TECHNISCHE UNIVERSITÄT DRESDEN

Minimierung von Logik-Funktionen unter Ausnutzung der Shannon-Zerlegung

Markus Petke

Dresden, 31.03.2016

Gliederung

- Motivation
- Grundlagen
- Implementierung
- Fazit und Ausblick

Motivation

- Logik-Funktionen als Gatterschaltung realisiert
- Minimierung aus verschiedenen Gründen wichtig
    - Platz
    - Kosten
    - Übersichtlichkeit

- Klassische Algorithmen basieren auf Zusammenfassung von Min-Termen → keine Erkennung von XOR möglich
- Anwendungsgebiet FPGAs
- Grundlage LUTs mit max. 6 Eingängen
- Konfiguration durch Wahrheitstabelle

Grundlagen Shannon-Zerlegung

- Auch Shannon-Expansion genannt
- Erstmals von George Boole aufgestellt
- Entwicklung von DNF aus beliebiger Funktion
- Zerlegung in Teilfunktionen

$$\begin{aligned} y &= F(x_0, x_1, \ldots, x_n) \\ &= x_i * F(x_0, x_1, \ldots, x_n)_{x_i} + \overline{x_i} * F(x_0, x_1, \ldots, x_n)_{\overline{x_i}} \end{aligned}$$

Beispiel Shannon-Expansion

$$y = f(x_0, x_1, x_2) = x_2 x_0 + \overline{x_1} x_0 + x_1 \overline{x_0}$$

Entwickeln nach x_2

$$
\begin{aligned}
y &= f(x_0, x_1, x_2) = x_2 x_0 + \overline{x_1} x_0 + x_1 \overline{x_0} \\
&= x_2 \left(x_0 + \overline{x_1} x_0 + x_1 \overline{x_0} \right) + \overline{x_2} \left(\overline{x_1} x_0 + x_1 \overline{x_0} \right)
\end{aligned}
$$

Entwickeln nach x_1

$$
\begin{aligned}
y &= f(x_0, x_1, x_2) = x_2 x_0 + \overline{x_1} x_0 + x_1 \overline{x_0} \\
&= x_2(x_0 + \overline{x_1} x_0 + x_1 \overline{x_0}) + \overline{x_2}(\overline{x_1} x_0 + x_1 \overline{x_0}) \\
&= x_2(x_1(x_0 + \overline{x_0}) + \overline{x_1}(x_0)) + \overline{x_2}(x_1(\overline{x_0}) + \overline{x_1}(x_0))
\end{aligned}
$$

Entwickeln nach x_0

$$\begin{aligned} y &= f(x_0, x_1, x_2) = x_2 x_0 + \overline{x_1} x_0 + x_1 \overline{x_0} \\ &= x_2(x_0 + \overline{x_1} x_0 + x_1 \overline{x_0}) + \overline{x_2}(\overline{x_1} x_0 + x_1 \overline{x_0}) \\ &= x_2(x_1(x_0 + \overline{x_0}) + \overline{x_1}(x_0)) + \overline{x_2}(x_1(\overline{x_0}) + \overline{x_1}(x_0)) \end{aligned}$$

Distributivgesetz

$$
\begin{aligned}
y &= f(x_0, x_1, x_2) = x_2x_0 + \overline{x_1}x_0 + x_1\overline{x_0} \\
&= x_2(x_0 + \overline{x_1}x_0 + x_1\overline{x_0}) + \overline{x_2}(\overline{x_1}x_0 + x_1\overline{x_0}) \\
&= x_2(x_1(x_0 + \overline{x_0}) + \overline{x_1}(x_0)) + \overline{x_2}(x_1(\overline{x_0}) + \overline{x_1}(x_0)) \\
&= x_2x_1x_0 + x_2x_1\overline{x_0} + x_2\overline{x_1}x_0 + \overline{x_2}x_1\overline{x_0} + \overline{x_2x_1}x_0
\end{aligned}
$$

Nutzung zur Minimierung

- Wahrheitstabelle als Grundlage
- Zerlegung in Teilfunktionen und Vergleich
- Gegebenenfalls Vereinfachung

Vergleiche und Ergebnisse

Vergleich	Ergebnis
$f_0 == f_1$	$f = f_0$
$f_0 == 0$	$f = x_i * f_1$
$f_1 == 0$	$f = \overline{x_i} * f_0$
$f_0 == -1$	$f = \overline{x_i} + f_1$
$f_1 == -1$	$f = x_i + f_0$
$f_0 ==\sim f_1$	$f = x_i \oplus f_0$

Beispiel Minimierung

- 4 Eingangsvariablen
- XOR dargestellt (siehe Tabelle nächste Folie)
- Espresso erkennt es nicht

