Non-Standard Rechner

Peter Marwedel
Informatik 12
TU Dortmund

2014年04月21日

2.7 ASIPs

Application-specific instruction set processor (ASIP):

Befehlssatz aufgrund der Anwendung festgelegt.

Einsatz:

- U.a. für effiziente eingebettete Prozessoren

Frühe Beispiele:

- Yasuura et al. (Fukuoka, Japan): Generische Architektur, u.a. Datenwortbreiten anwendungsabhängig.
- I.-J. Huang: Weglassen unbenötigter Befehle beim 6509.

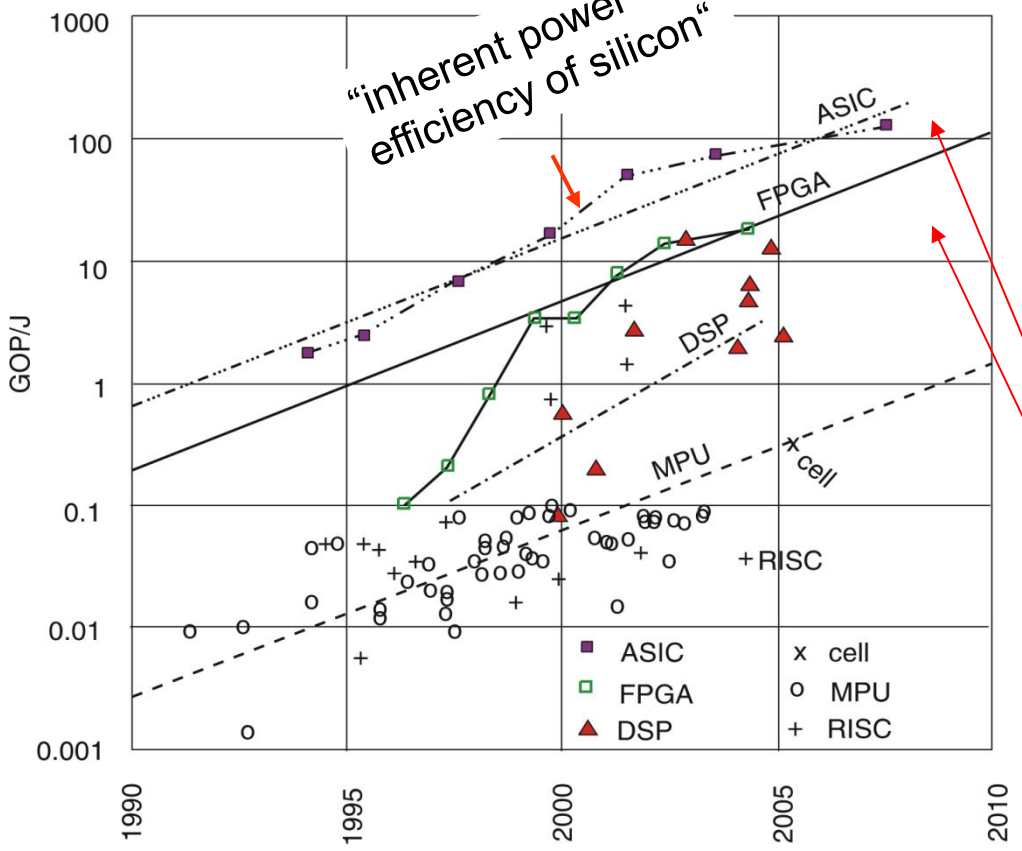
Wissenschaftliche Fragestellung:

