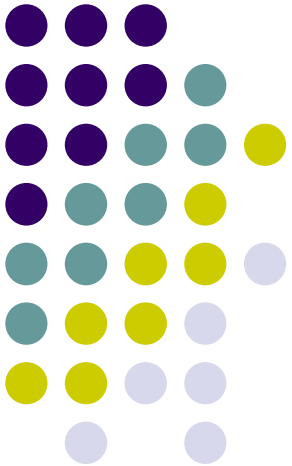
Technische
Universität
Darmstadt

FG Eingebettete Systeme
und ihre Anwendungen

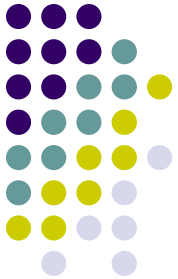
Neue Placer-Module für den "Universal Generator for Logic Circuits on FPGAs"

Jürgen Kunz

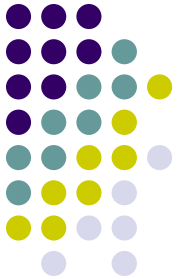


Inhaltsverzeichnis

Inhaltsverzeichnis

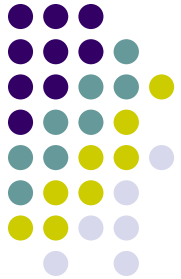


Ziele der Studienarbeit



- Erweiterung der „Universellen Logikgenerators“ um neue Platzierungsmodule
- Verbesserung der Platzierungen zum vorhandenen Platzierungsmodul
- Ausnutzung der Regelmäßigkeit in vielen Schaltungen zum schnelleren Generieren von Platzierungen

Einführung



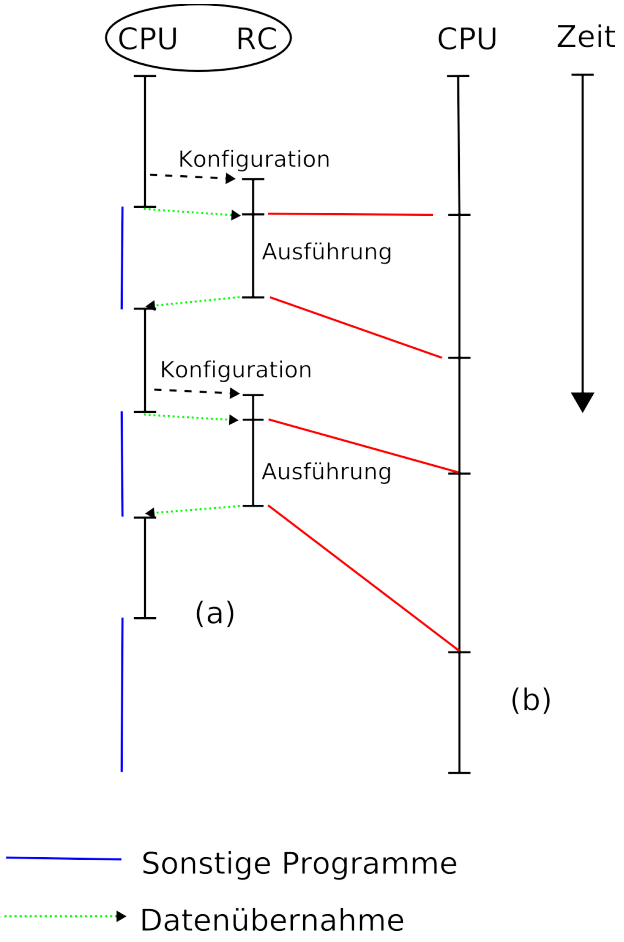
Nachfolgende Folien kurze Erklärung zu:

- Comrade / GLACE
- Universal Logic Generator (ULG)
- FPGA (Field Programmable Gate Array)
- Schaltung (im Sinne von FPGAs)
- Platzierung

Comrade / GLACE

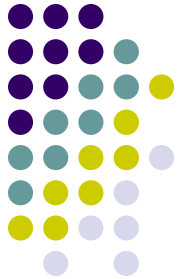
- Compiler, welcher Teile eines Programms bestimmt, um diese beschleunigt auf einer rekonfigurierbare Einheit (RC), z.B. FPGA, auszuführen
- Nicht alle Programmteile lassen sich schneller auf einer RC ausführen
- Comrade untersucht in C geschriebene Programme
- Verwendet GLACE zur Generierung von Hardware-Module für Programmteile, welche auf der RC ausgeführt werden sollen
- GLACE erlaubt nur Gleichungen/Schaltungen mit 4 Input- und 1 Output-Variable

Comrade / GLACE



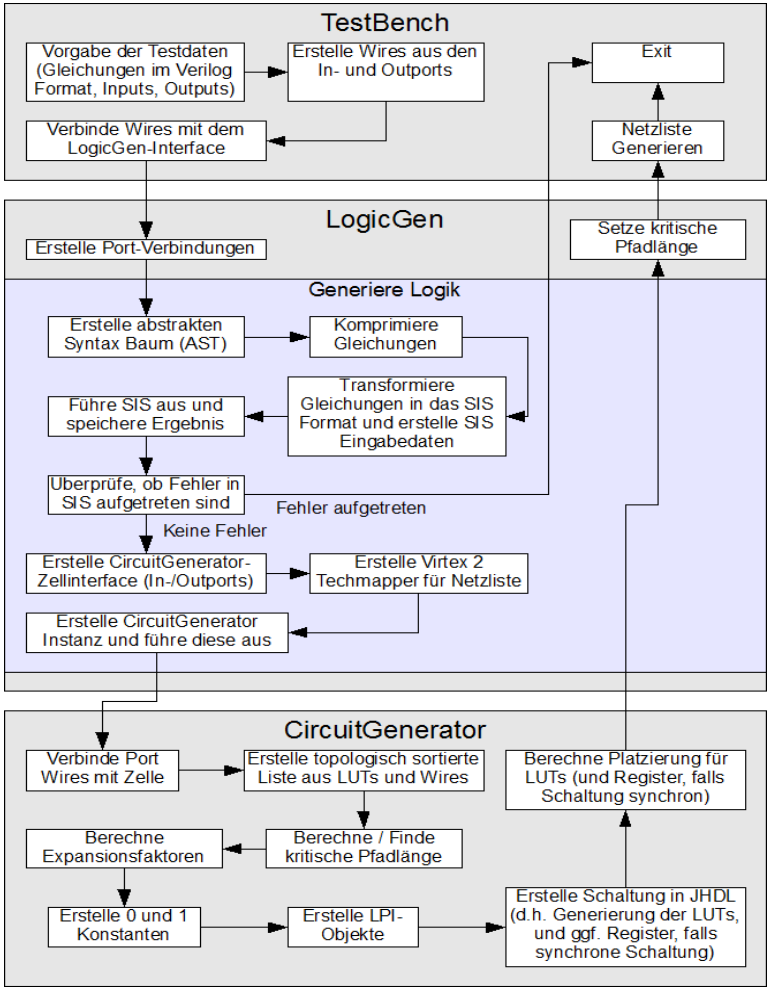
- Grafik zeigt Vergleich der Ausführungszeit auf CPU + RC (a) vs. CPU (b)

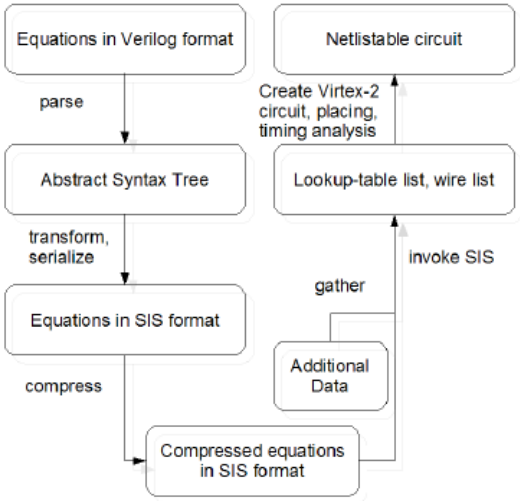
Universeller Logik-Generator (ULG)



