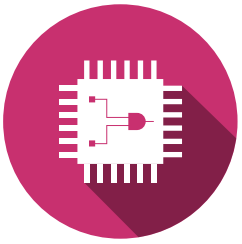


Hochgeschwindigkeitsaddierer (Pipelined)

Inhaltsverzeichnis

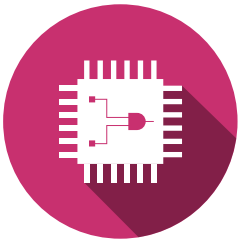
1 Einführung

In diesem Labor soll ein einfacher Addierer in eine ßchnelle"Version umgewandelt werden, die auf dem Prinzip der Pipeline"basiert.

2 Start

Die Schaltung befindet sich in der Bibliothek **PiepelinedOperators**. Der Prüfstand in der Bibliothek **PiepelinedOperators_test**.

*Erinnerung: Das Modellierungsprogramm muss über die Datei **pipelinedOperators.bat** gestartet werden.*



3 Ziele

3.1 Kombinatorischer Addierer

Ein kombinatorischer n-Bit-Addierer besteht aus n 1-Bit-Addierern, die jeweils aus Gattern mit bis zu 3 Verzögerungsstufen bestehen (der längste Weg führt nacheinander durch die Gatter XOR, AND und dann OR, um Cout zu bestimmen). Die Gültigkeit der Berechnung kann daher erst nach einer Mindestverzögerung von $3 * t_{delay}$ für den ersten Addierer und dann $2 * t_{delay}$ für jeden weiteren Addierer garantiert werden (da das erste XOR bereits einen gültigen Wert hat, sobald a und b zu Beginn der Berechnung geladen wurden, muss Cout nur noch neu berechnet werden).

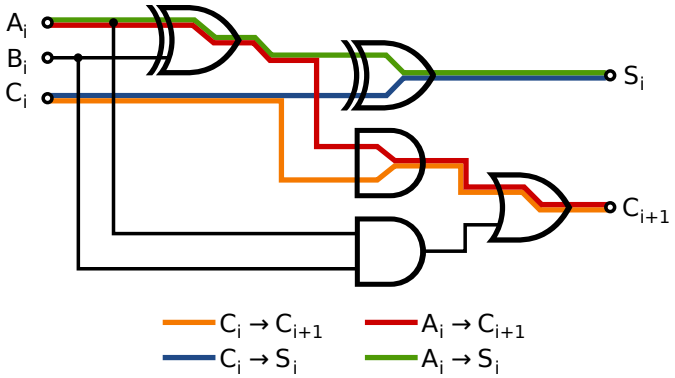


Abbildung 1: Vollständiger kombinatorischer Addierer

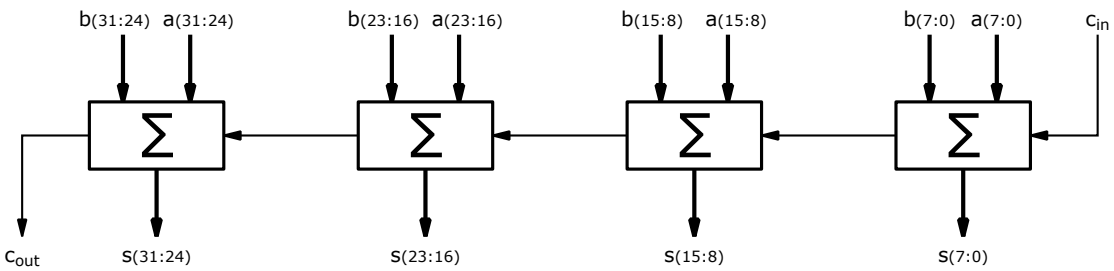
Letztendlich kommt eine einfache 16-Bit-Addition auf 33 t_{delay}. Je nach Technologie kann eine Verzögerung von 10 ns gelten, was eine minimale Wartezeit von 330 ns und damit einen Taktgeber mit einer maximalen Frequenz von 3 MHz impliziert. Es wird also notwendig, diesen Addierer in kleinere Teile zu zerlegen, um das System als Ganzes beschleunigen zu können.

3.1.1 Todo

Öffnen Sie den Teststand für den kombinatorischen Addierer **parallelAdder_tb**, schreiben Sie die VHDL-Architektur des Addierers **parallelAdder** und überprüfen Sie, ob alles richtig funktioniert.

3.2 Iterativer Addierer

Der schnelle Addierer wird einen Satz kombinatorischer Addierer verwenden. Dieser Addierer wird zunächst erstellt, um zu verstehen, wie man eine Schaltung in VHDL beschreibt, indem man Komponenten mithilfe von Schleifen instanziiert.



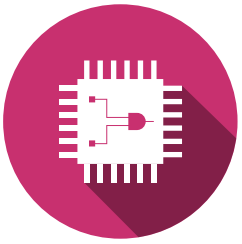


Abbildung 2: Iterativer Addierer

*Die zur Verfügung gestellte **noPipe**-Architektur gibt ein Beispiel für die VHDL-Beschreibung eines iterativen Systems durch das Platzieren von Komponenten mithilfe einer for ... generate-Schleife.*

3.2.1 Todo

- Setzen Sie in der Bibliothek **pipelinedOperators** die **noPipe**-Architektur als Standardarchitektur für den Block **pipelineAdder**.
- Öffnen Sie die Testbank für den kombinatorischen Addierer **pipelineAdder_tb** und überprüfen Sie, ob der Addierer ordnungsgemäß funktioniert.

