TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Technische Informatik, Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

Effiziente Bibliotheken für FPGA-Resynthese-Algorithmen

Basierend auf:

Kennings, Mishchenko, Vorwerk, Pevzner. Generating Efficient Libraries for use in FPGA Resynthesis Algorithms

Matthias Fetzer

Dresden, 23.01.2013

Gliederung

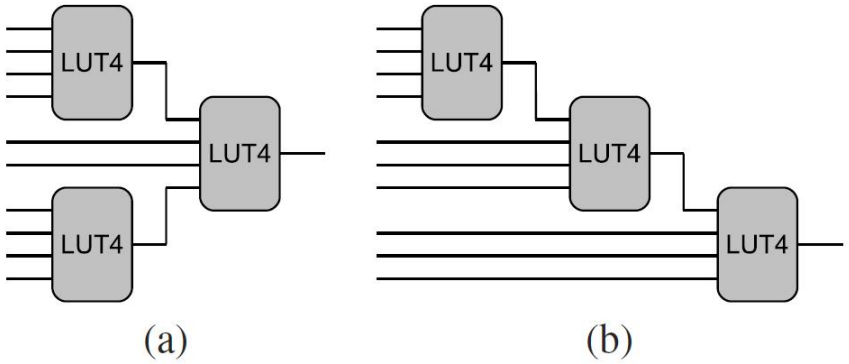
1. Einführung
2. Hintergrund
3. Funktionsabdeckung
4. Bibliotheken
    1. Erzeugung
    2. Einsatz bei der Resynthese
5. Ergebnisse
6. Zusammenfassung
7. Quellen

Einführung

Ziel:

Optimierung von technology-mapped und post-placement Netzwerken bezüglich Fläche, Verzögerung und/oder Verlustleistung

Einführung



-Matching von Logikfunktionen auf K-LUT-Struktur (meist 2<K<7 in aktuellen FPGAs)
- Sinnvoll für Funktionen mit 9-12 Eingängen

-2 Arten:
- LUT-Struktur gegeben → Kann die gegebene Logikfunktion implementiert werden?
- Logikfunktion gegeben → Wie sieht eine LUT-Struktur dafür aus?

Alle Grafiken und Tabellen, sofern nicht anders gekennzeichnet aus [1].

Hintergrund

-Vorhandene Matching-Strategien:
- Strukturbasiert
- SAT-basiert
- Klassenbasiert
- Zerlegungsbasiert

-Hier Kombination:
- Zerlegung von Logikfunktionen um eine klassenbasierte Bibliothek zu erstellen

Funktionsabdeckung

Anzahl der Eingänge	Anzahl möglicher binärer Funktionen
2	16
4	65536
9	**> 10^154**
n	2^(2^n)

[2]

Problem:

Für eine vollständige Bibliothek müssten alle möglichen Logikfuntionen enumeriert und gespeichert werden.

Funktionsabdeckung

NPN-Klassen:
2 Funktionen sind NPN-equivalent, wenn sie sich durch Permutation und/oder Negation der Eingänge und/oder Negation der Ausgänge ineinander überführen lassen. [2]

Lösung:

Nicht alle Logikfunktionen kommen in der Praxis vor → Beschränkung auf wenige NPN-Klassen.

Funktionsabdeckung

Analyse von 100 industriellen FPGA-Designs:

#Inputs	#Functions	#NPN Classes	$\frac{\#\text{NPN Classes}}{\#\text{Functions}}$
5	7768377	3269	4.21e-4
6	18814641	34225	1.82e-3
7	50239975	271646	5.41e-3
8	146678254	2317679	1.58e-2
9	165876500	7145748	4.31e-2