- Ersetzt GLACE als Hardware-Modul Generator in Comrade
- Entwickelt von C. Wewetzer im Rahmen seiner Diplomarbeit
- Übersetzt Logik-Gleichungen aus Verilog in ein „platziertes“ Hardware-Modul
- Verwendet SIS (Sequential Interactive Synthesis System) um aus Logikgleichungen LUTs und deren Konnektivität zu bestimmen
- Verwendetes Platzierungsmodul funktionsfähig, erzeugt jedoch keine guten Platzierungen

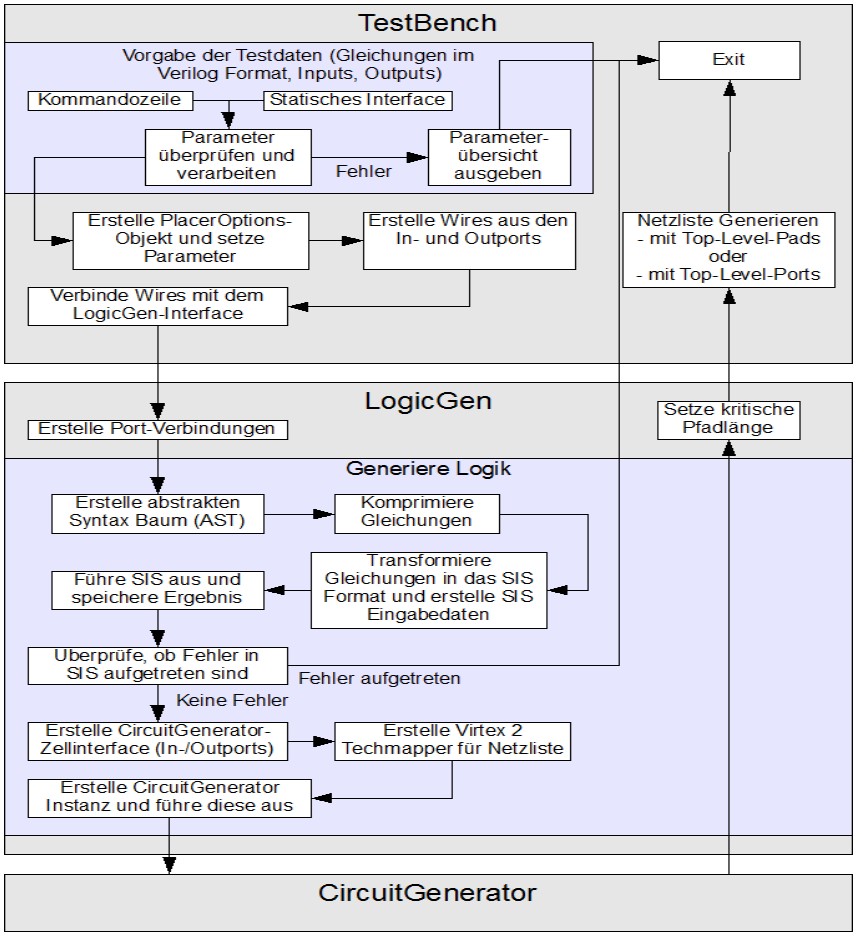
ULG - Übersicht

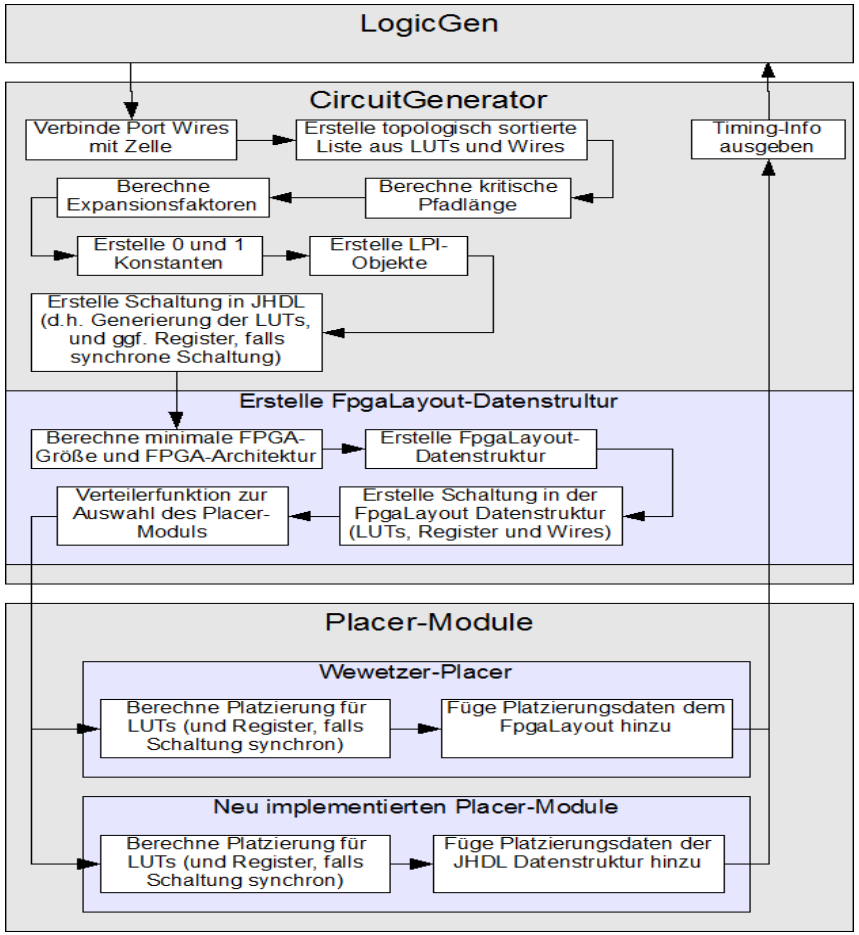




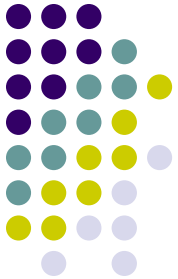
Schematischer Programmablaufplan des ULG

Änderungen am Programmaufbau



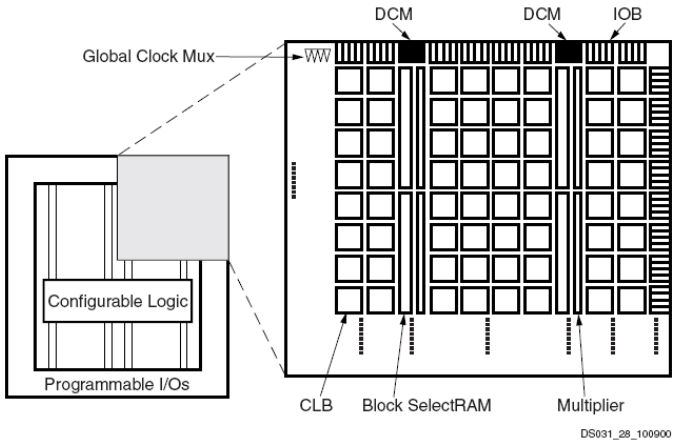


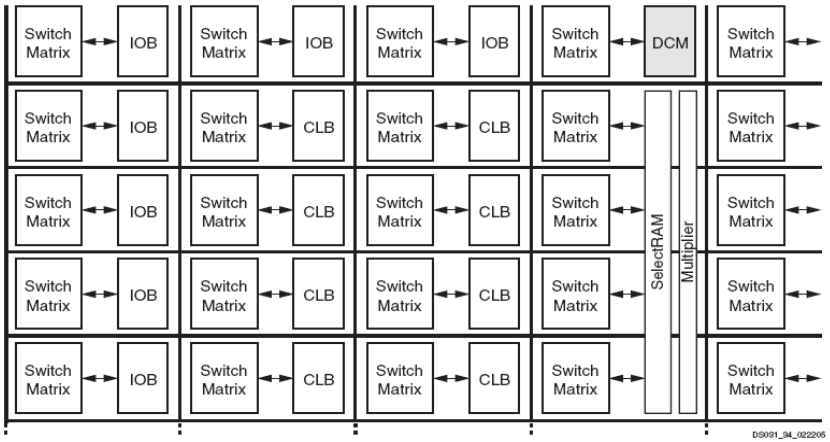
Änderungen am ULG



- Verarbeitung von Parametern über die Kommandozeile
- In Java: Aufruf über ein statisches Interface der Testbench-Klasse bzw. durch instanzieren eines Testbench-Objektes
- Generierung einer eigenen Datenstruktur für das FPGA im CircuitGenerator (von allen neuen Platzierungsalgorithmen benötigt)
- Auswahlfunktion, um gewünschtes Placer-Modul benutzen zu können