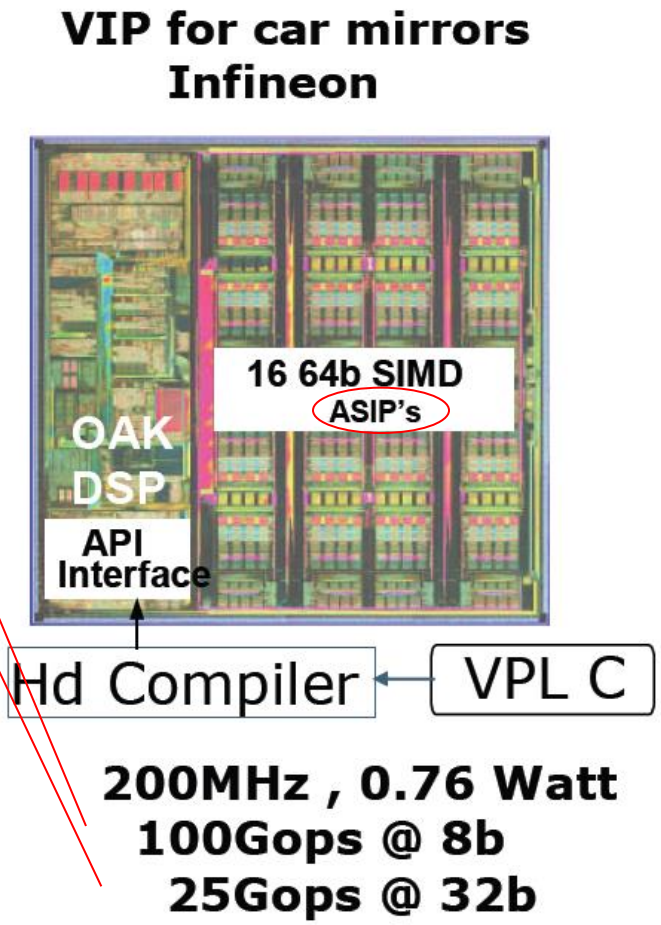
- Wie konstruiert man systematisch „den besten“ Befehlssatz zu einer Anwendung?

Literatur

- M. K. Jain, M. Balakrishnan, Anshul Kumar: *ASIP Design Methodologies* : Survey and Issues, VLSI Design, 2001
- M. Gries, K. Keutzer (Hrg.): *Building ASIPs: The Mescal Methodology*, Springer, 2005
- P. Ienne, R. Leupers (Hrg.): *Customizable Embedded Processors,* Morgan Kaufmann, 2006
- O. Schliebusch, H. Meyr, R. Leupers: *Optimized ASIP Synthesis from Architecture Description Language Models*, Springer, 2007
- T. Shiro, M. Abe, K. Sakanushi, Y. Takeuchi, M. Imai: *A Processor Generation Method from Instruction Behavior Description Based on Specification of Pipeline Stages and Functional Units*, ASP-DAC, 2007
- C. Wolinski, K. Kuchcinski: *Automatic selection of application-specific reconfigurable processor extensions*, DATE, 2008
- Laura Pozzi, Kubilay Atasu, Paolo Ienne: *Exact and approximate algorithms for the extension of embedded processor instruction sets*. IEEE Transactions on CAD, 2006.
- Tensilica Inc.: The xpres compiler: Triple-threat solution to code performance challenges. Tensilica Inc Whitepaper, 2005.

Existenznachweis der Energieeffizienz





Close to power efficiency of silicon

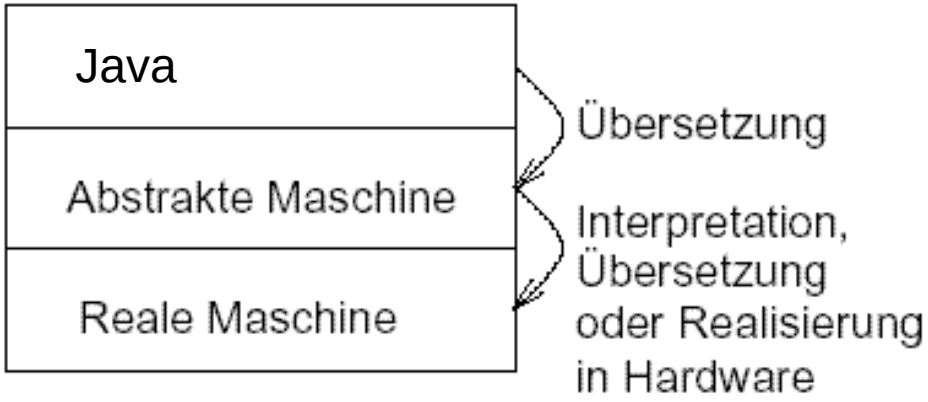
© Hugo De Man: From the Heaven of Software to the Hell of Nanoscale Physics: An Industry in Transition, *Keynote Slides*, ACACES, 2007

2.8 Abstrakte Maschinen

Maschinen, die jeweils einen abstrakten Befehlssatz interpretieren.

Programme werden in Befehle der abstrakten Befehlssätze übersetzt, nicht direkt in Maschinenbefehle realer Maschinen.

Beispiele: abstrakte Befehlssätze zur Realisierung von Java, (UCSD-) Pascal, PROLOG, LISP, FORTH, Smalltalk usw..