Term	x3	x2	x1	x0	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

$$\overline{x_3}x_2\overline{x_1 x_0} + \overline{x_3}x_2\overline{x_1}x_0 +$$
$$\overline{x_3}x_2x_1\overline{x_0} + \overline{x_3}x_2x_1x_0 +$$
$$x_3\overline{x_2 x_1 x_0} + x_3\overline{x_2 x_1}x_0 +$$
$$x_3\overline{x_2}x_1\overline{x_0} + x_3\overline{x_2}x_1x_0$$
$$= x_3\overline{x_2} + \overline{x_3}x_2$$
$$= x_3 \oplus x_2$$

Term	x3	x2	x1	x0	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

f0 = 00001111

f1 = 11110000

$$f0 = \sim f1 \Rightarrow$$

$$x_3 \oplus f0 =$$
$$x_3 \oplus (x_2\overline{x_1 x_0} + x_2\overline{x_1}x_0 + x_2x_1\overline{x_0} + x_2x_1x_0)$$

Term	x2	x1	x0	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

f0 = 0000

f1 = 1111

f0 = 0 → $x_2(\overline{x_1 x_0} + \overline{x_1} x_0 + x_1 \overline{x_0} + x_1 x_0)$

f1 = -1 → $x_2 + 0 = x_2$

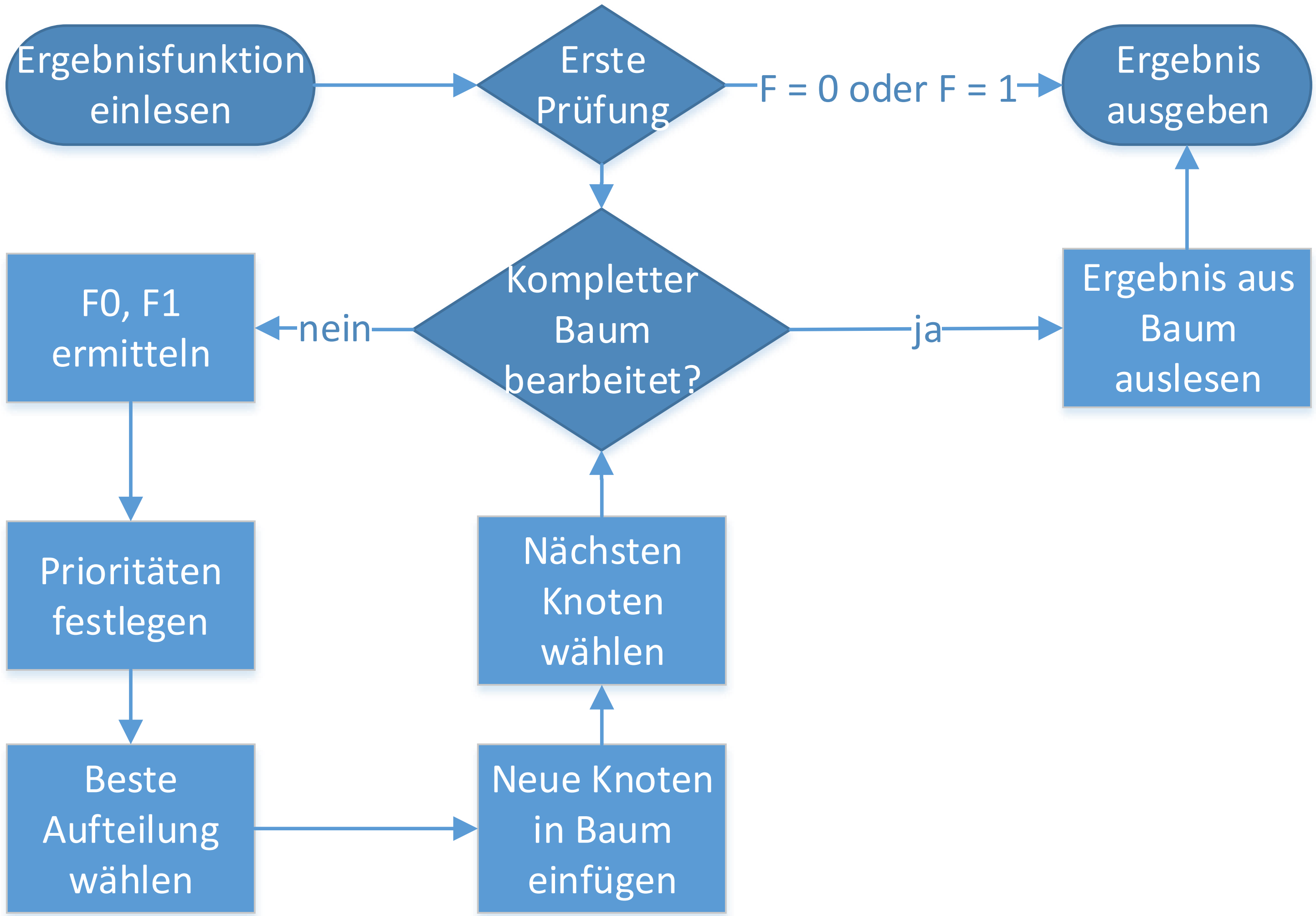
f0 = ~f1 → $x_2 \oplus 0 = x_2$

$\Rightarrow x_3 \oplus x_2$

Implementierung

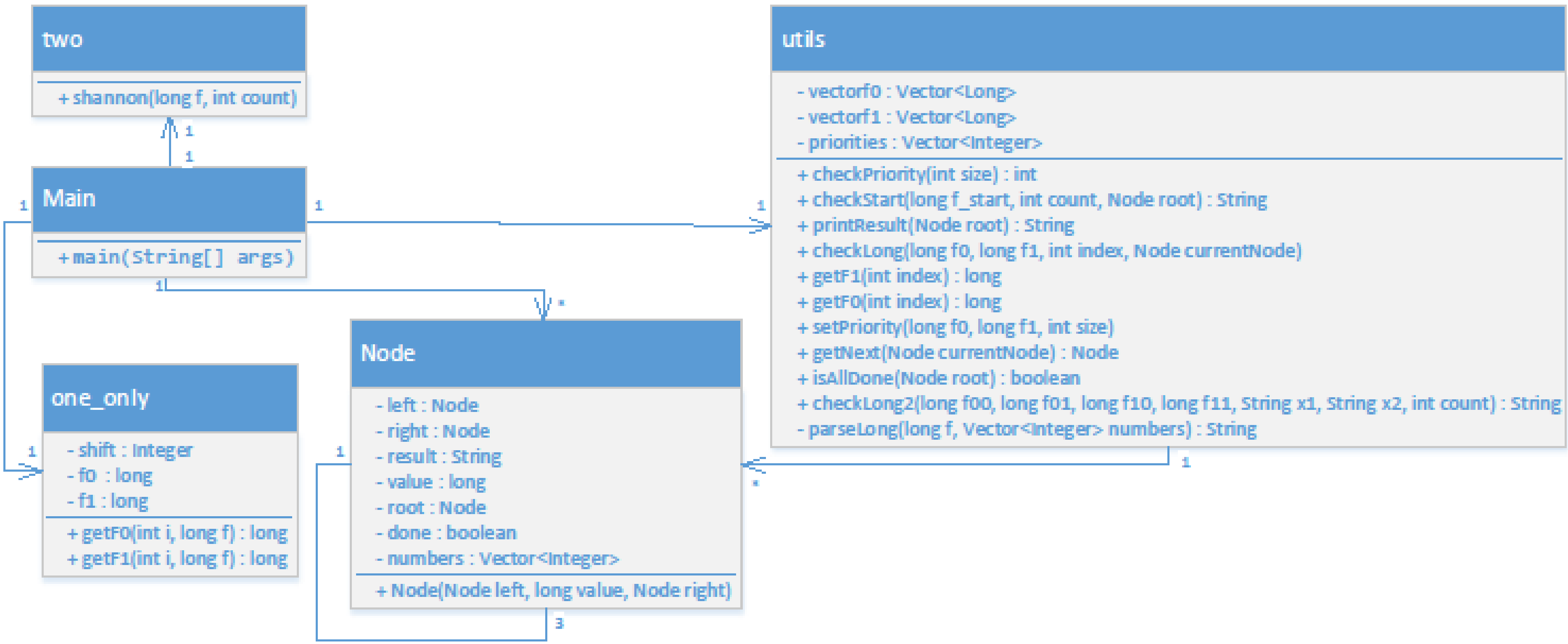
- Long als Eingabe
