Universität Bremen – Fachbereich 3
Mathematik & Informatik

Prof. Rolf Drechsler & Jannis Stoppe M.Sc., drechsle/jannis@informatik.uni-bremen.de, MZH 4330/1362
Dipl.-Math.techn. Oliver Keszöcze, keszocze@informatik.uni-bremen.de, MZH 4300

4. Übungsblatt zur Vorlesung

Rechnerarchitektur und Eingebettete Systeme

SystemC
SystemC ist, wie in der Vorlesung bereits erklärt, eine C++ Klassenbibliothekt zum Entwurft von Hardware-/Softwaresystemen. SystemC kann direkt von der eigenen Website unter `http://accellera.org/downloads/standards/systemc` in der aktuellen Version 2.3.1 oder alternativ auch von github unter `https://github.com/systemc` in der fast aktuellen Version 2.3 bezogen werden.

Eure Projekte müssen folgende Kriterien erfüllen, damit sie überall (also auch von uns) kompilier-, benutz- und bewertbar sind:

- Sie sind aus der Kommandozeile per make oder cmake kompilierbar, wahlweise mit clang oder gcc. Studenten die auf Windows-Rechnern arbeiten können dafür auf cygwin, den Rechnerpool o.ä. zugreifen. Die Bewertung (also inbs. das Kompilieren) wird auf Linux-Rechnern ausgeführt.
- Sie sind in sich geschlossen, bis auf den Zugriff auf die (fertig vorkompilierte) SystemC-Bibliothek die unter `../systemc` liegt, mit den includes und libraries in den entsprechenden Unterordnern.
- Sie sind (selbstverständlich) im Code dokumentiert sowie in der Abgabe als ganzes erklärt.
- Sie enthalten ausführliche Tests.

Abgabe ist am 17. 12. vor dem Tutorium um 08:30 Uhr.

Alle Module dieses Übungsblattes, die eine Clock verwenden, nehmen *ausschließlich* diese in ihre Sensitivitätsliste auf.

Aufgabe 1 (2 Punkte)

SystemC ist eine (standardisierte) Bibliothek für C++ und folgt damit einem grundlegend anderen Ansatz als dedizierte Hardware-Beschreibungssprachen. Wo liegen Vor- und Nachteile von SystemC gegenüber HDLs der Register-Transfer-Ebene? Erläutere.

Aufgabe 2 (8 Punkte)

Eine Arithemtisch-Logische Einheit (ALU) führt in einer CPU Berechnungen durch. Implementiert eine Klasse `alu`, die von `sc_module` erbt und folgende Eigenschaften aufweist:

- Sie besitzt einen Clock-Eingang.
- Sie besitzt einen Eingang `instruction` für die auszuführende Instruktion vom Typ `sc_uint<8>`.
- Sie besitzt zwei Eingänge für die Daten `dataA` und `dataB` vom Typ `sc_int<64>`
- Sie besitzt einen Ausgang `result` für das Resultat vom Typ `sc_int<64>`

Die ALU soll folgende Operationen ausführen können (mit den entsprechenden Werten für den Instruktionseingan in Klammern): Arithmetische Operationen: Addition (0x01), Subtraktion (0x02), Multiplikation (0x03), Division (0x04) sowie Modulo (0x05). Bitweise Operationen: Bitshifting (wobei der

zweite Operand die Distanz und die Richtung angibt) (0x10), bitweises OR (0x11), bitweises AND (0x12) sowie bitweises XOR (0x13). Außerdem soll ein Vergleich (0x14) realisiert werden. Dieser verhält sich wie folgt: das Ergebnis ist 0, wenn die Operanden gleich sind bzw. -1 oder 1 wenn der zweite Operand kleiner bzw. größer ist.

Das Ergebnis der ALU soll im nächsten Takt an `result` anliegen.

Aufgabe 3 (10 Punkte)

Im Tutorium wurde ein `switch` vorgestellt, der einen Ein- und mehrere Ausgänge besaß und Pakete zwischen diesen vermittelt hat.

In dieser Aufgabe erweitert ihr dieses Konzept um mehrere Eigenschaften. Zum einen ist die Anzahl an Ein- und Ausgängen nicht mehr statisch festgelegt, sondern wird variabel bei der Erzeugung des switches angegeben. Zusätzlich arbeiten die Absender schneller als die Empfänger (d.h. auf unterschiedlichen clocks).

Baut dafür ein SystemC-Programm, das (mindestens) folgende Klassen besitzt:

- `packet` ist die Datenstruktur, die eure Pakete beschreibt. Ein Paket enthält dabei eine Zieladresse die angibt, auf welchen Ziel-Port es geschickt werden soll, ein prioritäts-Flag das angibt ob das Paket priorisiert (also vor anderen, nicht priorisierten Paketen) durchgeleitet werden soll sowie das Datum selbst – letzteres wird als `unsigned long long` kodiert.
- `switch` erbt von sc_module und ist dafür zuständig, Pakete von einem Eingang an einen anderen Ausgang zu vermitteln. Dafür besitzt die `switch`-Klasse n Eingänge und m Ausgänge sowie einen Eingang für eine Clock.
- `producer` und `consumer` stellen die Klassen dar, die an den Eingängen bzw. Ausgängen des switches Angeschlossen werden und generieren die Pakete bzw. nehmen diese ein Empfang. Dafür enthalten diese jeweils einen Aus- oder Eingang sowie einen Eingang für eine Clock.

Das Programm wird (aus der Kommandozeile) auf folgende Weise gestartet:

```
switch numIn numOut genSpeed inClock outClock bufferSize
```

Dabei beschreiben die Parameter folgende Eigenschaften:

- `numIn` und `numOut` legen die Anzahl der Ein- und Ausgänge fest und liegt jeweils zwischen 0 und 127.
- `genSpeed` legt (in Prozent von 0 bis 100) die Wahrscheinlichkeit fest, mit der ein `producer` ein Paket (pro Clock-Zyklus) generiert.
- `inClock` und `outClock` legen die clock-Geschwindigkeiten für `producer`, `consumer` und den `switch` (der so schnell läuft die die producer) in NS von 1 bis 100 fest. Es muss `inClock` $\geq$ `outClock` gelten.
- `bufferSize` steuert, wie groß der fifo-Puffer pro Ausgang ist, um Pakete zwischenzuspeichern. Da die `consumer` langsamer getaktet sein können als der `switch` können die Pakete nicht in jedem Takt weitergegeben werden; entsprechend müssen sie teilweise gepuffert werden. Beachtet für die Pufferung die Prioritätsinformation der Pakete, die manche Pakete in der Queue vorne einsortiert werden lässt.

Die Signale sind per Templates auf eure `packet`-Typen zu spezifizieren (d.h. `sc_in<packet>` etc.). Solltet ihr zusätzliche Signale oder Ports (z.B. zur Kommunikation unter den Modulen) benötigen dürft ihr diese beliebig hinzufügen.

Ermittelt über verschiedene Testläufe, ab wann euer Switch anfängt Pakete zu verlieren. Welche Parameter spielen dafür eine Rolle, welche nicht? Warum?