VU Advanced Digital Design

Homework 3

Bernhard Urban

Matr.Nr.: 0725771

lewurm@gmail.com

21. Mai 2011

Inhaltsverzeichnis

1 Switching behavior

Die gegebene Schaltung zählt die Anzahl der gesetzen Bits im Eingabevektor. Logisch gesehen, kann man die Schaltung in zwei Bereiche teilen: Ein Schleifenkopf der die Bedingung prüft ("Vektor ungleich null?") und einen Schleifenkörper ("aktuelles Bit gesetzt?", ggf. Zähler inkrementieren und shiften).

Die Analyse für den gegebenen Inputvektor "1010" ist in Tabelle 1 ersichtlich. Das Ergebnis für diese Eingabe ist "2", wie aus Schritt 28 hervorgeht. Abgesehen davon, erkennt man sehr schön das Muster "(DATA) ⇒ DATA ⇒ (EMPTY) ⇒ EMPTY ⇒ (DATA) ⇒..." bei den einzelnen Elementen wieder.

Step	A	R_1	R_2	R_3	R_4	R_5	Output
1	(1010)	(0)	(E)	E	E	E	E
2	1010	0	(E)	(1010,0)	E	E	E
3	(E)	(E)	E	(1010,0)	E	E	E
4	(E)	(E)	(1)	1010,0	E	(1010,0)	E
5	E	E	(1)	(E)	(1010,0)	1010,0	E
6	E	(1)	1	(E)	(1010,0)	1010,0	E
7	E	(1)	(E)	E	(1010,0)	(E)	E
8	E	1	(E)	(101,0)	101,0	(E)	E
9	E	(E)	E	(101,0)	(E)	E	E
10	E	(E)	(1)	101,0	(E)	(101,0)	E
11	E	E	(1)	(E)	E	(101,0)	E
12	E	(1)	1	(E)	(10,1)	101,0	E
13	E	(1)	(E)	E	(10,1)	(E)	E
14	E	1	(E)	(10,1)	10,1	(E)	E
15	E	(E)	E	(10,1)	(E)	E	E
16	E	(E)	(1)	10,1	(E)	(10,1)	E
17	E	E	(1)	(E)	E	(10,1)	E
18	E	(1)	1	(E)	(1,1)	10,1	E
19	E	(1)	(E)	E	(1,1)	(E)	E
20	E	1	(E)	(1,1)	1,1	(E)	E
21	E	(E)	E	(1,1)	(E)	E	E
22	E	(E)	(1)	1,1	(E)	(1,1)	E
23	E	E	(1)	(E)	E	(1,1)	E
24	E	(1)	1	(E)	(0,2)	1,1	E
25	E	(1)	(E)	E	(0,2)	(E)	E
26	E	1	(E)	(0,2)	0,2	(E)	E
27	E	(E)	E	(0,2)	(E)	E	E
28	E	(E)	(0)	0,2	(E)	E	(2)
29	E	E	(0)	(E)	E	E	2
30	E	(0)	0	(E)	E	E	2
31	E	(0)	(E)	E	E	E	(E)

