Übung zur Vorlesung „Technische Informatik I“, SS03

Lösungsblatt 4 – „Zahlendarstellung und Rechenarithmetik“

Aufgabe 1 Zahlendarstellung

a) Bestimmen Sie Darstellungen der Zahl 113_{10} im Hexadezimalsystem (Basis b=16), im Binärsystem (Basis b=2), im Oktalsystem (Basis b=8) sowie im 12er-System. Wie kann die Umwandlung zwischen den Darstellungen der Basen 2, 8, 16 vereinfacht werden?

Lösung: $113_{10} = 71_{16}$ $113_{10} = 161_8$ $113_{10} = 1110001_2$ $113_{10} = 95_{12}$

$(0111|0001)_2 \Rightarrow (7|1)_{16}$ – Zusammenfassung von 4er Blöcken der Binärzahl

$(001|110|001)_2 \Rightarrow (1|6|1)_8$ – Zusammenfassung von 3er Blöcken der Binärzahl

b) Bestimmen Sie die 6-stellige Binärdarstellung der Zahl -17_{10} im Einer-Komplement und im Zweier-Komplement. Wie sieht jeweils die 8-stellige Darstellung dieser Zahl aus?

Lösung:

$17_{10}=010001_2$ (6-Stellig, positive)

101110_2 (1-er Komplement)

101111_2 (2-er Komplement)

11101111_2 (2-er Komplement mit 8 Stellen)

c) Bestimmen Sie die Binärdarstellung der Zahlen 0.375_{10} und 0.1_{10}. Wie viele Nachkommastellen werden zur exakten Darstellung jeweils benötigt?

Hinweis: Die Umwandlung einer dezimalen Bruchzahl in eine Bruchzahl zu Basis „b“ erfolgt durch „Sukzessive Multiplikation“ der Zahl mit „b“. Die ganze Zahl vor dem Komma im Produkt ist die entsprechende Nachkommabitstelle der Binärzahl. Die Multiplikation wird wiederholt bis das Ergebnis „1“ ist oder bis eine gewünschte Anzahl von Nachkommastellen erreicht wird.

Beispiel : $0.25_{10}=0.01_2$

$0.25 \times 2=\underline{0}.5\ (0)$

$0.5 \times 2=\underline{1}.0\ (1)$

Lösung:

$0.375_{10}=0.011_2$, $0.1_{10} \approx 0.0(0011)_2$ – Der Ausdruck in den Klammern wiederholt sich periodisch.

d) Wie lautet die Zweier-Komplement 8-Bit Binärdarstellung der Zahl -4.375_{10} bei k=4 Nachkommastellen?

Lösung:

2-er Komplement: $-4.375_{10} = 1011.1010_2$

!Vorsicht!: Bei Zweier-Komplement wird „1“ auf die niederste Bitstelle der Zahl addiert (z.B. 1011.100<u>1</u>=>1011.1010), NICHT auf die niederste Vorkomma-Bitstelle (Falsch: 101<u>1</u>.1001=>1100.1001).

Aufgabe 2 Addition von Binärzahlen

Gegeben seien die Dezimalzahlen x = 100, y = –10 und z = –120.

a) Wandeln Sie diese Zahlen unter Verwendung des Zweier-Komplements zunächst in 8-Bit Binärzahlen um. Berechnen Sie dann x + y , x + z und y + z. Kontrollieren Sie jeweils Ihr Ergebnis.

Lösung:

Dezimal	Zweier-Komplement (8-Bit)
100	01100100
–10	11110110
–120	10001000

	x + y	x + z	y + z
Resultat	01011010	11101100	01111110 – Falsch Echter $-130_{10} = 101111110_2$

Aufgabe 3 Multiplikation von Binärzahlen

a) Gegeben seien die im Zweier-Komplement kodierten 6-Bit Zahlen $x=010110_2$ und $y=110010_2$. Multiplizieren Sie x und y mit dem Algorithmus für positive Zahlen (vgl. Vorlesungsfolie E-28). Welche Korrektur ist erforderlich, um das korrekte Produkt zu erhalten?