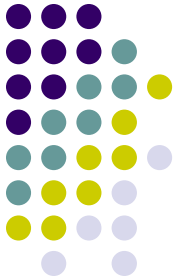
Einführung
FPGA-Architektur



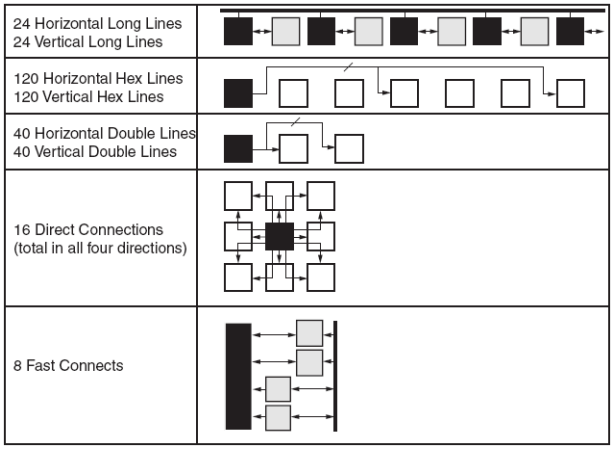


- Sehr regelmäßige Struktur
- Am Rand vorhandene I/O-Einheiten (IOB)
- Innen im Wesentlichen konfigurierbare Logikeinheiten (CLB)
- sowie Switch-Matrizen zur Verschaltung der CLBs und IOBs
- Erzeugung einer Schaltung = Konfigurieren von CLBs, IOBs und Switch-Matrizen

FPGA-Architektur
Xilinx Virtex-2



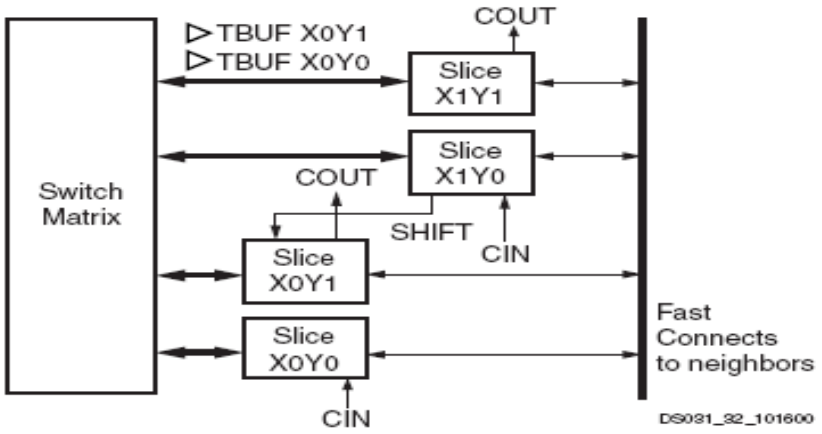
Device	System Gates	CLB Array Row x Col.	CLB Slices	CLB Max Distr. RAM Kbits	Max I/O Pads
XC2V40	40K	8 x 8	256	8	88
XC2V80	80K	16 x 8	512	16	120
XC2V250	250K	24 x 16	1,536	48	200
XC2V500	500K	32 x 24	3,072	96	264
XC2V1000	1000K	40 x 32	5,120	160	432
XC2V1500	1500K	48 x 40	7,680	240	528
XC2V2000	2000K	56 x 48	10,752	336	624
XC2V3000	3000K	64 x 56	14,336	448	720
XC2V4000	4000K	80 x 72	23,040	720	912
XC2V6000	6000K	96 x 88	33,792	1,056	1104
XC2V8000	8000K	112 x 104	46,592	1,456	1108

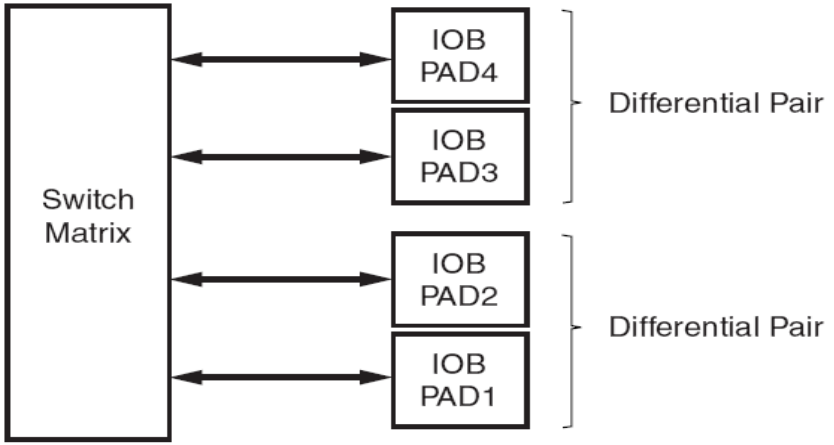


- Tabelle zeigt verschiedene Chip-Größen mit entsprechender Anzahl an CLBs und IOBs
- Da „nur“ reine Platzierung implementiert -> Routing-Ressourcen (RR) werden weitestgehend „ignoriert“
- RR werden schemenhaft zur Bestimmung der Verzögerung zwischen zwei Elementen verwendet (Schätzung der Verzögerung)