Lediglich der Interpreter der abstrakten Befehlssätze muss für verschiedene Maschinen jeweils neu erzeugt werden.
Nachteil: niedrigere Ausführungsgeschwindigkeit.

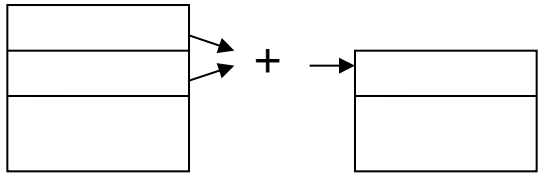
Java Virtual Machine

Java:

- Objektorientierte Programmiersprache
- Datentypen: *byte, short, int, long, float, double, char*
- Unterstützt Netzwerk-Programmierung
- Im Hinblick auf Sicherheit entworfen:
  - keine Pointer-Manipulation wie in C, C++.
  - beschränkte Möglichkeit, Informationen über die momentane Umgebung zu erfahren
- Automatische Freispeicherverwaltung
- *Multithreading*
- Über Netze auf alle relevanten Maschinen zu laden.
- Java-Programme können dort ausgeführt werden, wo die JVM realisiert ist.

Eigenschaften der JVM

- Die JVM ist eine Kellermaschine,
- Operationscodes in einem Byte kodiert,
- Typische Befehle:
  - lade integer auf den Keller,
  - addiere die obersten Kellerelemente,
  - starte nebenläufige Ausführung,
  - synchronisiere nebenläufige Ausführung,
  - Befehle zur Realisierung der Objektorientierung



- Byte-Code ist kompakt (≈ 1/3 der Größe von RISC-Code). Wichtig, wenn Code auf Prozessorchip zu speichern ist.
- JVM verzichtet weitgehend auf *Alignment*-Beschränkungen

3 Methoden der Realisierung von JVMs:

1. Durch **Interpretation** von JVM-Befehlen in Software,

2. Durch Übersetzung in den Maschinencode der aufrufenden Maschine unmittelbar vor der Ausführung; *just-in-time compilation*.

3. Durch Realisierung einer JVM als „echte" Maschine.

Warum interessiert man sich für eine „echte" Java-Maschine?

