

Verkabelter Multiplizierer

Labor Digitales Design

Inhalt

1 Ziel	1
2 Multiplizierer für natürliche Zahlen	2
2.1 Algorithmus	2
2.2 Analyse	2
2.3 Schaltung	3
2.4 Erstellung	3
2.5 Simulation	4
3 Multiplizierer für Arithmetische Zahlen	5
3.1 Algorithmus	5
3.2 Analyse	5
3.3 Erstellung	5
3.4 Simulation	6
4 Analyse	7
5 Checkout	8

1 | Ziel

In diesem Labor beschäftigen Sie sich mit der Implementierung von Multiplizierern für natürliche und arithmetische Zahlen. Sie lernen, wie Multiplikationsalgorithmen in Hardware realisiert werden und welche Besonderheiten bei der Verarbeitung von binären Zahlen berücksichtigt werden müssen.

Durch die schrittweise Entwicklung eines Multiplizierers für natürliche Zahlen und dessen Erweiterung auf arithmetische Zahlen im Zweierkomplement gewinnen Sie ein vertieftes Verständnis für digitale Schaltungen. Dabei setzen Sie verschiedene logische Bausteine ein und analysieren deren Verhalten unter Berücksichtigung von Berechnungsverzögerungen.



2 | Multiplizierer für natürliche Zahlen

In diesem Abschnitt entwickeln Sie einen Multiplizierer für natürliche Zahlen, also für nicht-negative Ganzzahlen, die in Binärform gespeichert sind.

2.1 Algorithmus

Abbildung 1 stellt den Algorithmus zur Multiplikation zweier binärer Zahlen dar. Dieser basiert auf der Berechnung von Teilprodukten, die anschliessend summiert werden. Diese Methode entspricht dem schriftlichen Multiplizieren im Dezimalsystem, nur dass hier binäre Stellen verwendet werden. Jedes Teilprodukt entsteht durch die Multiplikation einer Zahl mit einer einzelnen Binärstelle der anderen Zahl. Die resultierenden Teilprodukte werden dann bitweise nach links verschoben und aufaddiert, um das Endergebnis zu erhalten.

				a ₃	a ₂	a ₁	a ₀
				× b ₃	b ₂	b ₁	b ₀
				b ₀ *a ₃	b ₀ *a ₂	b ₀ *a ₁	b ₀ *a ₀
			b ₁ *a ₃	b ₁ *a ₂	b ₁ *a ₁	b ₁ *a ₀	
		b ₂ *a ₃	b ₂ *a ₂	b ₂ *a ₁	b ₂ *a ₀		
	b ₃ *a ₃	b ₃ *a ₂	b ₃ *a ₁	b ₃ *a ₀			
p ₇	p ₆	p ₅	p ₄	p ₃	p ₂	p ₁	p ₀

Abbildung 1 - Multiplikationsalgorithmus

2.2 Analyse

Zuerst werden wir die Multiplikationen von 2 natürlichen Zahlen (unsigned) mit 4 Bits codiert analysieren. Die Multiplikation von 2 Zahlen mit 4 Bits ergibt ein Produkt mit 8 Bits. Wie es in der Abbildung 1 ersichtlich ist.



Bestimmen Sie die grösstmögliche sowie die kleinstmögliche Zahl welche mit diesem Multiplizierer erreicht werden kann.

Erweiternd bestimmen Sie die Anzahl der Bits, die das Produkt zweier natürlicher Binärzahlen benötigt, die mit n_1 bzw. n_2 Bits codiert sind.



2.3 Schaltung

Abbildung 2 zeigt die Schaltung eines Multiplikierers, welcher nach dem oben angegebenen Algorithmus arbeitet.