Tabelle 1: Schritt für Schritt Ausführung der gegebenen Schaltung

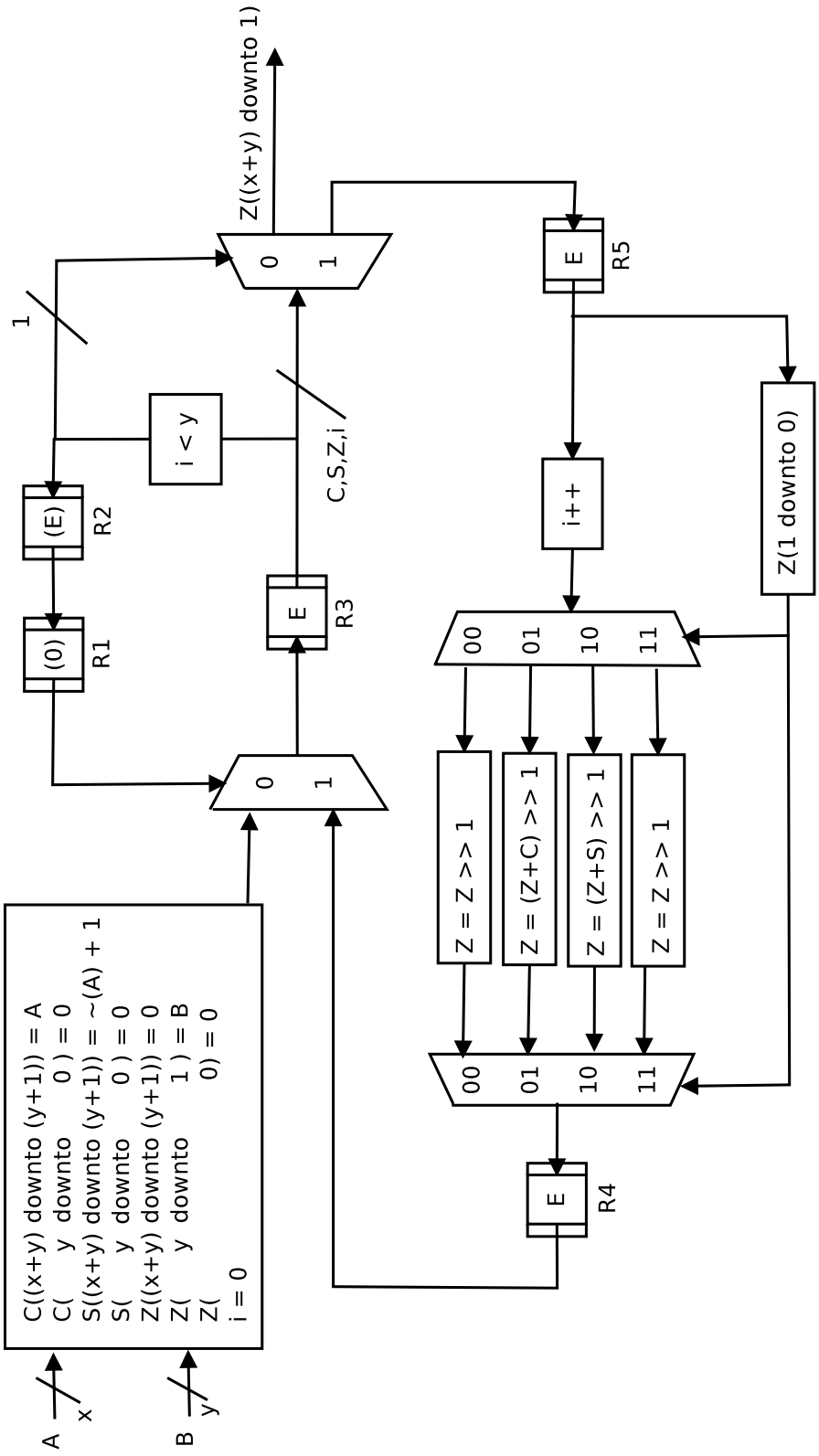


Abbildung 1: Multiplikation mittels Booth Algorithm

2 Circuit design

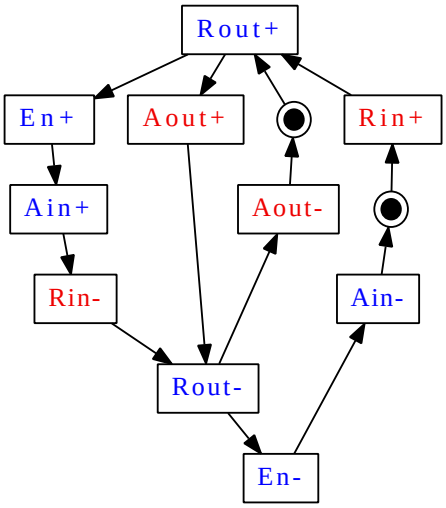
Prinzipell wurde die Vorlage der Schleife aus der ersten Aufgabenstellung entnommen. Entsprechend des Booth Algorithm wurde die Schleifenbedingung und der Schleifenkörper angepasst. Die Lösung ist in Abbildung 1 ersichtlich.

3 Synthesis

Zunächst war der Handshake Controller in der Petrify Input Language zu spezifizieren:

```
.outputs En Rout Ain
.inputs Rin Aout
.graph

Rin+ Rout+
Rout+ En+
En+ Ain+
Ain+ Rin-
Rin- Rout-

Rout+ Aout+
Aout+ Rout-

Rout- En-
En- Ain-
Ain- Rin+

Rout- Aout-
Aout- Rout+

.marking {<Aout-,Rout+> <Ain-,Rin+>}
.end
```

Mittels `petrify` kann man sich nun einerseits seine Spezifikation grafisch darstellen lassen (vgl. Abbildung 2) und andererseits eine Netlist generieren lassen (vgl. Abbildung 3).



INPUTS: Rin,Aout
OUTPUTS: En,Rout,Ain

Abbildung 2: Handshake Controller als Petrinetz dargestellt

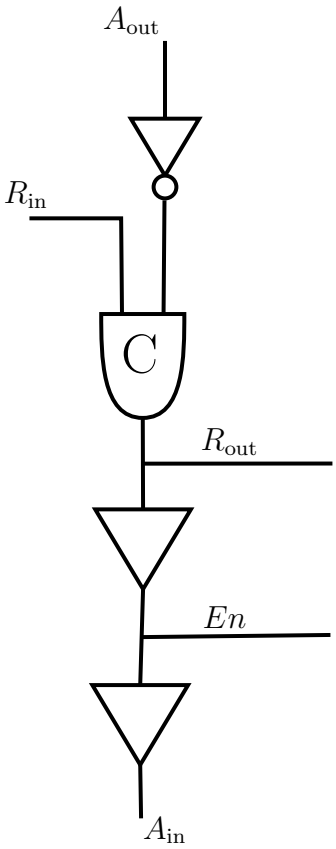


Abbildung 3: Schematic des generierten Handshake Controllers