Java inside

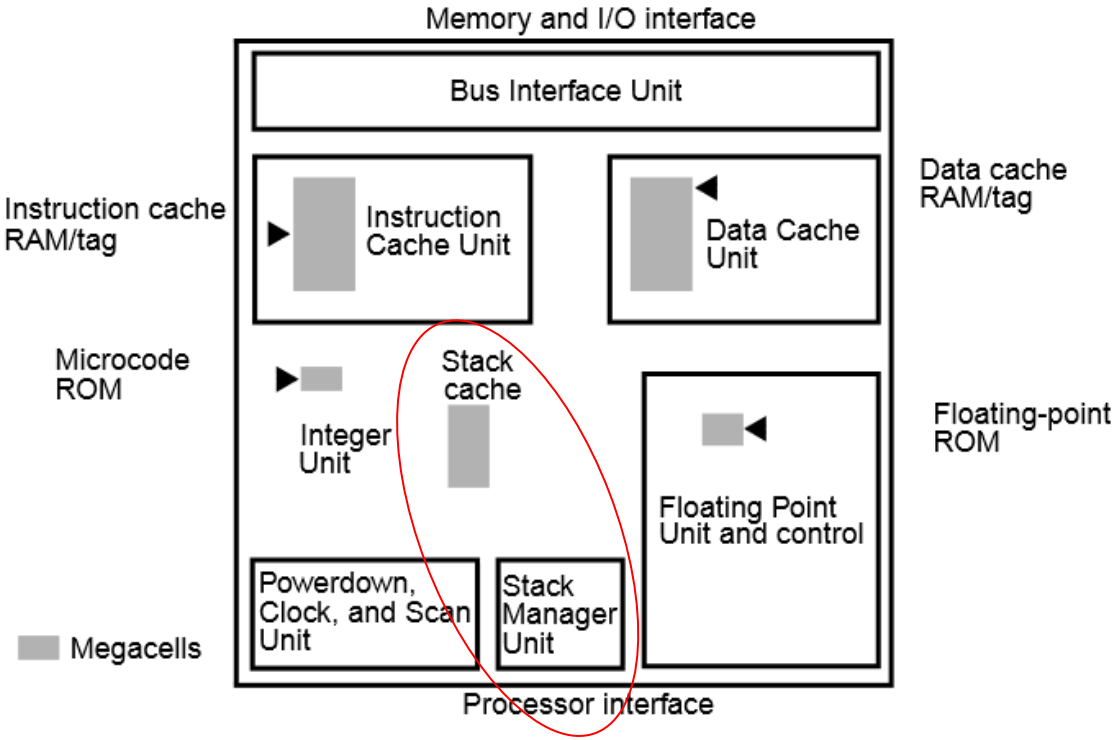
© Microsoft

☞ Exemplarische Betrachtung von Entwicklungszielen

PicoJava

- Ziel v.a. Verbesserung der Performance und für den Einsatz in vernetzten eingebetteten Systemen (GPS, Mobiltelephon, Netzwerktechnik, Chipcard-Systeme ..)
- PicoJava (I) (Sun, 1997): Spezifiziert, aber nie realisiert
  - Oberste Kellerelemente im Prozessor in schnellem Speicher
  - Häufigste Befehle in Hardware ausgeführt, CPI=1.
  - Komplexere Befehle: Mikroprogramm.
  - Ganz komplexe Befehle: Interrupt + Software
- PicoJava II (Sun, 1999): Spec frei verfügbar
  (u.a. http://www1.pldworld.com/@xilinx/html/pds/HDL/picoJava-II/DOCS/pj2-uarch-guide.pdf

PicoJava (2)



Drippling-Mechanismus für stack

Ausgangsbasis für Realisierungen in FPGAs etc.

Komodo (1)

Design: S. Uhrig (jetzt Fak. ET+IT, TU Dortmund!), Ungerer, ..

© Microsoft

- Schwergewicht auf Realzeitfähigkeit
- Mehrfädig (engl. *multi-threaded*)
- *Thread-tag* wird im Fließband mitgeführt
- Prioritäten Manager selektiert nächsten Befehl:
  - ***Fixed Priority Preemptive*** (FPP): #(Prioritätsebenen) = #(Hardware-*Threads*)
  - ***Earliest Deadline First*** (EDF): Jedem Hardware-*Thread* muss eine 32-Bit Deadline zugeordnet werden.
  - ***Least Laxity First*** (LLF): Jedem Hardware-*Thread* 32-Bit *Laxity* = (*Deadline*-Laufzeit) zugeordnet.
  - ***Guaranteed Percentage*** (GP): Jeder *Thread* erhält Prozentanteil (genau, minimal oder maximal)

Komodo (2): Realzeitfähigkeit der Garbage-Collection

Anforderungen:

- Inkrementell
- Zeitlich vorhersagbar (feste obere Zeitschranken)
- Möglich (verfügbare Zeit nicht überschreiten)
- Robust (wenige Fehlermöglichkeiten für Programmierer)
- Effizient (wenig Prozessorzeit)
- Wenig Synchronisation (keine Pausen für Anwendungen)
- Nicht-kopierend

© Microsoft

Lösung

- Ausnutzen der Multi-*threading* Möglichkeit
- Anpassen eines existierenden Algorithmus an Real-Time-Anforderungen

S. Fuhrmann, M. Pfeffer, J. Kreuzinger, Th. Ungerer, U. Brinkschulte: Real-Time Garbage Collection for a Multithreaded Java Microcontroller, 4th International Symposium on Object-Oriented Real-Time Distributed Computing, 2001

JOP - Java Optimized Processor

- *JOP is a processor designed to implement the JVM in hardware.*
- *It is part of a PhD thesis at the Vienna University of Technology*
- *Goal a simple and small processor optimized to execute Java*
- *Thread support can simplify development of embedded systems*
- *Common implementations of the JVM as interpreter or just-in-time compiler, are not practical.*
- *http://www.jopdesign.com/ describes JOP (a Java Optimized Processor) is a hardware implementation of the JVM with predictable execution time for embedded real-time systems.*
- *Due to the small size, it can be implemented in a low cost FPGA. For low volume systems, the flexibility of an FPGA can be important*
- *Complete VHDL source and tools in Java are available for download from http://www.jopdesign.com/ .*

http://www.jopdesign.com/

Jazelle DBX (Direct Bytecode Execution)

- Erlaubt es einigen ARM-Prozessoren Java Bytecode als 3. Betriebsmodus neben ARM und Thumb Modi
- Gestartet mittels „*branch to Java*“ Befehl BXJ
- Der erste Prozessor war der ARM926EJ-S
- Häufig ausgeführte Codes werden in Hardware ausgeführt (gemäß ARM, 95% der Befehle in HW ausgeführt)
- Andere Codes werden in SW ausgeführt
- Basis: schnelle *binary translation*: JVM Codes werden während der Ausführung in ARM-Befehle übersetzt
- *just-in time translation* nicht mehr notwendig

http://en.wikipedia.org/wiki/Jazelle

Nicht-Von-Neumann-Maschinen

Wir geben die sequentielle Ausführung von Programmcode auf ….