- Bis zu 64 Stellen (max. 6 Eingänge)
- Parser für DNF

- Zerlegung nach einer Variablen
- Nacheinander alle Variablen
- Teilfunktionen bewertet → beste gewählt
- Wiederholen bis alle Variablen bearbeitet
- Teilfunktionen als Knoten eines Binärbaums gespeichert


Ergebnisfunktion einlesen
Erste Prüfung
F = 0 oder F = 1
Ergebnis ausgeben
Kompletter Baum bearbeitet?
nein
ja
F0, F1 ermitteln
Ergebnis aus Baum auslesen
Prioritäten festlegen
Nächsten Knoten wählen
Beste Aufteilung wählen
Neue Knoten in Baum einfügen

- Abspaltung von Variablenpaaren
- Zerlegung funktioniert
- Einige Heuristiken implementiert
- Als Zusatzinformation

TECHNISCHE UNIVERSITÄT DRESDEN



Fazit

- Abspaltung einer Variablen liefert annehmbare Ergebnisse
- XOR werden zuverlässig erkannt
- Abspaltung von Variablenpaaren nur als Zusatzinfo

Ausblick

- Bewertungen ändern
- Abspaltungen von Variablenpaaren benötigen weitere Heuristiken
- Abspalten von mehr Variablen → Kosten/Nutzen-Faktor beachten

TECHNISCHE UNIVERSITÄT DRESDEN

Vielen Dank für die Aufmerksamkeit