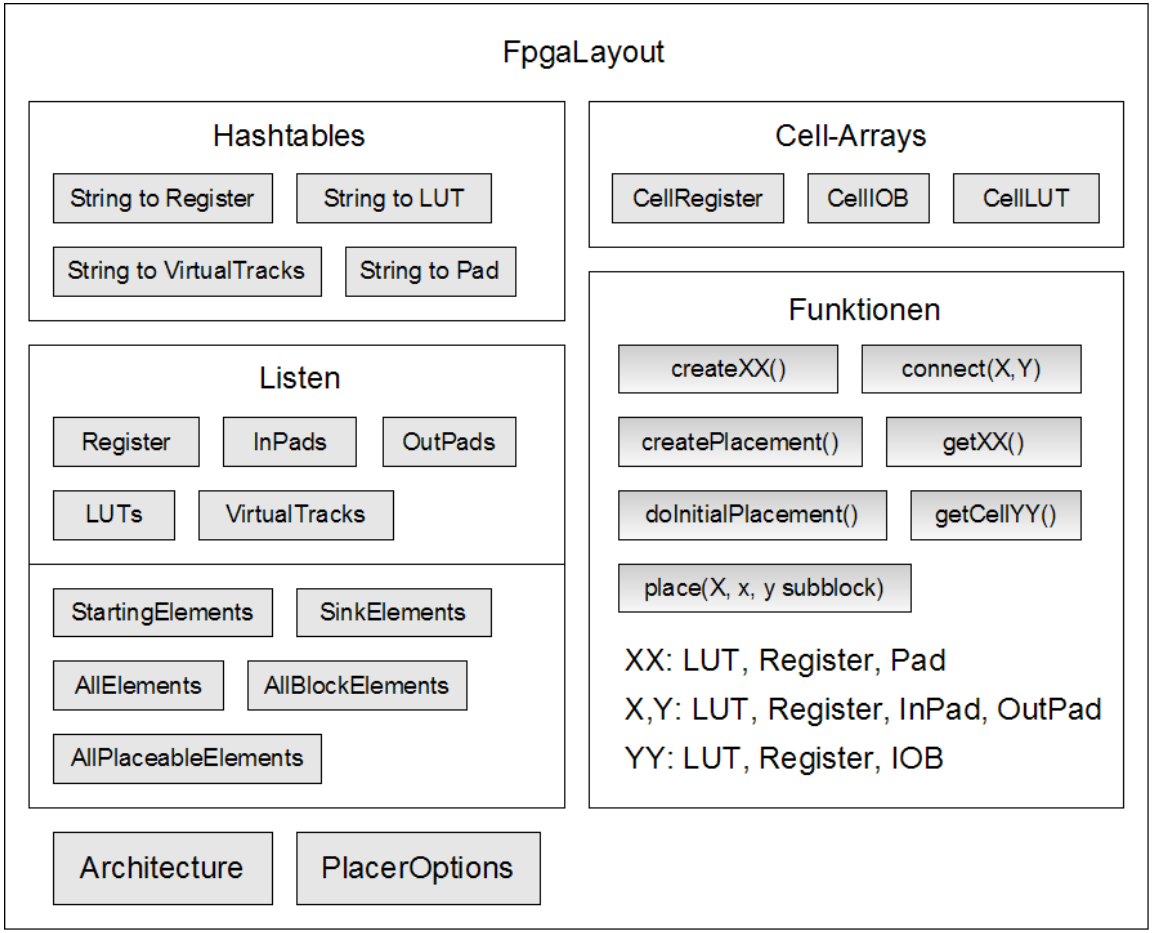
Einführung FPGA-Architektur

- CLBs bestehen aus 4 Slices
- Slice besteht aus 2x LUTs + Register
- LUT = einfacher 1-Bit breiter Speicher
- IOB Block besteht aus 4 IOB-Pads
- IOB-Pad = I/O-Pad + Register

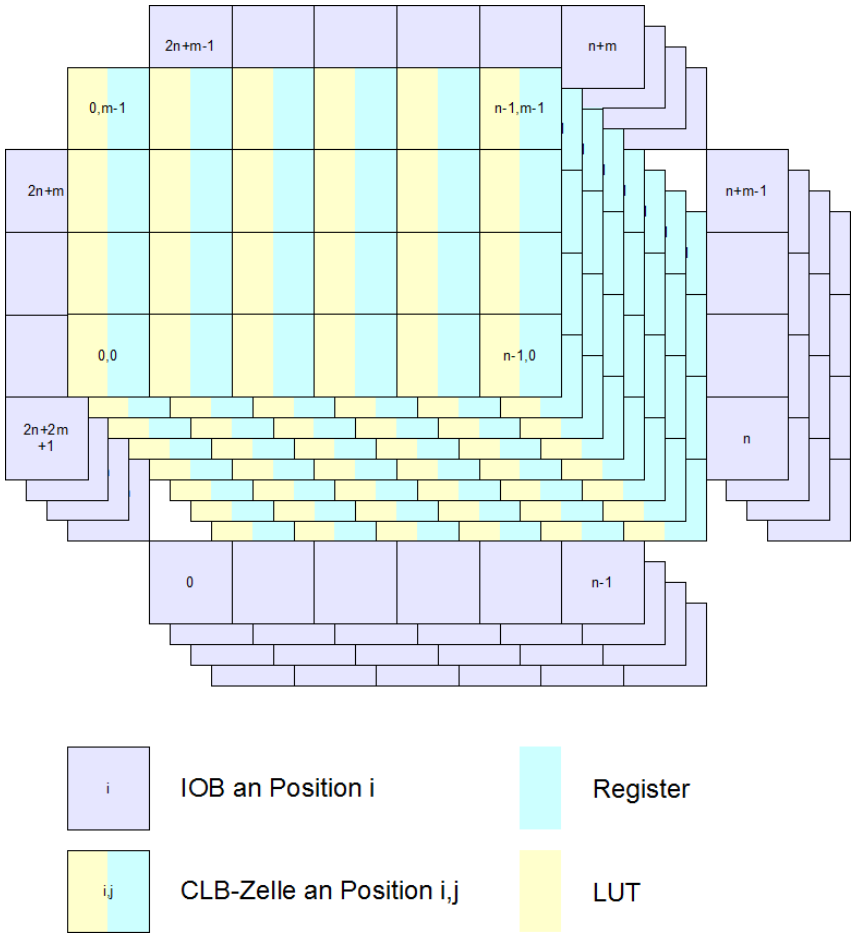




FPGA-Datenstruktur

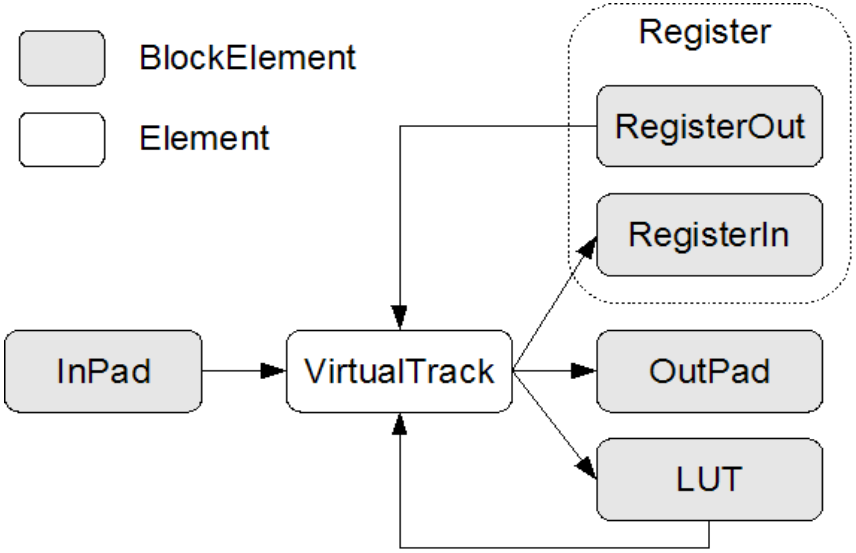


FPGA-Datenstruktur



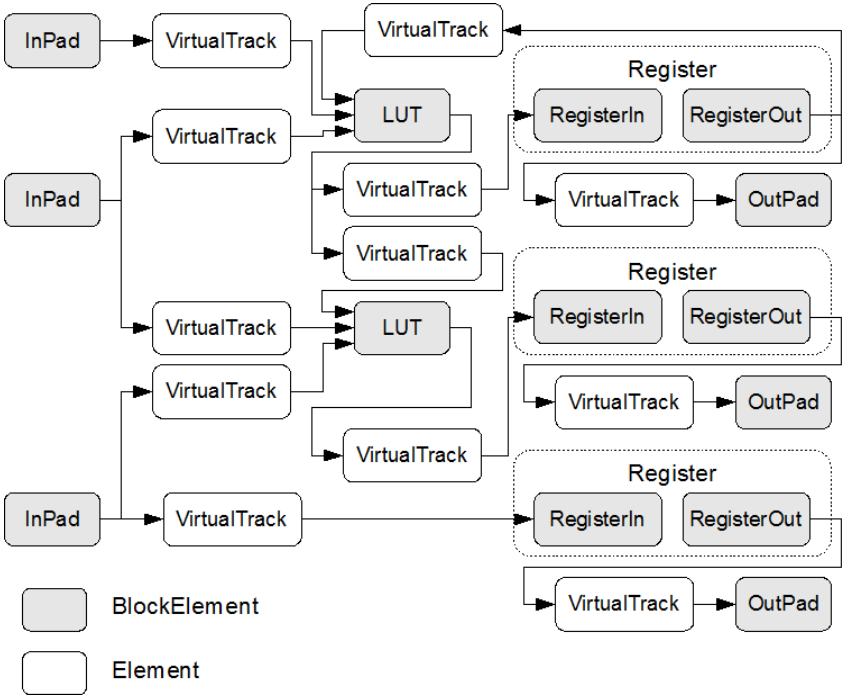
- Wird als Abstraktion der realen FPGA-Architektur verwendet
- Verwaltet die Schaltungselemente (anlegen, verbinden, platzieren, abrufen)
- Konsistenzprüfung (auch um Programmierfehler schneller zu finden)

FPGA-Datenstruktur
Infrastruktur



- Datenstrukturen für CLB/IOB
- VirtualTrack bildet Zeitverhalten des Routings nach (geschätztes Routing)
- BlockElement = Elemente, die beim Placement eine Koordinate auf dem FPGA zugeordnet wird

