



Kolloquium zur Projektarbeit des Moduls ET-INF-D-900

Portierung eines geeigneten LZ-basierten Kompressors auf LegUp-HLS

Jens Voß – jens.voss@mailbox.tu-dresden.de

Dresden, 11.02.2015



DRESDEN
concept
Exzellenz aus
Wissenschaft
und Kultur

Gliederung

- Aufgabenstellung
- LZRW und LegUp
- Erweiterungen
- Analyse
- Fazit

Aufgabenstellung

Portierung eines geeigneten LZ-basierten Kompressors auf LegUp-HLS

- Anpassung des gegebenen LZRW C-Quellcodes
- Generierung einer FPGA-Implementation mit LegUp
- Beseitigung entstandener Probleme
- Analyse der entstandenen Schaltung(en)

LZRW

- von Dr. Ross Williams 1991 entwickelt
- Kompression mittels History
- Hash-Tabelle zur schnelleren Verweissuche

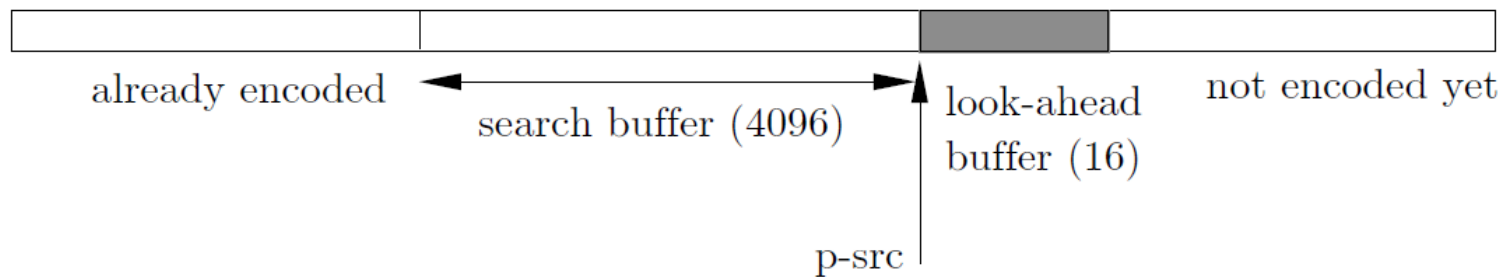


Abb.: LZRW Buffer, vgl. [4]

LZRW

- Unterteilung Ausgabedaten in Blöcke von 16 Items
- Dekompression mittels Informationen aus Verweisen und Steuerbits

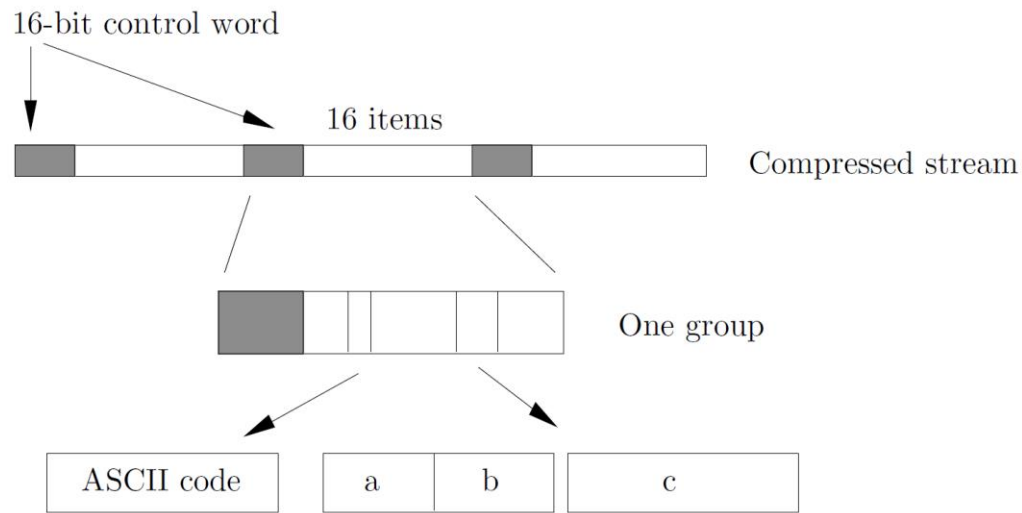


Abb.: LZRW Output, vgl. [4]