Funktionsabdeckung

Num Input	#NPN classes to cover % of logic functions							
	99%	98%	97%	96%	95%	90%	85%	80%
5	268(8.2%)	158(4.8%)	120(3.7%)	101(3.1%)	87(2.7%)	55(1.7%)	42(1.3%	34(1.0%)
6	3573(10.2%)	1822(5.3%)	1168(3.4%)	831(2.4%)	638(1.9%)	311(0.9%)	207(0.6%	148(0.4%)
7	54266(20.0%)	25364(9.3%)	15093(5.6%)	10139(3.7%)	7322(2.7%)	2437(0.9%)	1313(0.5%	865(0.3%)
8	920963(39.7%)	410955(17.7%)	232892(10.0%)	148041(6.4%)	101368(4.4%)	26956(1.2%)	11399(0.5%	6155(0.3%)
9	5486984(76.8%)	3828219(53.6%)	2276353(31.9%)	1460943(20.4%)	991073(13.9%)	215899(3.0%)	70242(1.0%	29554(0.4%)

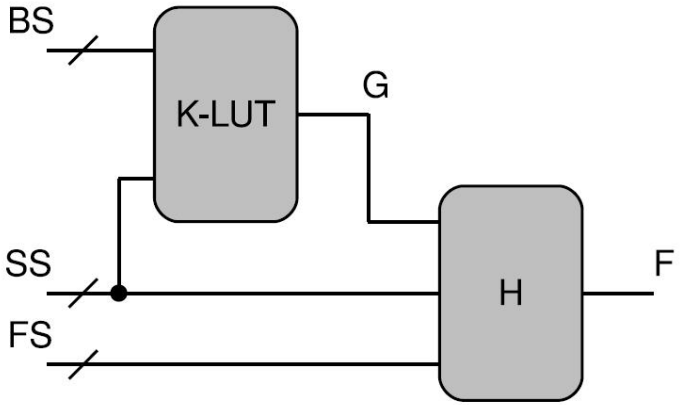
Durch „Mut zur Lücke“ große zusätzliche Ersparnis möglich.

Bibliotheken

Inhalte einer Bibliothek:

1. Abgedeckte Funktionslassen
2. Verschiedene LUT-Strukturen
3. Mapping von Funktionen auf LUTs

Bibliotheken
→Erzeugung



1. Zerlegen der NPN-Klassen durch (rekursive) Roth-Karp-Zerlegung
2. Behalten von dominanten und Verwerfen der restlichen Strukturen
   (A dominant B → Verzögerung jedes Eingangs von A ist <= bei B **und** die Fläche von A <= von B)

Bibliotheken
→Einsatz bei der Resynthese

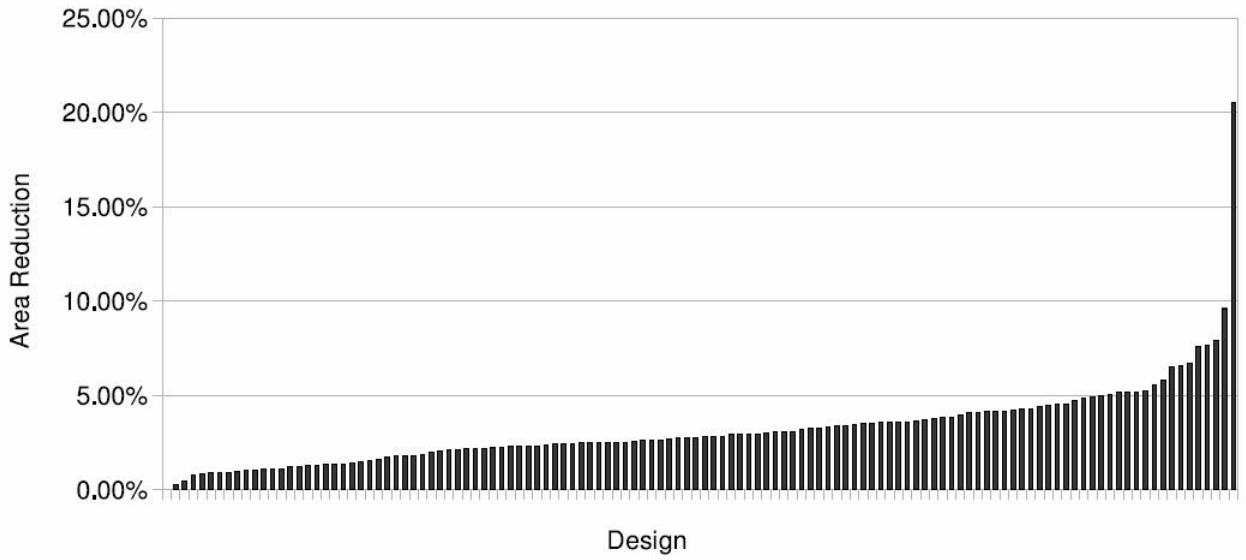
1. Resynthesealgorithmus berechnet einen zu optimierenden Logikteil mit k Eingängen
2. Berechnen der Logikfunktion und NPN-Kodierung
3. Zuordnen zu einer LUT-Struktur der Bibliothek per Hash-Lookup
4. Zurückgeben der gefundenen LUT-Strukturen