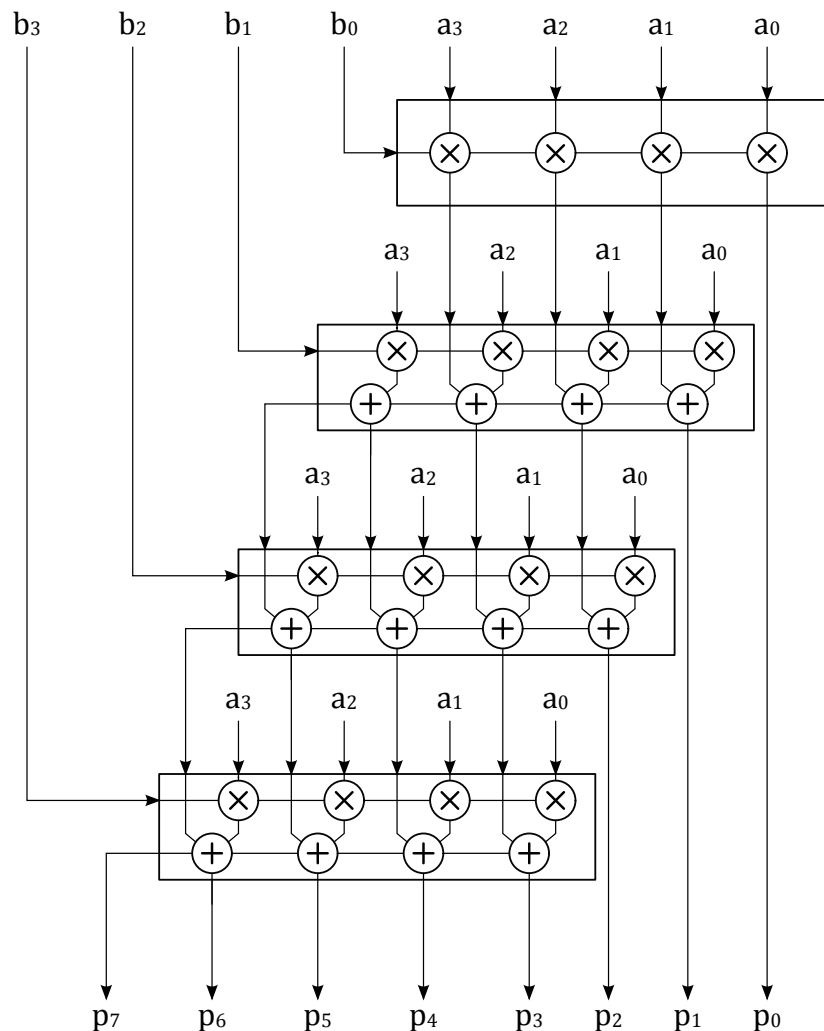


Abbildung 2 - Architektur des Multiplikierers

2.4 Erstellung

Erstellen Sie die Schaltung des Multiplikierers der Abbildung 2. Einige Schaltungselemente sind im Block **MUL/mul4Unsigned** bereits vorgegeben. Ergänzen Sie die fehlenden Elemente, um die Funktionalität des Multiplikierers zu gewährleisten.



Mit Hilfe von INV, UND, ODER und XOR Gattern, ergänzen Sie das hierarchische Schema **MUL/mul4Unsigned** des Multiplikierers der Abbildung 2.



2.5 Simulation

Jede implementierte Schaltung muss korrekt getestet werden. Die Testbench **MUL_test/mul4Unsigned_tb** muss die **vollständige** Funktion der Schaltung überprüfen. Wie wird dies im Falle des Testers **MUL_test/mul4Unsigned_tester** realisiert?



Simulieren Sie die Testbench **MUL_test/mul4Unsigned_tb** mit der Simulationsdatei **\$SIMULATION_DIR/MUL1.do**.



3 | Multiplizierer für Arithmetische Zahlen

In diesem Abschnitt erweitern Sie den zuvor implementierten Multiplizierer auf arithmetische Zahlen, also Zahlen, die im Zweierkomplement dargestellt sind. Dies ermöglicht die Multiplikation sowohl positiver als auch negativer Ganzzahlen.

3.1 Algorithmus

Abbildung 3 stellt den Algorithmus von Baugh-Wooley zur Multiplikation von zwei im Zweier-Komplement codierten arithmetischen Zahlen (signed) mit derselben Anzahl an Bits dar. Im Vergleich zum Abbildung 1 werden hier mehrere zusätzlich Additionen sowie Invertierungen durchgeführt, um das korrekte Ergebnis zu erhalten.