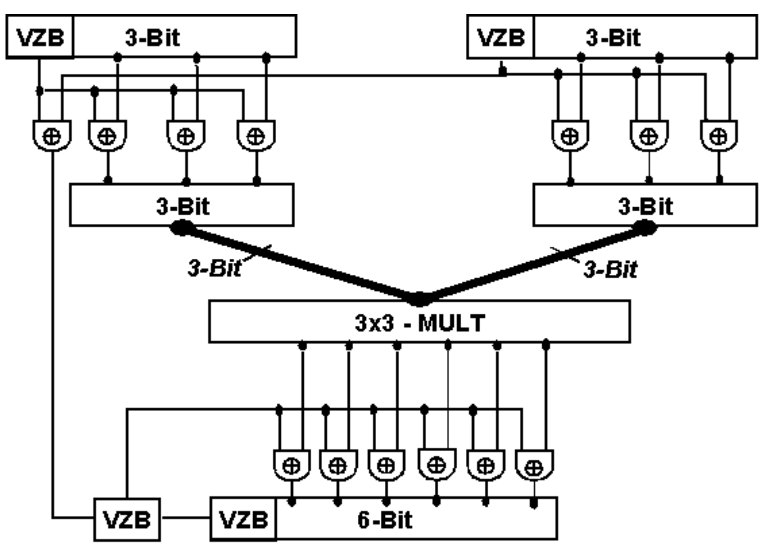
Lösung:

$x=010110_2=22_{10}$ und $y=110010_2=-14_{10}$

$x \,.\, y=111011001100_2=-308_{10}$

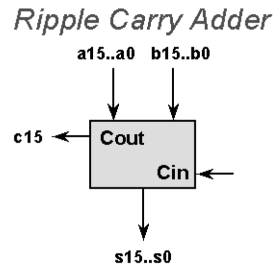
b) Gegeben sei ein 3 x 3-Bit Multiplizierer für vorzeichenlose Zahlen. Entwerfen Sie eine Schaltung, die unter Zuhilfenahme dieses Multiplizierers sowie zusätzlichen Gattern vorzeichenbehaftete 4-Bit Zahlen multiplizieren kann. Sowohl Eingabe als auch Ausgabe sollen im Einer-Komplement dargestellt sein. Welche Wortbreite sollten Sie für die Ausgabe mindestens vorsehen, um Fehler durch Überläufe zu vermeiden?

Lösung:



Aufgabe 4 Addierwerke

a) Ein 16-Bit Addierwerk ist intern als „*Ripple Carry Adder*“ (RCA) realisiert, d. h. durch eine Reihenschaltung von 16 Volladdierern. Die Verzögerung in einem Volladdierer ist $\tau_{FA} = 2\tau$ (wobei τ die Gatterlaufzeit darstellt). Welche maximale Zeit τ_{RCA} wird für die Addition zweier 16-Bit Zahlen benötigt ?

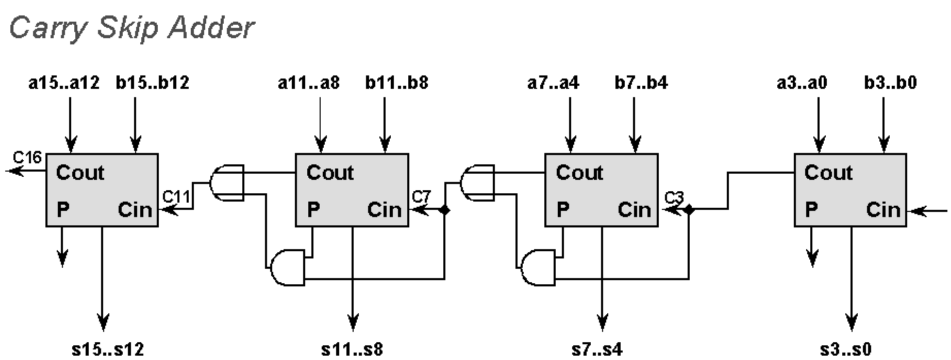


Lösung:

$\tau_{RCA} = 32\tau$

b) Nun sei angenommen, daß das 16-Bit Addierwerk intern als sogenannter „*Carry Skip Adder*“ realisiert wird, der gemäß folgendem Bild aus vier 4-Bit Addierern aufgebaut ist.

