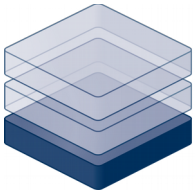

Chair for Embedded Systems

Prof. Dr.-Ing. Diana Göhringer
Professur für Adaptive Dynamische Systeme
Fakultät Informatik, TU Dresden
Nöthnitzer Straße 46
01187 Dresden

Praktikum ”Informatik 2 für ET/MT/RES“

Im Rahmen des Praktikums sollen Sie einen Logik–Simulator in Java implementieren. Als Grundelemente der Simulation sollen Sie dabei einfache Gatter (Und, Oder, Exor, Nand, Nor) und Speicherelemente (D-Latch, D-FlipFlop) vorsehen. Außerdem sollen die speziellen Elemente Input und Output vorgesehen werden, die als Ein- und Ausgänge der Schaltung fungieren. Diese Grundelemente sollen durch Signale mit einander verbunden werden. Signal ist somit auch eine Klasse, die Sie implementieren müssen.

Die Aufgabe soll in vier Stufen bearbeitet werden. Mit der Bearbeitung der ersten Stufe können Sie direkt zum Beginn des Praktikums anfangen. Die weiteren Stufen bauen darauf auf, so dass die Ergebnisse der vorhergehenden Stufen direkt in der nächsten Stufe weiter verwendet werden können. Gehen Sie hierbei ruhig das Risiko ein, am Anfang etwas zu machen, was Sie später noch mal überarbeiten müssen. Dieser Vorgang gehört zum Programmieren dazu! Ebenso sollten Sie mit der Bearbeitung der ersten drei Teilaufgaben so früh wie möglich anfangen. Sie werden Erfahrungen sammeln, die Ihnen die Bearbeitung der vierten Teilaufgabe sehr vereinfachen!

Im Einzelnen sind die folgenden Teilaufgaben zu bearbeiten:

1. Implementieren Sie zunächst nur die beiden Klassen `Nand` und `Signal`. Sie dürfen davon ausgehen, dass eine Verbindungsleitung nur an eine Quelle (und beliebig viele Ziele) angeschlossen werden kann. Das Nand-Gatter soll in diesem Fall idealisiert arbeiten, also keinerlei Zeitverzögerung vom Eingang zum Ausgang besitzen. Zur weiteren Vereinfachung dürfen Sie ebenfalls davon ausgehen, dass keine Schaltungen mit kombinatorischen Rückkopplungen simuliert werden sollen (Überlegen Sie sich, warum Rückkopplungen nicht ohne weiteres in diesem idealisierten Fall funktionieren würden). Die Beschreibung der zu simulierenden Schaltung geschieht im Programmtext durch geeignetes Verbinden der benötigten Instanzen der Grundelemente. Zu diesem Zweck erhalten Sie auf der Web-Seite zum Praktikum den Quelltext der Klasse `FunctionalSimulator`. Diesem Quelltext können Sie auch die zu implementierende Schnittstelle der Klassen `Nand` und `Signal` entnehmen. Die Beschreibung der Sequenz von Eingangssignalen geschieht ebenfalls im Programmtext. Jede Änderung eines Eingangssignals wird durch die Schaltung bis zu den Ausgängen propagiert. Jede Ausgangsänderung soll ausgegeben werden.

2. Die Simulation soll nun um eine Zeitverzögerung erweitert werden. D.h. die Änderung eines Ausgangs erfolgt nicht direkt mit der Änderung des Eingangs, sondern erst nach einer spezifizierbaren Verzögerung. Für alle Grundelement muss daher eine Verzögerung zw. den Ein- und Ausgängen angegeben werden können. Vereinfachend nehmen wir dabei an, dass diese Verzögerung zwischen allen Ein- und Ausgängen gleich ist. Implementieren Sie dazu eine ereignisgesteuerte Simulation mit einer Ereignis–Warteschlange (wird in der Vorlesung rechtzeitig erläutert).
   Schaltung und Eingangssequenz sollen weiterhin durch den Programmtext vor gegeben werden. Zu diesem Zweck erhalten Sie auf der Web–Seite zum Praktikum die Klasse `TimingSimulator` (inkl. Quelltext).
   Erweitern Sie nun Ihre Klasse `Nand` und `Signal` entsprechend. Außerdem verwendet die Klasse `TimingSimulator` noch die Klassen `EventQueue` und `Event`. Auch deren Schnittstelle ergibt sich aus der Verwendung der Klassen im Simulator. Beachten Sie, dass in der Klasse `TimingSimulator` vor Beginn der eigentlichen Simulation eine besondere Behandlung der Schaltung erfolgt, um den ein-geschwungenen Zustand der Schaltung zu ermitteln.

3. Erweitern Sie Ihren Simulator nun um alle geforderten Gattertypen und machen Sie dazu Gebrauch von der Vererbung. Dazu müssen Sie Ihre Klasse `Nand` modifizieren und von einer geeigneten Superklasse ableiten. Ob Sie die Klassen `LATCH` und `FF` von der gleichen Superklasse ableiten können, sollten Sie sich zunächst gründlich überlegen. Eventuell lohnt es sich, noch mehr Vererbungshierarchien einzuführen.
   Sie können die korrekte Funktionen Ihrer Klassen anhand der vorgegebenen Klasse `FullTimingSimulator` testen, die alle zu implementierenden Elemente verwendet. Sie finden diese Klasse inkl. Quelltext ebenfalls auf der Web-Seite zum Praktikum.

4. Im letzten Teil der Aufgabenstellung sollen die Schaltung und die Eingangssequenz nun aus einer Datei eingelesen werden. Dazu bekommen Sie von den Web-Seite zum Praktikum die Hilfsklasse `Dateileser`. Das verbindliche Format der Beschreibungs-dateien wird auf den Web–Seiten zum Praktikum bekannt gegeben. Dort finden Sie auch einige Beispieldateien zum Ausprobieren Ihres Simulators.

   In diesem Fall müssen Sie also selbst eine neue Klasse `DateiSimulator` schreiben, die die zu simulierende Schaltung aus einer Datei einliest und dann aus einer weiteren Datei die Sequenz von Eingangsdaten einliest und simuliert.

Liste der zu implementierenden Gatter

`AND2`, `AND3`, `AND4`
`NAND2`, `NAND3`, `NAND4`
`OR2`, `OR3`, `OR4`
`NOR2`, `NOR3`, `NOR4`
`EXOR2`, `EXOR3`, `EXOR4`
`NOT`, `BUF`
Bei diesen Elementen werden die Eingänge jeweils mit `i1...in` benannt. Der Ausgang soll immer `o` heißen. `NOT` stellt eine einfache Negation dar, `BUF` ein Element mit einem Eingang und einem Ausgang (wird im Wesentlichen nur als Verzögerungselement gebraucht).

`LATCH`
Dieses pegelgesteuerte Latch hat die beiden Eingänge `e` (Enable) und `d` (Data), sowie die Ausgänge `q` und `nq` (entsprechend dem normalen und dem negierten Ausgang)

`FF`
Dieses positiv flankengesteuerte FlipFlop hat die beiden Eingänge `c` (Clock) und `d` (Data), sowie die Ausgänge `q` und `nq` (entsprechend dem normalen und dem negierten Ausgang)

Die Aufgabe wurde von Prof. Dr.–Ing. Christian Hochberger erstellt.