Gibt es Rechner jenseits der von-Neumann Rechner?

Reconfigurable Logic

Custom HW may be too expensive, SW too slow.

Combine the speed of HW with the flexibility of SW

- ☞ HW with programmable functions and interconnect.
- ☞ Use of configurable hardware;
  common form: field programmable gate arrays (FPGAs)

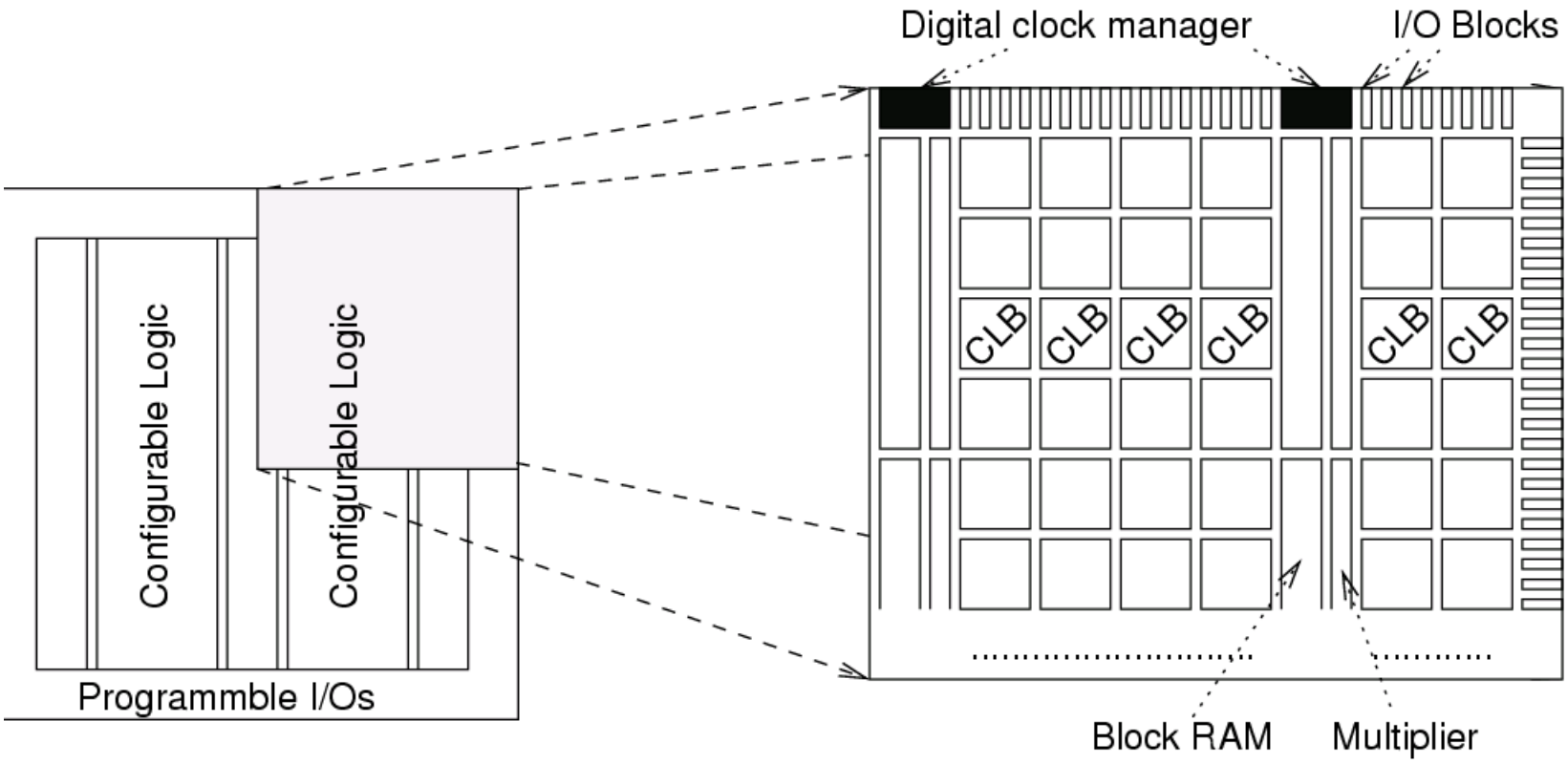
Applications:

- algorithms like de/encryption,
- pattern matching in bioinformatics,
- high speed event filtering (high energy physics),
- high speed special purpose hardware.

