datenübertragungsrate (netto): $800Mbit/\text{s}$

belegte Ressourcen: 6 BRAM-Blöcke, ca. 5% der logischen Ressourcen

10 Zusatzpunkte

Die Abgabe der Programmieraufgaben soll bitte folgendermaßen erfolgen:

1. **Verschieben Sie oder benennen Sie eine evtl. generierte Konfigurationsdatei zunächst um, damit sie beim Cleanup nicht gelöscht wird!**
2. Dann führen Sie in ISE "`Project` → `Cleanup Project Files`" aus.
3. Verpacken Sie den Projektordner in ein gepacktes Archiv (`.zip`, `.tar.gz` o.ä.).
4. Senden Sie das Archiv per Mail an `lwi@informatik.uni-kiel.de`