Jeder 4-Bit Addierer ist intern als RCA gemäß a) implementiert, verfügt jedoch zusätzlich über eine Logik, die in einer Zeit von 2τ ein „*Block Propagate*“–Signal P aus allen lokalen Eingangssignalen a_i und b_i berechnet. P gibt an, ob ein 4-Bit Addierblock ein Carry-Signal von Cin nach Cout weiterleitet (P=1) oder nicht (P=0).



Bestimmen Sie die maximale Verzögerungszeit $\tau_{CarrySkip}$ bei der Addition zweier 16-Bit Zahlen.

Lösung:

$\tau_{CarrySkip} = 16\tau$ + Zusatzzeit der „Und“ und „Oder“ Gatter

Aufgabe 5 IEEE Zahlendarstellung

Eine IEEE 754 32-Bit Gleitkommazahl hat folgendes Format: $x = (-1)^s \cdot (1.f) \cdot 2^{e-127}$ wobei die einzelnen Bits ($b_{31}..b_0$) in einem Rechner in folgender Reihenfolge abgelegt werden: ($s, e_7, ..e_0, f_{-1}, .. f_{-23}$). Details zu diesem Format finden Sie auf den Vorlesungsfolien.

a) Welche Zahlen werden durch die Bitmuster 0|10000101|00000010000000000000000 und 1|10000100|10010000000000000000000 dargestellt? (Die senkrechten Striche dienen hierbei lediglich der Übersicht.)

Lösung:

0|10000101|00000010000000000000000 = 64.5

1|10000100|10010000000000000000000 = –50

b) Bestimmen Sie die IEEE Gleitkommadarstellung der Zahlen 1, –5 und $6.25 \cdot 10^{-3}$.

Lösung:

1 = 0| 01111111| 000000000000000000000000000

–5 = 1| 10000001| 010000000000000000000000000

$6.25 \cdot 10^{-3}$ =0| 01111000| 1001100110011001100110<u>1</u>

$6.25 \cdot 10^{-3} \cdot 2 = 0 + 0.0125$
$0.0125 \cdot 2 = 0 + 0.025$
$0.025 \cdot 2 = 0 + 0.05$
$0.05 \cdot 2 = 0 + 0.1$
$0.1 \cdot 2 = 0 + 0.2$
$0.2 \cdot 2 = 0 + 0.4$
$0.4 \cdot 2 = 0 + 0.8$
$0.8 \cdot 2 = 1 + 0.6$
$0.6 \cdot 2 = 1 + 0.2$

Diese Zahl ist periodisch.

$6.25 \cdot 10^{-3}{}_{10} = 0.00000(0011)_2 = 1.(1001) \cdot 2^{-8}$

$6.25 \cdot 10^{-3} = (-1)^0 \cdot 1.10011001100110011001101 \cdot 2^{119-127}$

$6.25 \cdot 10^{-3} = 0|\ 1110111|\ 1001100110011001100110\underline{1}$

Die letzte Stelle der Zahl $6.25 \cdot 10^{-3}$ ist aufgerundet.

c) Berechnen Sie die Summe der Zahlen aus Aufgabe a). Beachten Sie dabei, daß die Eingaben sowie das Ergebnis in normalisierter Form vorliegen.

Lösung:

0|10000101|00000010000000000000000 + 1|10000100|10010000000000000000000 =

0|10000010|11010000000000000000000