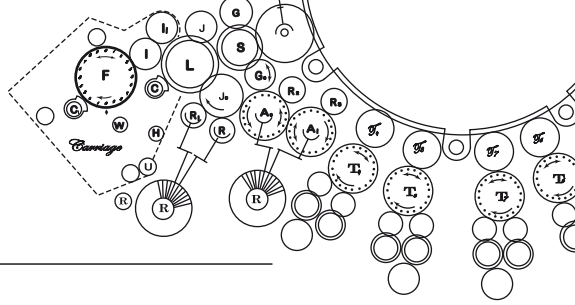


Aufgaben zur Booth-Multiplikations-Methode

Aufgabe 1 Sequentielle Multiplikation mit Booth-Recoding

KAP 04

Führen Sie folgende Multiplikationen mithilfe der Booth-Recoding-Methode durch!
Stellen Sie hierbei selbständig fest, welche Anzahl von Bits notwendig ist!

1. $(-19)\cdot(-23) =$
2. $22\cdot 502 =$
   (spielt es eine Rolle, welche der beiden Zahlen Sie als A bzw. B wählen?)
3. $29\cdot(-17) =$

Aufgabe 2 Implementierung Booth Recoder

KAP 04

In den Folien haben Sie eine Software-Implementierung der sequentiellen Multiplikation kennengelernt (Kapitel 4, Arithmetik). Eine derartige Implementierung wird in der Praxis oft als ausführbare *Spezifikation* verwendet, und erlaubt, die Funktion der „Schaltung“ zu testen, bevor sie in Verilog implementiert wird.

a) Implementieren Sie den Booth-Recoding-Algorithmus (für 32-Bit-Zahlen) in der Programmiersprache Ihrer Wahl!

Hinweis: In den Programmiersprachen C, C++ und Java können Sie mit Hilfe des bitweisen Und-Operators (&) einzelne Bits einer Binärzahl maskieren. Der Ausdruck (`x & 1`) entspricht dem niederwertigsten Bit `x`, während (`x & 2`)`>>1` das zweite Bit von rechts ergibt. Die 32 niederwertigeren Bits einer 64-Bit-Zahl `x` erhalten Sie mithilfe des Ausdrucks `x & 0x00000000ffffffff`. (Sie dürfen annehmen dass der `>>`-Operator *arithmetisch* shifted.)

In Verilog ist die Manipulation einzelner Bits einfacher, da Sie direkt auf diese zugreifen können (z.B. `x[1]` oder `x[31:0]`).

b) Der Booth-Recoding-Algorithmus für 32-Bit-Zahlen berechnet 32 partielle Produkte. Die Anzahl dieser partiellen Produkte kann halbiert werden, indem man in jedem Schritt *drei* Bits A_i, A_{i+1}, A_{i-1} des Multiplikators A betrachtet. Diese drei Bits ergeben in jedem der i Schritte (wobei $0 \leq 2\cdot i < 32$) einen Faktor $-2 \leq F_i \leq 2$:

$$F_i(A) = A_{2\cdot i-1} + A_{2\cdot i} - 2\cdot A_{2\cdot i+1}$$

Aufgrund des Lemmas (siehe auch Aufgabenstellung *c*)

$$A = \sum_{i=0}^{15} 2^{2\cdot i}\cdot F_i(A)$$

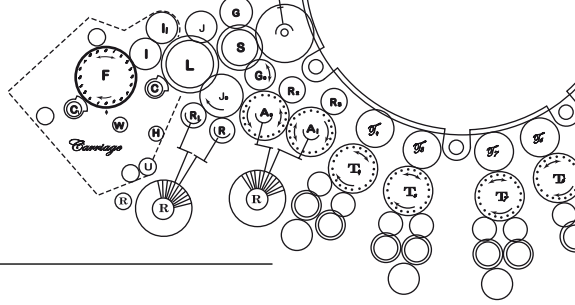
kann man die Multiplikation wie folgt in partielle Produkte aufteilen:

$$B\cdot A = B\cdot\sum_{i=0}^{15} 2^{2\cdot i}\cdot F_i(A) = \sum_{i=0}^{15} 2^{2\cdot i}\cdot B\cdot F_i(A)$$

Die Berechnung des Faktors $F_i(A)$ kann effizient durch einen Multiplexer realisiert werden.

Diese Methode nennt sich „Radix 4“-Booth-Multiplikation und wird zum Beispiel in dem in [OMJ+06] vorgestellten CELL-Prozessor eingesetzt.

Implementieren Sie diese Methode in der Programmiersprache Ihrer Wahl!



c) Beweisen Sie das Lemma

$$A = \sum_{i=0}^{15} 2^{2 \cdot i} \cdot (A_{2 \cdot i-1} + A_{2 \cdot i} - 2 \cdot A_{2 \cdot i+1}) \,,$$

das in Aufgabe *b* verwendet wurde!

Hinweis: Ein ähnliches Lemma wurde in den Folien und im Buch auf Seite 144 bewiesen.

Literatur

[OMJ+06] Hwa-Joon Oh, S.M. Mueller, C. Jacobi, K.D. Tran, S.R. Cottier, B.W. Michael, H. Nishikawa, Y. Totsuka, T. Namatame, N. Yano, T. Machida, and S.H. Dhong. A fully pipelined single-precision floating-point unit in the synergistic processor element of a CELL processor. *Solid-State Circuits, IEEE Journal of*, 41(4):759–771, April 2006.