LegUp

- Open Source High-Level-Synthese Framework
- entwickelt an der University of Toronto
- Aktuell: Version 3.0 (Januar 2013)
- Ziel: FPGA-Entwicklung für Softwareentwickler vereinfachen / attraktiver gestalten
- unterstütze FPGA:
 - Altera Stratix IV (EP4SGX530KH40C2)
 - Altera Cyclone II (EP2C35F672C6)

LegUp

- ermöglicht Synthese von C Programmen
- großes Subset von ANSI-C wird unterstützt:
 - Kontrollflussfunktionen
 - alle arithmetischen und logischen Operationen
 - Funktionen, Structs, Arrays
 - vollständige Pointer-Arithmetiken
 - Seit Version 3.0: Floating-Point Operationen

LegUp

- nicht unterstützt:
 - dynamisch allozierter Speicher
 - Rekursion

LegUp

- 4 Methoden zur Synthese:
 - Software Flow
 - Processor/Accelerator Hybrid Flow
 - Hardware Flow
 - Hardware Flow mit Loop-Pipelining

LegUp

- Software Flow:
 - LegUp stellt Tiger-MIPS Prozessor zur Verfügung
 - wird für den FGPA synthetisiert
 - kompiliert C-Programm mit LLVM zu MIPS-Assembler
 - Ausführung auf dem Tiger-MIPS Prozessor

LegUp

- Processor/Accelerator Hybrid Flow

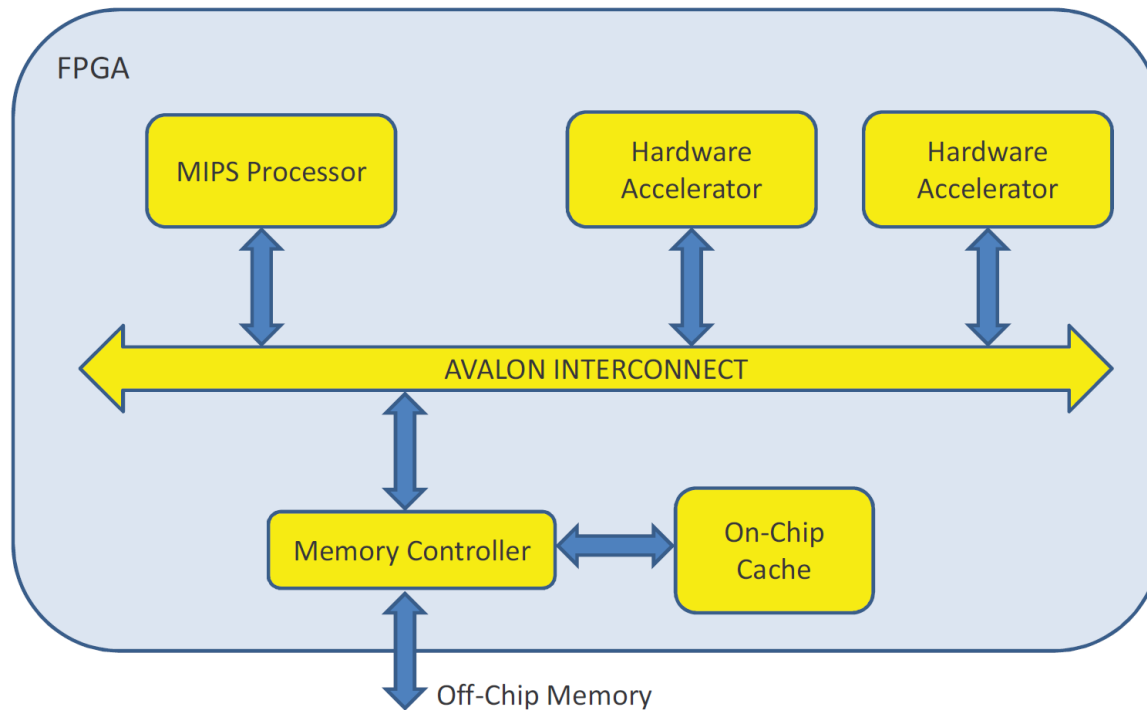


Abb.: Processor/Accelerator Architektur, vgl. [2]