				a ₃	a ₂	a ₁	a ₀
				× b ₃	b ₂	b ₁	b ₀
			1	<u>b₀*a₃</u>	b ₀ *a ₂	b ₀ *a ₁	b ₀ *a ₀
			<u>b₁*a₃</u>	b ₁ *a ₂	b ₁ *a ₁	b ₁ *a ₀	
		<u>b₂*a₃</u>	<u>b₂*a₂</u>	<u>b₂*a₁</u>	b ₂ *a ₀		
1	b ₃ *a ₃	<u>b₃*a₂</u>	<u>b₃*a₁</u>	<u>b₃*a₀</u>			
p ₇	p ₆	p ₅	p ₄	p ₃	p ₂	p ₁	p ₀

Abbildung 3 - Multiplikationsalgorithmus für Zahlen im Zweier-Komplement

3.2 Analyse

Zuerst analysieren wir die Multiplikation zweier arithmetischer Zahlen (signed), die im Zweierkomplement mit 4 Bits codiert sind. Im Gegensatz zur Multiplikation natürlicher Zahlen (unsigned) müssen wir hier zusätzlich negative Zahlen berücksichtigen.



Bestimmen Sie die grösstmögliche sowie die kleinstmögliche Zahl welche mit diesem Multiplizieren erreicht werden kann.

3.3 Erstellung

Erstellen Sie die Schaltung des Multiplizierers der Abbildung 3. Einige Schaltungselemente sind im Block **MUL/mul4Signed** bereits vorgegeben. Ergänzen Sie die fehlenden Elemente, um die Funktionalität des Multiplizierers zu gewährleisten.



Mit Hilfe von INV, UND, ODER und XOR Gattern, ergänzen Sie das hierarchische Schema **MUL/mul4Signed** des Multiplizierers der Abbildung 3.



3.4 Simulation

Jede implementierte Schaltung muss korrekt getestet werden. Die Testbench **MUL_test/mul4Signed_tb** muss die **vollständige** Funktion der Schaltung überprüfen.



Simulieren Sie die Testbench **MUL_test/mul4Signed_tb** mit der Simulationsdatei **\$SIMULATION_DIR/MUL2.do**.

Wie erklären Sie sich die vielen Glitches die in der Simulation auftreten?



4 | Analyse

Unter der Annahme, dass alle Logikgatter dieselbe Verzögerung von 1ns haben, analysieren Sie die maximale Berechnungsverzögerung des Multiplikators für natürliche Zahlen (unsigned), dargestellt in Abbildung 2.

Die Verzögerung ergibt sich aus der Anzahl der hintereinander geschalteten Gatter, die das Signal durchlaufen muss, bevor das endgültige Ergebnis erreicht wird.



Bestimmen Sie die maximale Berechnungsverzögerung des Multiplikators Abbildung 2.



Schlagen Sie eine optimierte Architektur vor, um die Berechnungsverzögerung zu reduzieren.



5 | Checkout

Bevor Sie das Labor verlassen, stellen Sie sicher, dass Sie die folgenden Aufgaben erledigt haben:

- ☐ Schaltungsentwurf
 - ☐ Der Multiplizierer für natürliche Zahlen **MUL/mul4Unsigned** wurde entworfen und getestet.
 - ☐ Der Multiplizierer für arithmetische Zahlen **MUL/mul4Signed** wurde entworfen und getestet.
- ☐ Simulationen
 - ☐ Die Schaltungen wurden mit den jeweiligen Testbenches **MUL_test/mul4Unsigned_tb** und **MUL_test/mul4Signed_tb** erfolgreich getestet.
- ☐ Analyse
 - ☐ Die grösstmögliche und kleinstmögliche Zahl, die mit dem Multiplizierern erreicht werden kann, wurde bestimmt.
 - ☐ Die Theoretische Anzahl der Bits der Multiplikationen wurde bestimmt.
 - ☐ Die Berechnungsverzögerungen für den Multiplizierer **MUL/mul4Unsigned** wurden bestimmt.
 - ☐ (*optional*) Eine optimierte Version des Multiplizierers wurde erstellt, welche die Berechnungsverzögerung reduziert.
- ☐ Dokumentation und Projektdateien
 - ☐ Stellen Sie sicher, dass alle Schritte (Entwurf, Schaltung, Simulationen) in Ihrem Laborbericht gut dokumentiert sind.
 - ☐ Speichern Sie das Projekt auf einem USB-Stick oder dem gemeinsamen Netzlaufwerk (**\\filer01.hevs.ch**).
 - ☐ Teilen Sie Dateien mit Ihrem Laborpartner, um die Arbeitskontinuität sicherzustellen.