FPGA-Datenstruktur Infrastruktur

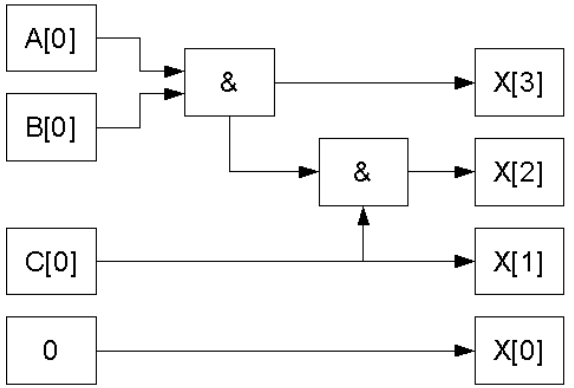


- Grafik zeigt erlaubt Verbindungen zwischen den Elementen
- Hier für synchrone Schaltkreise

Einführung Platzierung

- Platzierung (bei FPGAs) = Zuweisung einer Konfiguration zu den vorhandenen CLBs/IOBs auf dem FPGA, welche die gewünschte Funktionalität einer Funktionseinheit widerspiegelt
- **Fett** geschriebene Ausgangsbits ergeben n-Bit Binärzahl = „magische“ Zahl = CLB / LUT Konfiguration

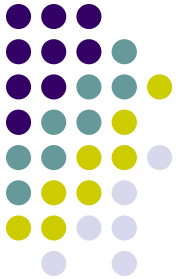
X[3:0] = {A[0] & B[0], ((A[0] & B[0]) & C[0]), C[0], 0}



Eingänge			Ausgang
x_1	x_2	x_3	y
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**

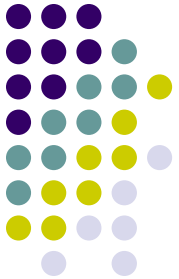
Eingänge			Ausgang
x_1	x_2	x_3	y
1	0	0	**1**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Einführung
Platzierung



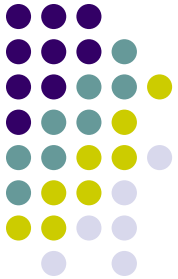
- Platzierung P ist eine Abbildung P: E(S) -> Pos(fpga,typ(E(S)))
- Eine Schaltung S ist ein Tupel S = (E(S),C(S)) = (E(S),E(S) x E(S)), E(S) = alle Elemente der Schaltung S, C(S) = Menge der Verknüpfungen, typ(E(S)) = Elementtyp von E, fpga = verwendetes FPGA, Pos(…) = Position des Elementes auf dem FPGA
- Zur Bestimmung der Qualität *q* einer Platzierung *P* wird eine Bewertungsfunktion *f* verwendet: *f* : *P* -> *q* = R_0^+
- Platzierungsalgorithmus ist ein Verfahren, welches *f*(*P*) = *q* minimiert (wenn kleine q besser)

Einführung
Platzierung



- Finden von optimaler Platzierung und optimalem Routing in einem Schritt zu komplex
- Aufspalten in zwei Schritte:
  - Bestimmen einer „guten“ Platzierung mit geschätztem Routing
  - Bestimmen eines „optimalen“ Routings zur berechneten Platzierung
- Gesamtergebnis nicht mehr optimal, sondern nur noch „suboptimal“
- Suboptimales Ergebnis wesentlich schneller bestimmbar
- Dadurch höherer Nutzen in der Praxis

Neue Platzierungsmodule



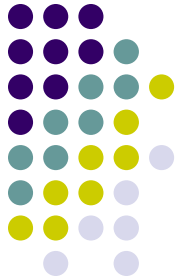
- TVPlace Placer Algorithmus
- Ext. TVPlace Placer Algorithmus
- Master Modul TVPlace Placer Algorithmus
  - Bounding Box TVPlace Placer Algorithmus
- Channel TVPlace Placer Algorithmus
- Meander TVPlace Placer Algorithmus
- Meander Search Placer Algorithmus

(Aus zeitlichen Gründen werden nur (Ext)TVPlace und Meander Search Placer vorgestellt)

TVPlace Placer Modul

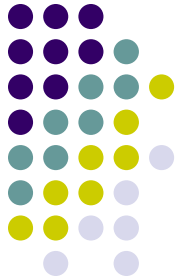
- Implementierung eines Timing-Driven Placers für FPGAs nach A.Marquardt, V.Betz und J.Rose
- Verwendet Simulated Annealing Verfahren, um eine möglichst gute Platzierung der Elemente zu berechnen
- Verwendet Kostenfunktion zur Bewertung einzelner Konfigurationen
  - Kann z.B. aus geschätzten Timing- und Verdrahtungskosten bestehen
- Ziel des Algorithmus: Minimierung der Gesamtkosten

TVPlace Placer Modul Funktionsweise



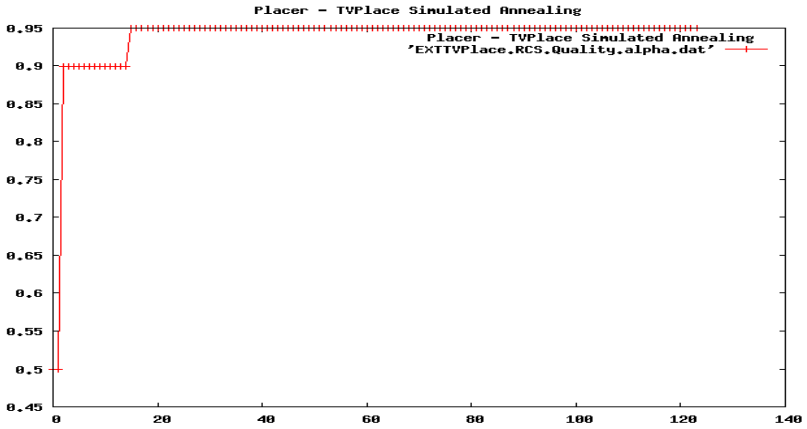
- Temperaturstufen: anfangs hohe Temperatur, sinkt mit der Zeit
- Zugweiten: Verringern von Veränderungen über die Zeit
- Akzeptanzrate: Anzahl akzeptierter Konfigurationen pro Temperaturstufe T normiert mit Gesamtzahl Konfigurationen in T
- Beginnt mit einer zufälligen Startkonfiguration
- Pro Temperaturstufe T:
  - Timing-Analyse, Berechnung der Kosten
  - Innere Schleife:
    - Generierung von Änderungen (Vertauschen von Elementen)
    - Berechnung der neuen Kosten (bzw. der Differenz)
    - Wenn Verbesserung -> akzeptiere immer
    - Wenn Verschlechterung -> akzeptiere u.U., abhängig von Temperaturstufe und Größe der Verschlechterung
  - Update der Temperaturstufe und Zugweite