3.3 Schneller Addierer

Der Schnelladdierer erfolgt durch Einfügen von Flipflops zwischen die kombinatorischen Addierer:

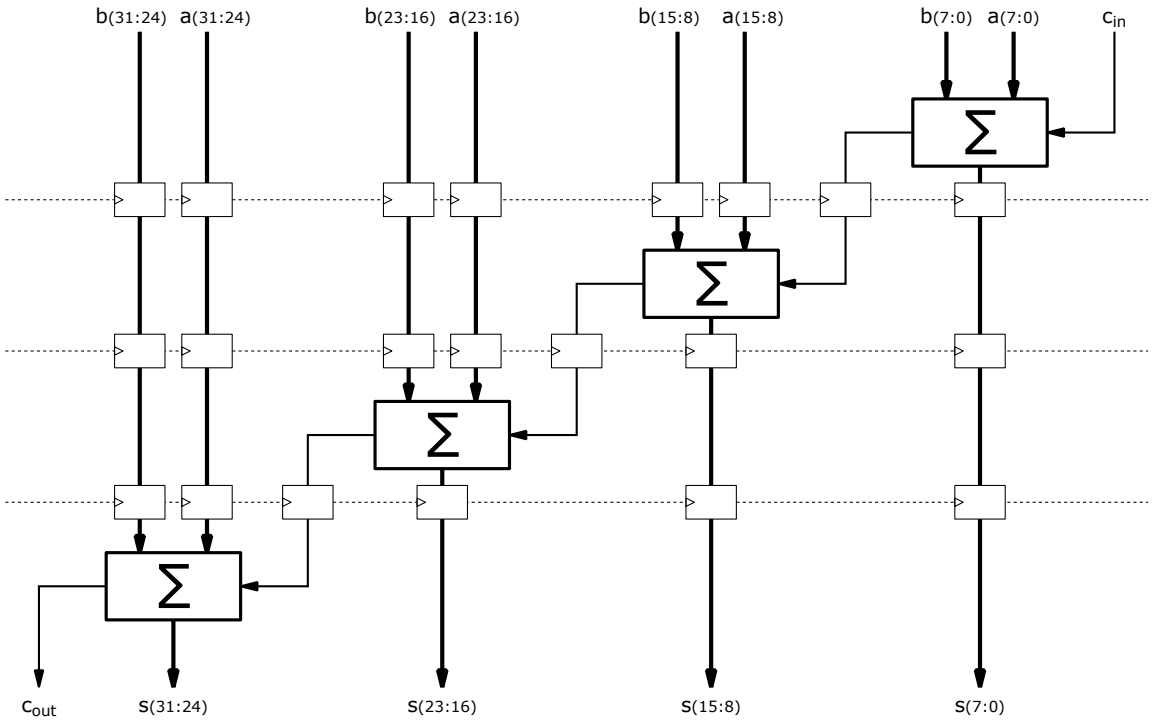
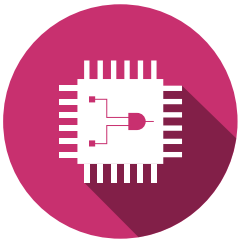


Abbildung 3: Schneller Addierer

Aus der Sicht der Clock"dauert eine einfache Berechnung nun natürlich n Taktschläge, um ein Ergebnis zu liefern. Aber zur Erinnerung: Es ist möglich, einen viel schnelleren Taktgeber zu haben, und wenn die Pipeline mit mehreren aufeinanderfolgenden Berechnungen gefüttert wird, ist es möglich, bei jedem Taktschlag ein Ergebnis zu erhalten.

3.3.1 Todo

- Setzen Sie in der Bibliothek **pipelinedOperators** die Architektur **studentVersion** als Standardarchitektur für den Block **pipelineAdder**.
- Schreibe die VHDL-Architektur des Addierers **pipelineAdder** und überprüfe, ob der Addierer



richtig funktioniert.

3.4 Schneller Zähler

Der schnelle Addierer ist also in der Lage, in jeder Taktperiode eine neue Summe zu liefern. Er benötigt jedoch eine bestimmte Anzahl von Taktperioden, bevor er das Ergebnis der Summe liefert.

3.4.1 Todo

Schreibe die VHDL-Architektur des Zählers **pipelineCounter** und überprüfe seine Funktionstüchtigkeit.

3.5 Comparaison

Der Zähler mit Pipeline sollte prinzipiell mit einer höheren Geschwindigkeit arbeiten können als ein Zähler mit kombinatorischem, übertragsausbreitendem Addierer.

3.5.1 Todo

Führen Sie die Synthese des Zählers mit den beiden Varianten des Addierers durch und vergleichen Sie die Betriebsgeschwindigkeiten. *Bei Auswahl des kombinatorischen Addierers (**noPipe**-Architektur) wäre der Betrieb nicht korrekt, aber hier geht es nur darum, die Betriebsgeschwindigkeiten zu vergleichen.*