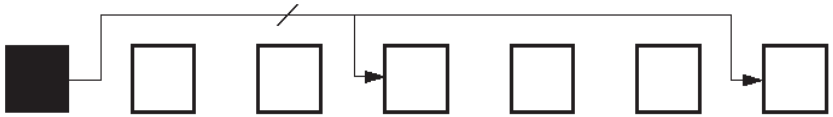
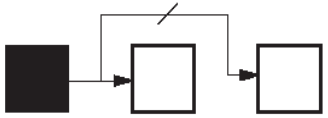
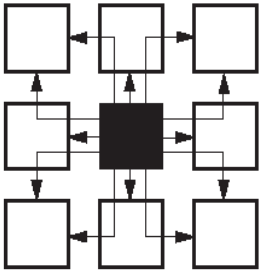
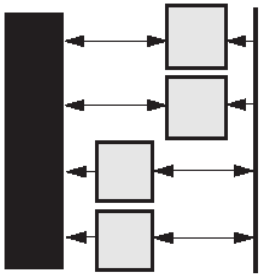
Very popular devices from

- XILINX, Actel, Altera and others

Floor-plan of VIRTEX II FPGAs



Interconnect for Virtex II

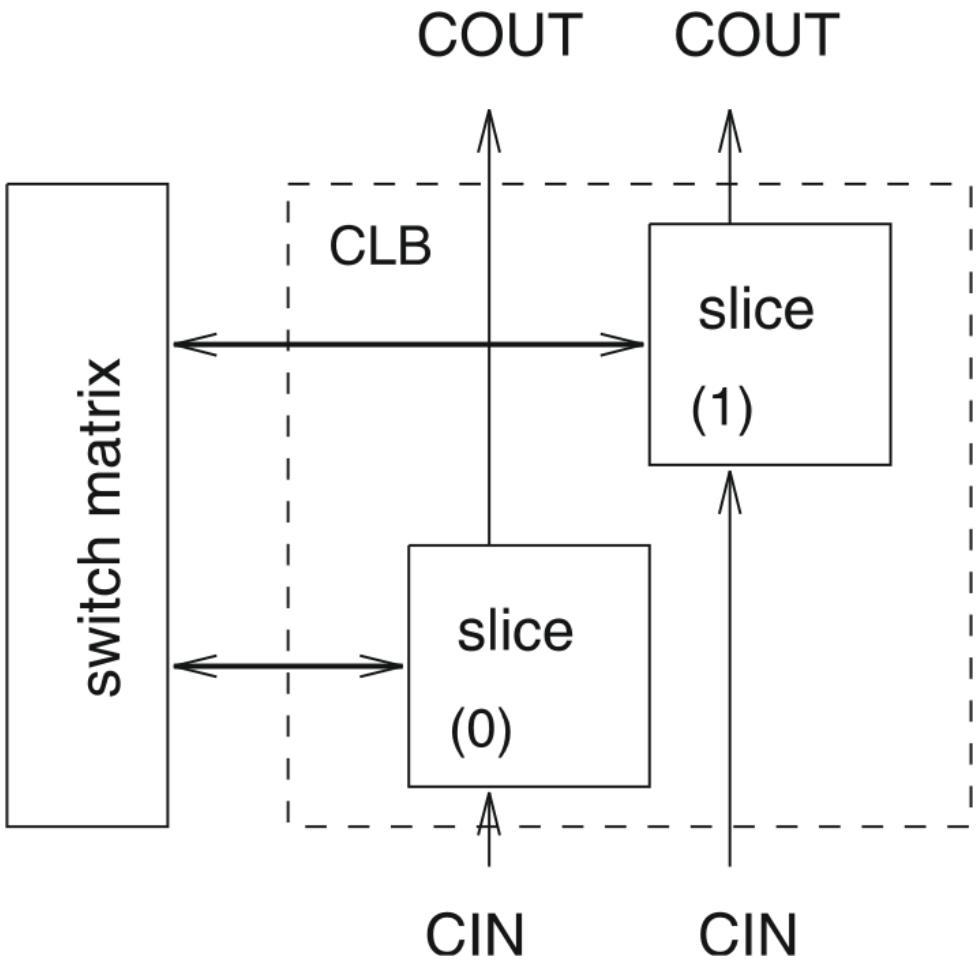
24 Horizontal Long Lines 24 Vertical Long Lines	
120 Horizontal Hex Lines 120 Vertical Hex Lines	
40 Horizontal Double Lines 40 Vertical Double Lines	
16 Direct Connections (total in all four directions)	
8 Fast Connects	