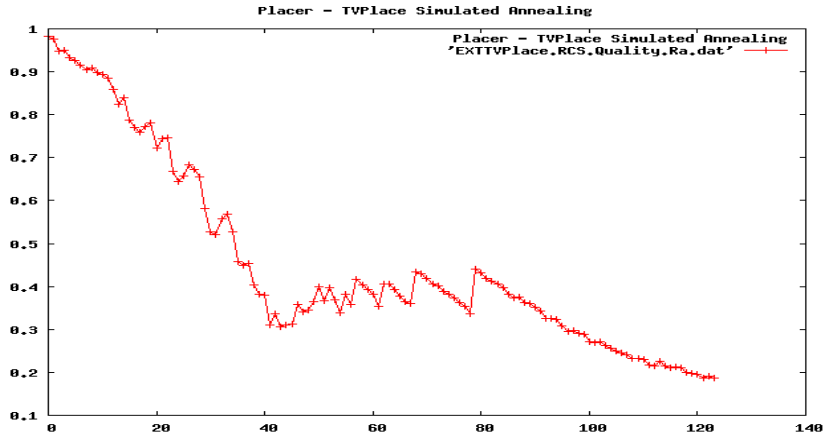
TVPlace Placer Modul Funktionsweise

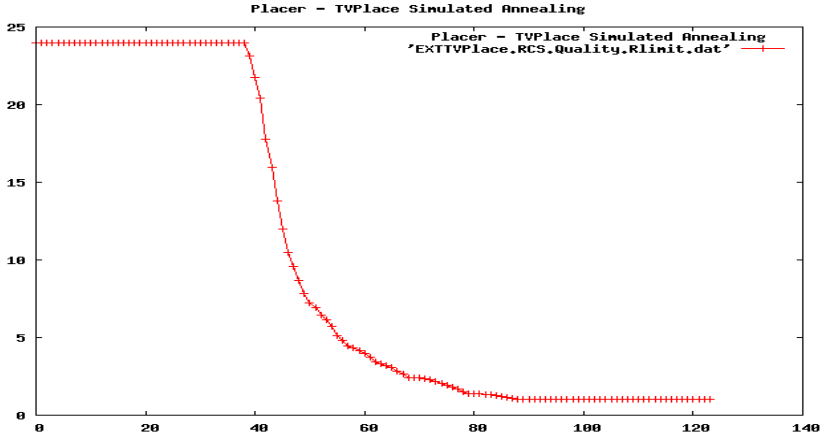


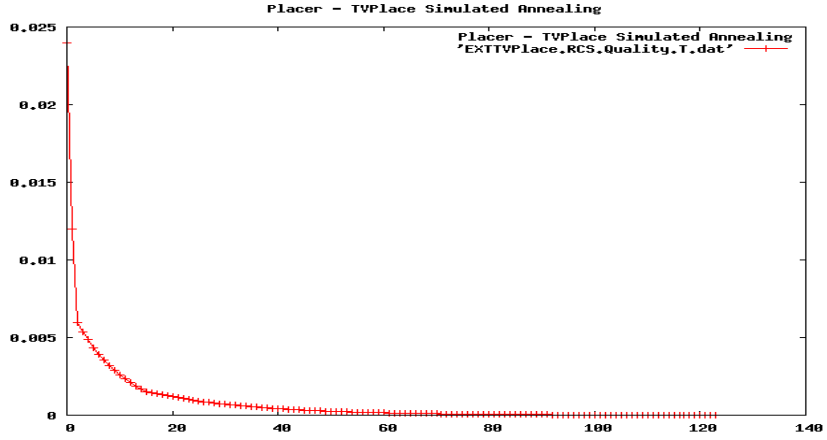
- Innere Schleife führt feste Anzahl an Iterationen aus
- Äußere Schleife endet, wenn bestimmtes Exit-Kriterium erfüllt wird (z.B. keine Verbesserung der besten Konfiguration seit x Temperaturstufen unterhalb einer bestimmten Temperatur)
- Beginnt mit großer Schrittweite, wird mit der Zeit eingeschränkt, um Veränderungen an der Schaltung zu reduzieren
- Akzeptanzrate wird zwischenzeitlich durch Veränderung der Zugweite um ca. 36% gehalten -> meiste Verbesserungen in diesem Bereich