LegUp

- Hardware Flow:
 - als Grundlage dient die Zwischendarstellung des LLVM
 - Basisblöcke bestehen aus „einfachen“ Methoden
 - diese können direkt in Hardware-Operationen überführt werden
 - bearbeitete Basisblöcke werden in eine FSM überführt
 - anschließend Zuweisung der Funktionseinheiten und Register
 - Variablen werden als eigene RAM-Blöcke synthetisiert

LegUp

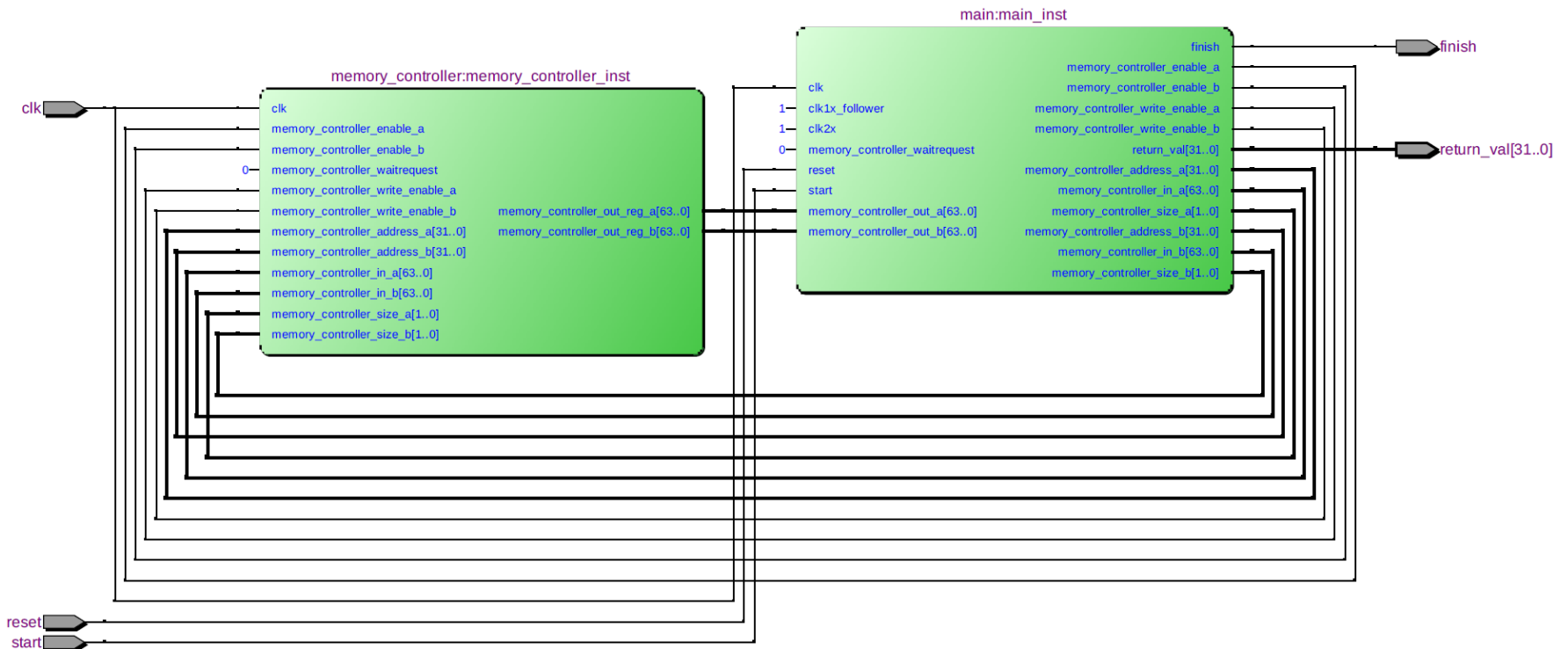


Abb.: generiertes LegUp-Module

Erweiterungen

- Anpassungen am C-Programm:
 - Parameterliste der Funktionen
 - Entwicklung einer main-Funktion
 - einfache Überprüfung der Kompression/Dekompression
- anschließende Synthese mit LegUp