Hierarchical Routing Resources;

More recent: Virtex 5, 6, 7

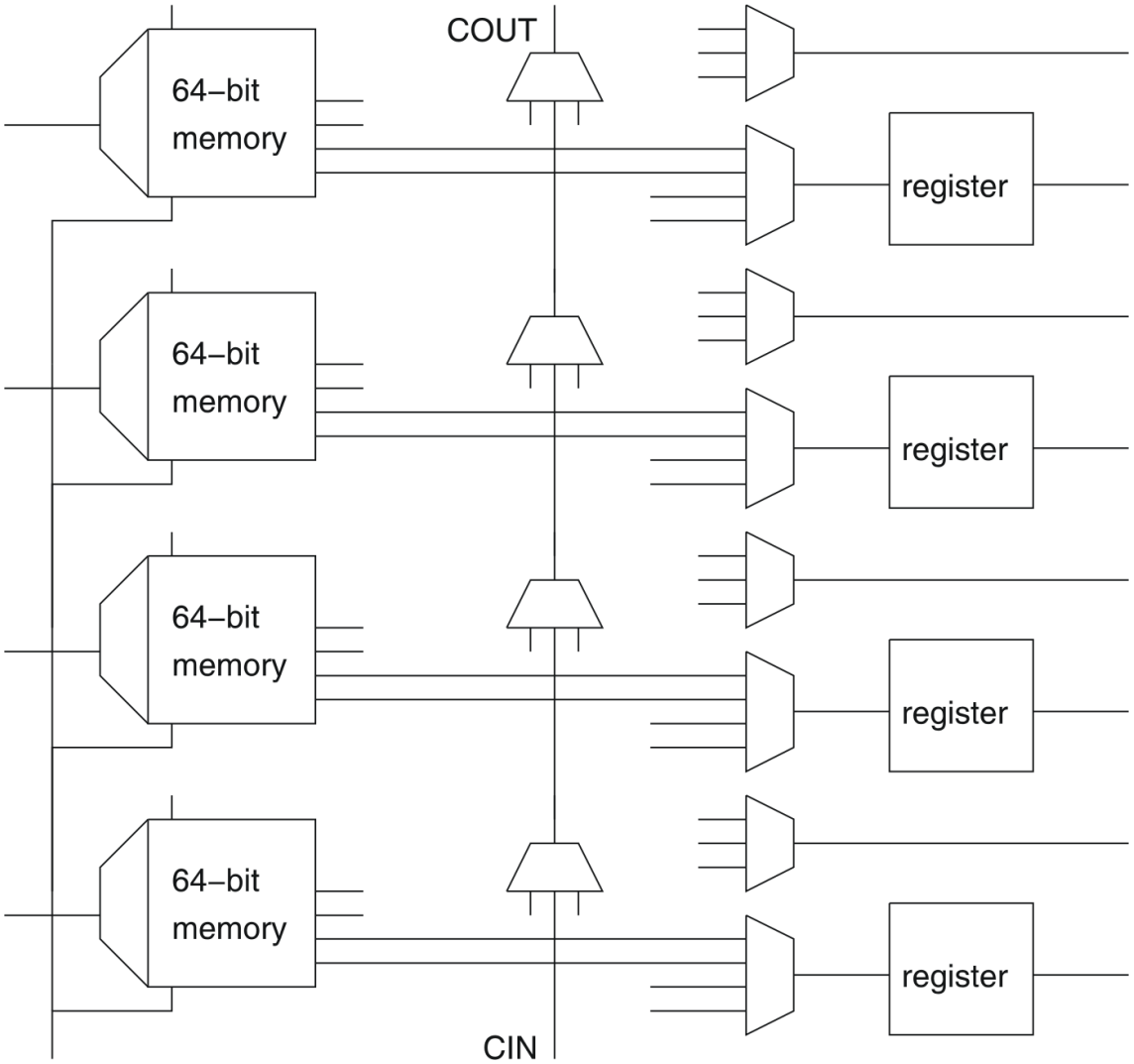
no routing plan found for Virtex 7.