Bibliotheken
→Einsatz bei der Resynthese

Sonderfälle:

- keine passende LUT-Struktur in der Bibliothek
  - Direktes Verwenden der Roth-Karp-Zerlegung
- Anzahl der Eingänge k größer als gespeicherte LUT-Strukturen
  - Hybridansatz: Anwenden der Roth-Karp-Zerlegung bis für den verbleibenden Teil passende Strukturen in der Bibliothek vorhanden sind

Ergebnisse



Flächenoptimierende Resynthese (100 Designs):
-Durchschnittliche Flächenreduktion 3,21%
-Maximale Flächenreduktion 20,51%

Ergebnisse

NPN Coverage	CPU Ratio	Hit Ratio
100%	1.00	0.98
99%	1.71	0.96
98%	3.44	0.94
97%	3.94	0.92
96%	4.66	0.90
95%	5.33	0.89
90%	11.5	0.82

Effizienz bei unvollständiger Bibliothek

-Signifikant erhöhte Rechenzeit

- Kritischer Punkt ist die Zerlegung von unbekannten Funktionen → Optimierter, heuristischer Algorithmus kann Operation deutlich beschleunigen

-Jedoch deutlich handlichere Bibliothek

Zusammenfassung

Bei

1. Sorgfältiger Auswahl der Logikfunktionen,
2. Nutzung von Äquivalenzklassen und
3. Eliminierung von redundanten LUT-Strukturen

ist Bibliotheken-basierte Resynthese möglich!

Zusammenfassung

Vorteile:

1. Sehr gut erweiterbar auf größere LUT-Strukturen
2. Beschleunigung bei der Fläche-optimierenden Resynthese im Vergleich zu reinen Zerlegungsalgorithmen
3. Verhältnis zwischen Rechenzeit und Größe der Bibliothek sehr gut einstellbar

Quellen

[1] Kennings, Mishchenko, Vorwerk, Pevzner. Generating Efficient Libraries for use in FPGA Resynthesis Algorithms

[2] Correia, Reis. Classifying n-Input Boolean Functions

Anhang

Begrenzung des |Shared Set| auf 1.

NPN Coverage	%Area Reduction Avg	Max	CPU Ratio	Hit Ratio
100%	3.20	20.51	1.00	0.98
99%	3.17	20.51	1.47	0.96
98%	3.17	20.51	1.76	0.94
97%	3.16	20.51	1.89	0.92
96%	3.16	20.51	1.99	0.90
95%	3.16	20.51	2.02	0.89
90%	3.16	20.51	2.87	0.82

		Percentage coverage of logic functions								
#Function Inputs	#Avg Decomp	100% # Bytes	99% # Bytes	98% # Bytes	97% # Bytes	96% # Bytes	95% # Bytes	90% # Bytes	85% # Bytes	80% # Bytes
5	2.77	289.8K	23.8K	14.0K	10.6K	9.0K	7.7K	4.9K	3.7K	3.0K
6	3.56	3.9M	407.0K	207.6K	133.1K	94.7K	72.7K	35.4K	23.6K	16.9K
7	5.87	51.0M	10.2M	4.8M	2.8M	1.9M	1.4M	457.8K	246.6K	162.5K
8	5.52	409.4M	162.7M	72.6M	41.1M	26.2M	17.9M	4.8M	3.0M	1.1M
9	4.91	1122.7M	862.1M	601.5M	357.7M	229.5M	155.7M	33.9M	11.0M	4.6M