TVPlace Placer Modul
Übliche Parameterkurven

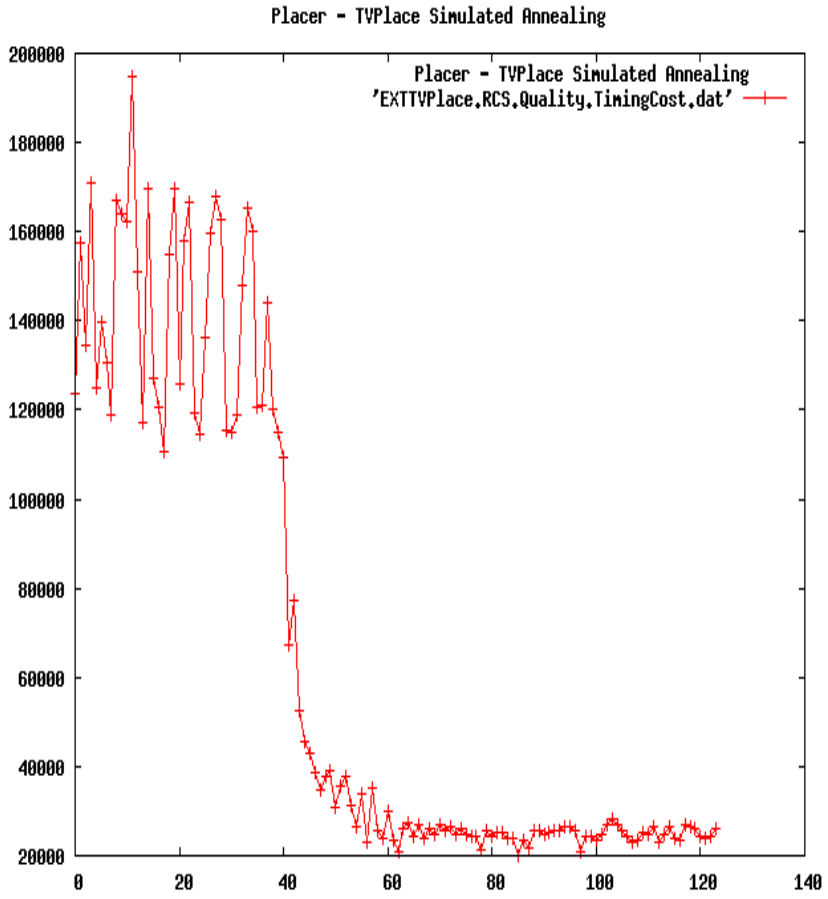


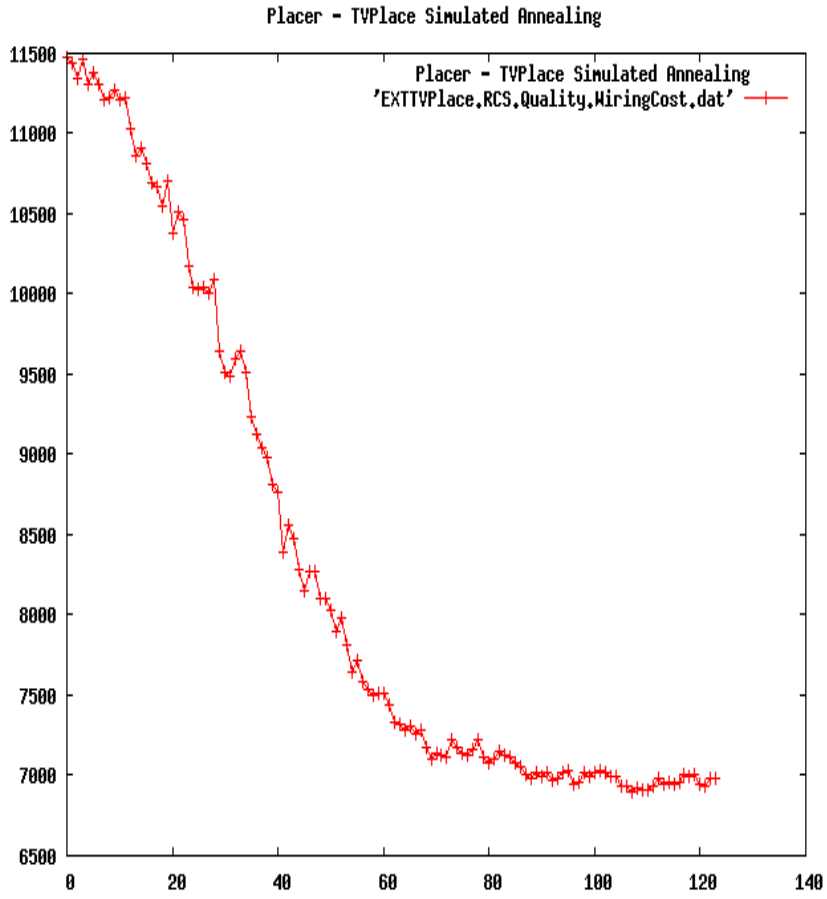






TVPlace Placer Modul
Übliche Parameterkurven

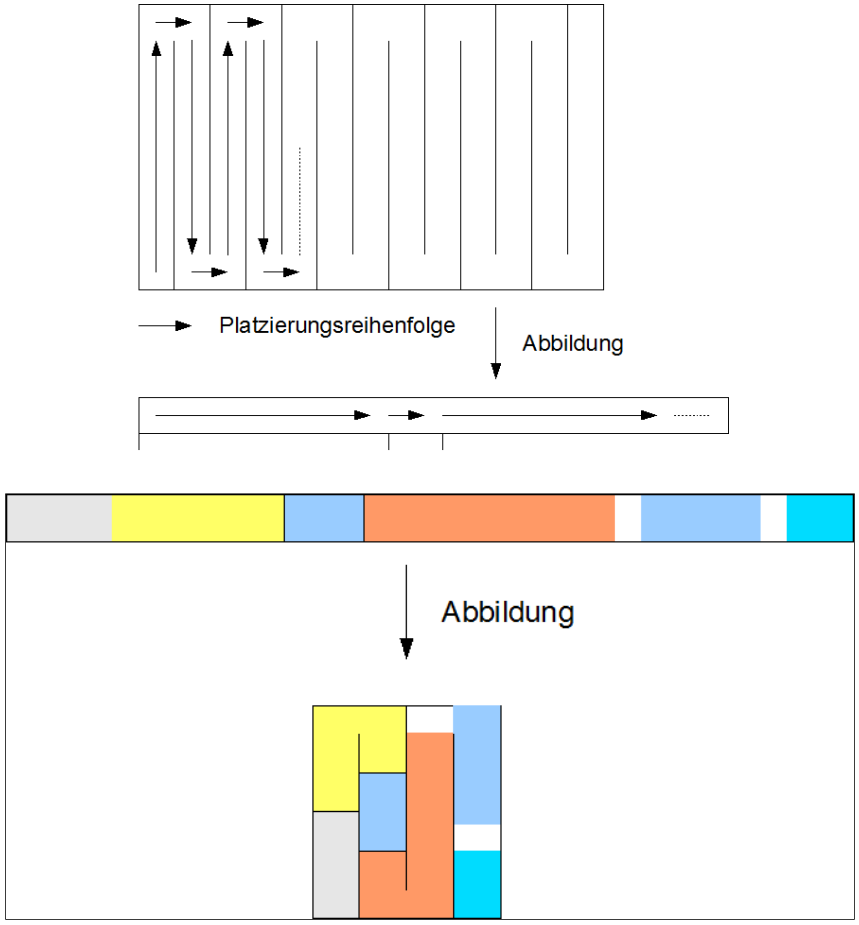




Ext. TVPlace Placer Modul

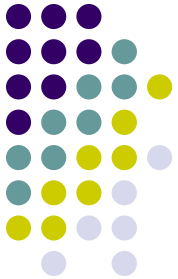
- Basiert auf dem TVPlace Placer Algorithmus
- Bei asynchronen Schaltungen identisch mit TVPlace Placer
- Bei synchronen Schaltungen werden LUTs und die vom ULG direkt zugeordneten Register gemeinsam verschoben
- Bringt bei synchronen Schaltungen eine Verbesserung von ca. 1% (bei den getesteten Schaltungen/Gleichungen) gegenüber TVPlace
- Wie TVPlace nur einzelne Elemente betrachtet
- Verhalten nicht Deterministisch! -> jede Ausführung erzeugt ein anderes Ergebnis

Meander Search Placer



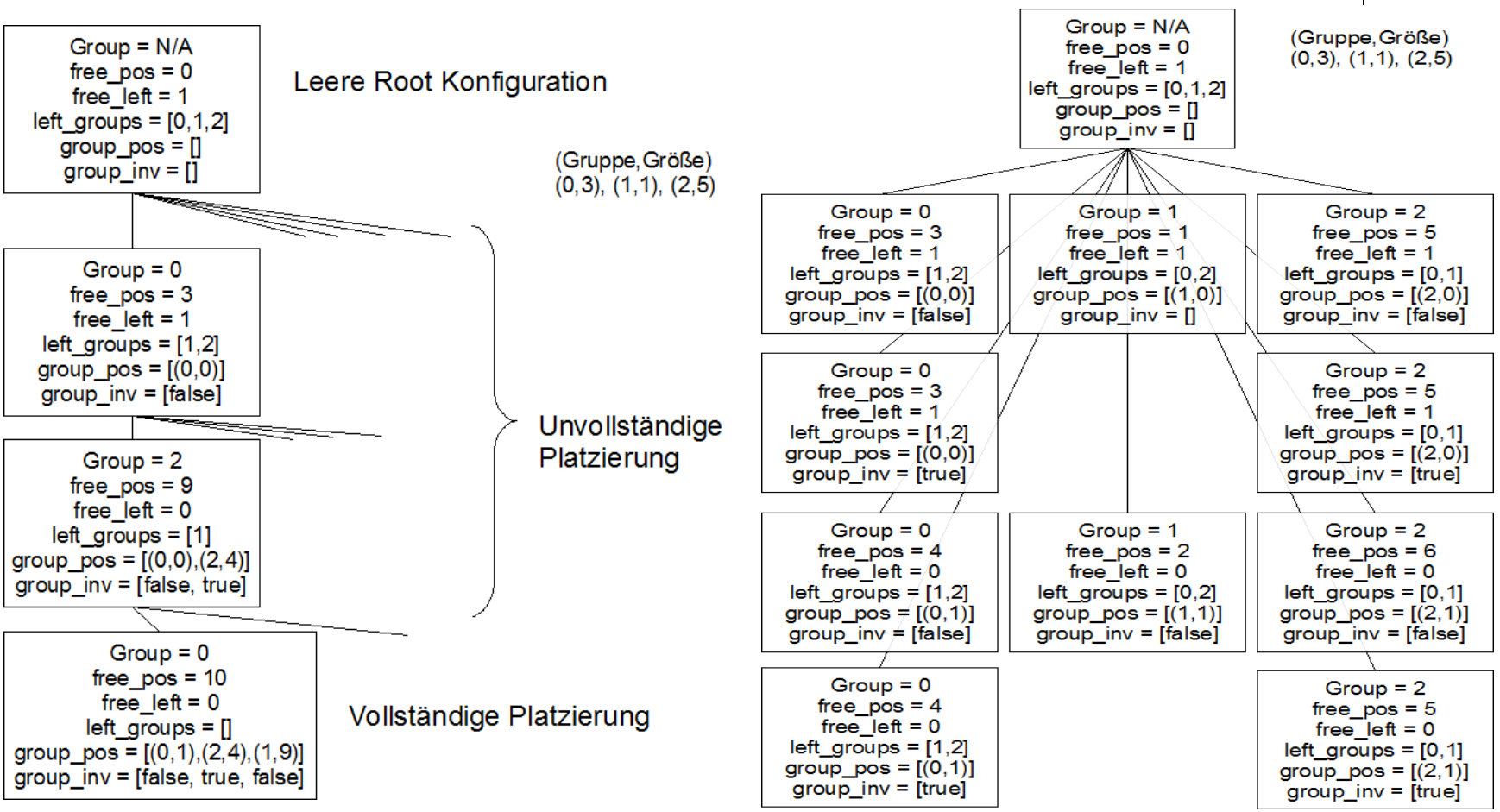
- Elemente werden zu Gruppen zusammengefasst
- Mäanderförmige Abbildung der 2D-Chip-Fläche auf ein 1D-Array
- Dadurch Reduktion der Komplexität der Platzierung von Gruppen
- Durch gewählte Abbildung: Benachbarte Elemente sind auch auf dem Chip benachbart -> wichtig!