Virtex 7 Configurable Logic Block (CLB)



http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

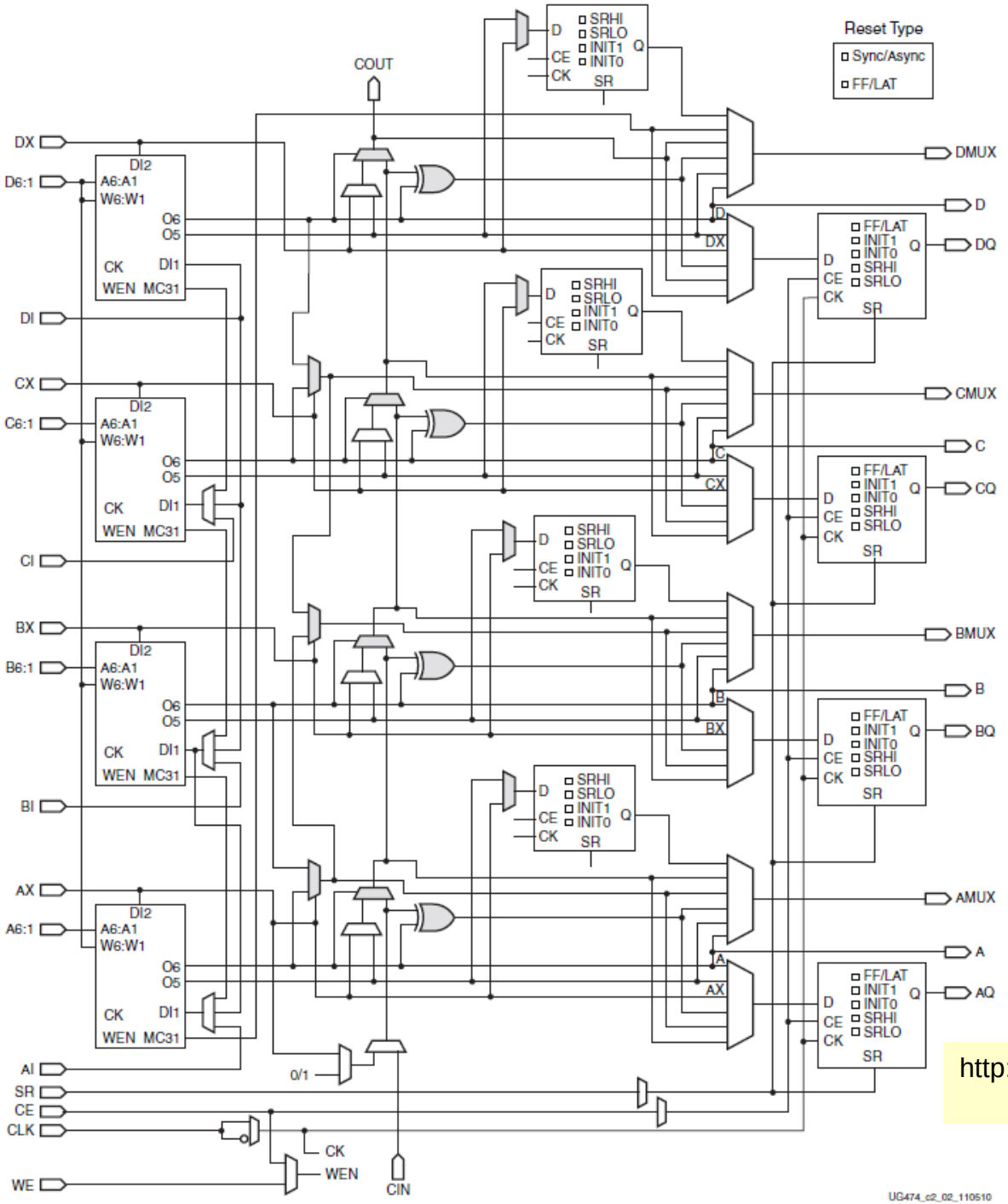
Virtex 7 Slice (simplified)



Memories typically used as look-up tables to implement any Boolean function of ≤ 6 variables.

Processors typically implemented as “soft cores” (microblaze)

Virtex 7 SliceM



SliceM supports using memories for storing data and as shift registers

Enables implementation of many non-standard architectures ☞

http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

Zusammenfassung

- *Application Specific Instruction Set Processors*
- Abstrakte Maschinen
- *Field programmable gate arrays* (FPGAs)