Meander Search Placer

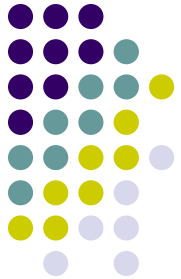


- Bestimmung der Lösung durch Suche im Konfigurationsraum
  - jede Konfiguration wird nur einmal betrachtet
  - deterministisches Verhalten
  - Schlechte Konfigurationspfade können während der Suche aussortiert werden
  - Es werden Elemente zu Gruppen zusammengefasst (a[0:7] = b[0:7] & c[0:7])
- Erstellen eines Konfigurationsbaumes

Meander Search Placer - Funktionsweise



Meander Search Placer - Funktionsweise

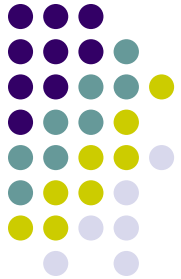


1. Bestimmung der „optimalen“ Abmaße der benutzten Chipfläche
2. Suche der „optimalen“ Konfiguration im Konfigurationsraum (zwei Teile):

Teil1:

- Beginne mit leerer Konfiguration (Wurzel des Baumes) -> Stufe 0 des Konfigurationsbaumes (Stufe i = i Gruppen auf dem FPGA platziert)
- Erstelle zu jeder Konfiguration auf Stufe i eine neue Konfiguration auf Stufe i+1, wobei je eine Gruppe verwendet wird, die noch nicht platziert wurde

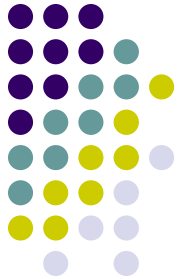
Meander Search Placer - Funktionsweise



1. Berechne zu jeder neuen Konfiguration DMAX_best und DMAX_worst
2. Konfigurationen mittels PriorityQueue nach DMAX_worst und #platzierter Gruppen sortieren
3. Dadurch relativ schnelle fertige Konfiguration k gefunden

- DMAX_best = wie DMAX, jedoch: normale geschätzte Verzögerung zwischen 2 platzierten Elementen, wenn mind. 1 nicht platziert -> minimal mögliche Verzögerung angenommen
- DMAX_worst = wie DMAX_best, bei mind. 1 nicht platziertem Element -> maximale Verzögerung angenommen

Meander Search Placer - Funktionsweise

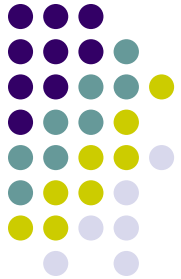


- Ist DMAX(k) = DMAX_best(root) = bestmögliche Verzögerung der Schaltung, dann ist optimale Platzierung gefunden, andernfalls:
- Es können immer noch Verbindungen auftreten, welche eine bessere Verzögerung erlauben (DMAX_best(k) < DMAX(k))

Teil 2:

2. Lösche alle verbliebene Konfigurationen k(i), deren DMAX_best(k(i)) > DMAX (k), da keine Verbesserung mehr erreicht werden kann
3. Übrig gebliebene Konfigurationen erneut in PriorityQueue nach DMAX_worst sortieren
4. Suche weiter, bis 4-fache Anzahl an Iterationen durchgeführt wurden, welche benötigt wurde, um Konfiguration k zu finden.

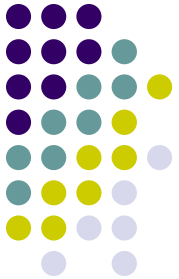
Meander Search Placer - Funktionsweise



1. Die bis dato beste gefundene vollständige Konfiguration wird verwendet.

- Die Beschränkung ist notwendig, damit nicht der gesamte Konfigurationsraum abgesucht wird.
- Die angegebene Anzahl an restlichen Iterationen ist willkürlich gewählt.
- Es hat sich jedoch herausgestellt, dass diese einen guten Mittelweg zwischen schneller Laufzeit und weiterer Verbesserung darstellt

Ergebnisse



- Die Placer Module „TVPlace“, „ExtTVPlace“ und „MeanderSearch“ liefern gute Ergebnisse
- Verbesserungen bei DMAX bis ca. -53%, bei der Chipfläche bis ca. Faktor 1/8 bei den getesteten Schaltungen
- Abhängig von der Schaltung auch „Master Modul TVPlace“ halbwegs gute Ergebnisse
- MeanderSearch liefert teilweise bessere Ergebnisse wie (Ext)TVPlace, jedoch bis 4-fache Rechenzeit und 10-facher Speicherbedarf
- (Ext)TVPlace stark Laufzeit-optimiert, MeanderSearch nicht optimiert
- TVPlace und ExtTVPlace unterscheiden sich kaum bei Schaltungen mit Register („nur“ 1% Verbesserung)

Ergebnisse

Placer	Chip-	DMAX [ps]		Xilinx TA		
	größe	mit Pads	ohne Pads	maximum combinational path delay [ps]	minimum input required time before clock [ps]	maximum output delay after clock [ps]
ohne Register						
Wewetzer	62x1	24600	21930	34868		
TVPlace	3x3	17000	14330	26779		
ExtTVPlace	3x3	17100	14330	24892		
MMTVPlace	1x36	17030	14360	31413		
ChannelTVPlace	8x1	28090	14170	33923		
MeanderTVPlace	4x2	22140	18920	31753		
MeanderSeachPlacer	4x2	16510	13640	23529		
mit Register						
Wewetzer	62x1	23240	22280	3586	22709	15332
TVPlace	3x3	15690	14730	2264	18373	9079
ExtTVPlace	3x3	15460	14500	2234	17118	8962
MMTVPlace	1x38	15670	14710	3532	22075	13571
ChannelTVPlace	8x1	15130	14170	2256	19019	9057
MeanderTVPlace	4x2	20360	19050	2234	23623	9226
MeanderSeachPlacer	4x2	15130	14170	2234	16991	9079

Tabelle 8.8: Ergebnisse mit Eingabedaten aus C,3

Ergebnisse

Placer	Chipgröße	DMAX [ps]		Zeit [s]	RAM [MB]
		mit Pads	ohne Pads		
ohne Register					
Wewetzer	62x1	24600	21930	5.1	15
TVPlace	3x3	16956	14276	2.1	1.9
ExtTVPlace	3x3	16926	14246	2.1	2.6
MMTVPlace	1.1x29.9	17051	14381	7.3	24.8
ChannelTVPlace	8x1	28090	14170	5.6	15.2
MeanderTVPlace	4x2	21466	18506	20.3	14
MeanderSeachPlacer	4x2	16510	13640	7.1	17
mit Register					
Wewetzer	62x1	23240	22280	5.1	13
TVPlace	3x3	15610	14650	3.1	2.5
ExtTVPlace	3x3	15468	14516	3	2.5
MMTVPlace	1.1x35.4	15730	14770	12.4	105.6
ChannelTVPlace	8x1	15309	14349	6.4	16
MeanderTVPlace	4x2	20302	19256	25.7	14
MeanderSeachPlacer	4x2	15130	14170	8.1	17

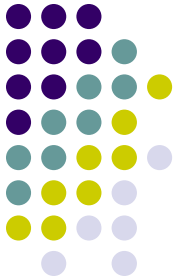
Tabelle 8.4: Ergebnisse mit Eingabedaten aus C,3

Ergebnisse

Placer	Datensatz			
	1	2	3	4
ohne Register				
Wewetzer	±0%	±0%	±0%	±0%
TVPlace	+4.21%	+1.51%	-34.90%	-51.41%
ExtTVPlace	+4.03%	+2.07%	-35.04%	-51.22%
MMTVPlace	+1.51%	+99.43%	-34.42%	-50.45%
ChannelTVPlace	+0.35%	+1.27%	-35.39%	-47.29%
MeanderTVPlace	+63.52%	+31.43%	-15.61%	-29.38%
MeanderSeachPlacer	-4.41%	-4.00%	-37.80%	-48.18%
mit Register				
Wewetzer	±0%	±0%	±0%	±0%
TVPlace	+4.02%	+2.78%	-34.25%	-49.76%
ExtTVPlace	+3.30%	+2.04%	-34.85%	-49.84%
MMTVPlace	+2.47%	+3.13%	-33.71%	-48.89%
ChannelTVPlace	-1.07%	+1.61%	-35.60%	-42.64%
MeanderTVPlace	+72.33%	+27.89%	-13.57%	-28.68%
MeanderSeachPlacer	-3.08%	-1.36%	-36.40%	-42.31%

Tabelle 8.6: Vergleich von DMAX (ohne Pads) bezogen auf den Wewetzer Placer

Zusammenfassung



- ULG um Kommandozeilenparameter und einer Placer-Modul-Auswahl erweitert
- 6 neue Placer-Module implementiert
- Ein Teil davon liefert schaltungsabhängig deutlich bessere Ergebnisse
- Beschleunigte Platzierung durch zusammenfassen von Elementen konnte nicht erreicht werden (Vergleich TVPlace vs. MeanderSearch)