Erweiterungen

- Aufgetretene Probleme:
 - keine Möglichkeit Daten in/aus der Schaltung zu laden
 - `return_val` nicht geeignet
 - Kommunikation über PCIe ebenfalls ungeeignet
 - 2. Instanz des Memory-Controllers erzeugt neue RAM-Blöcke

Erweiterungen

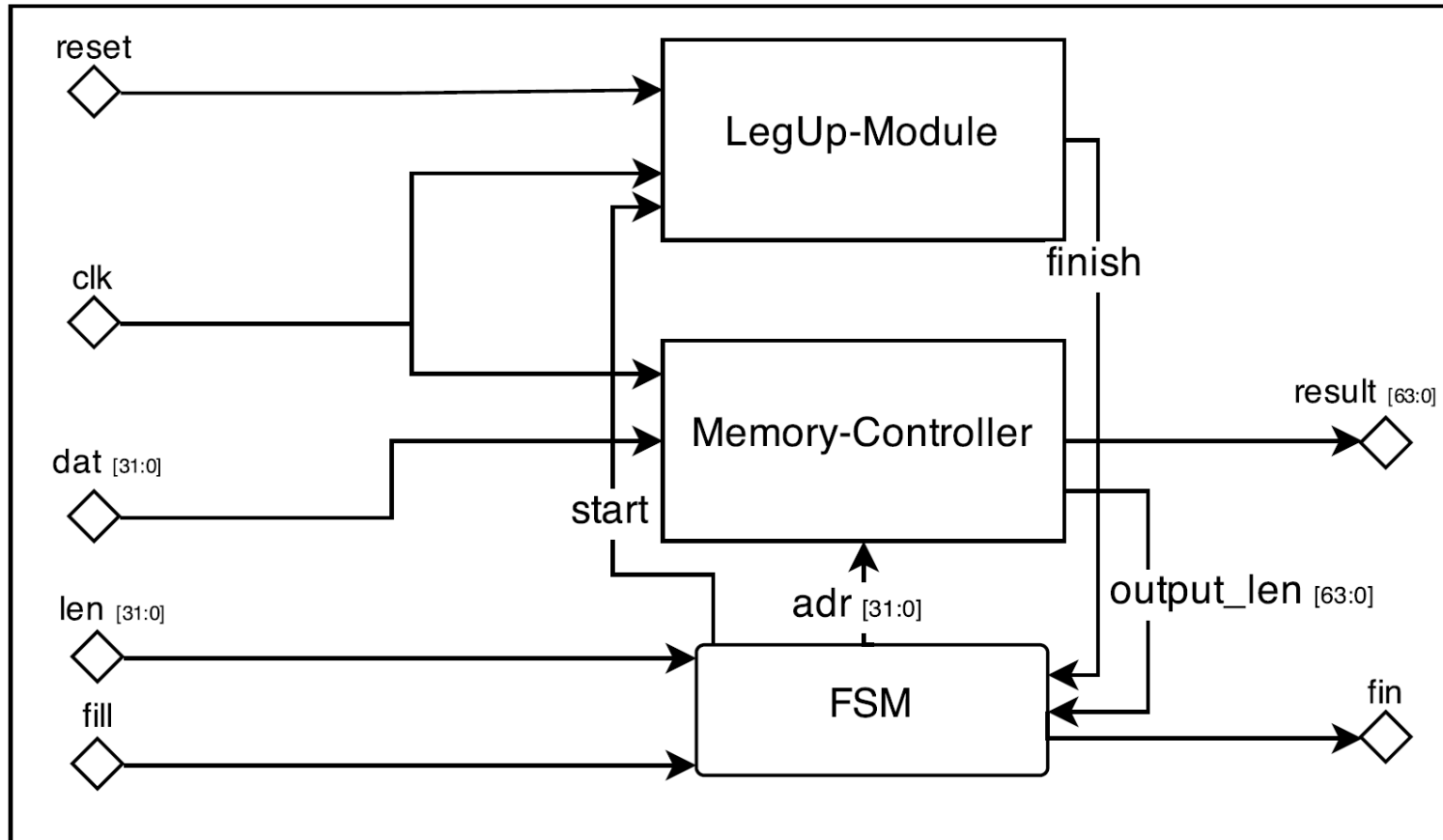


Abb.: ursprünglicher Wrapper-Entwurf

Erweiterungen

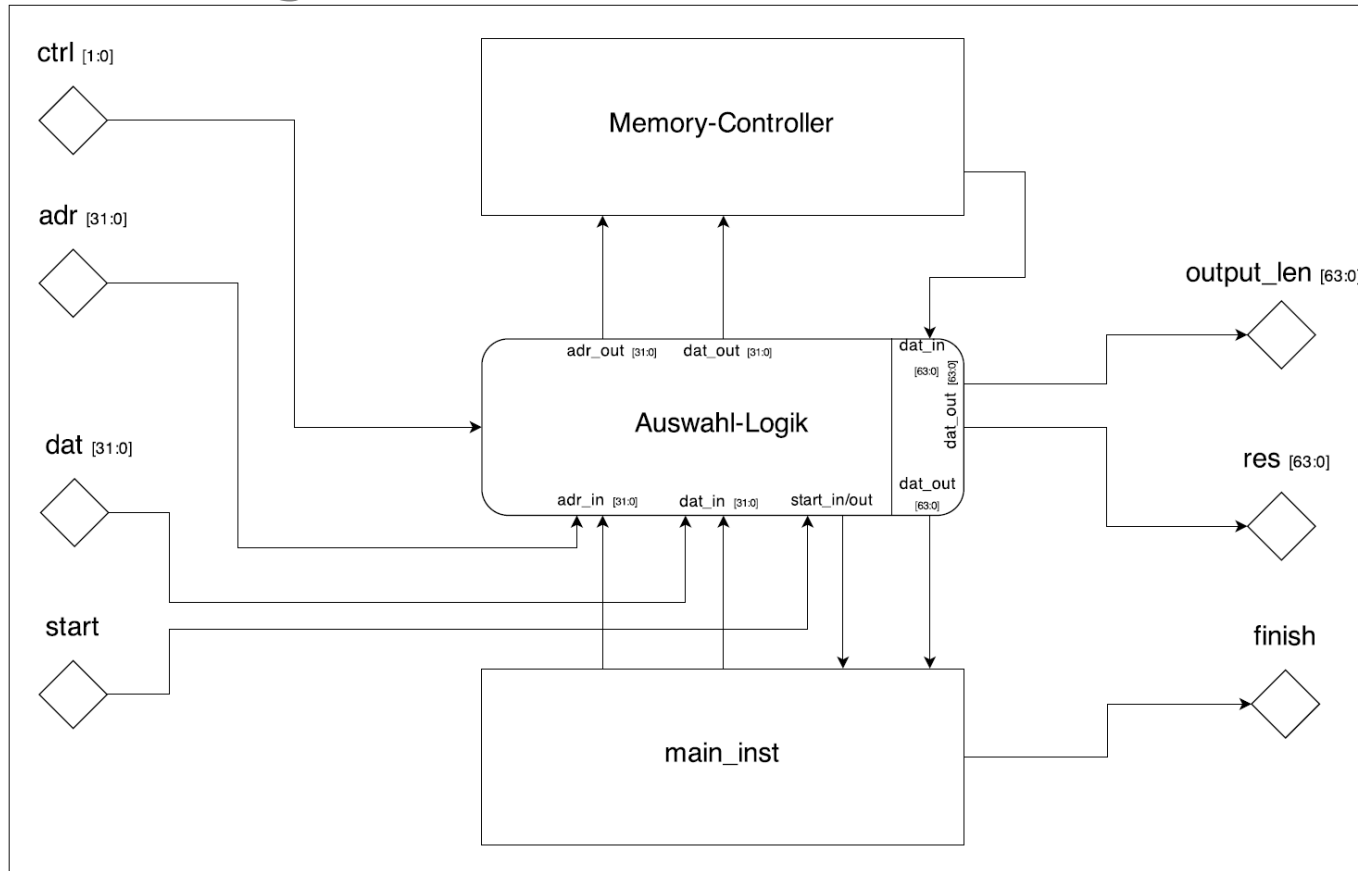


Abb.: überarbeitetes LegUp-Module

Erweiterungen

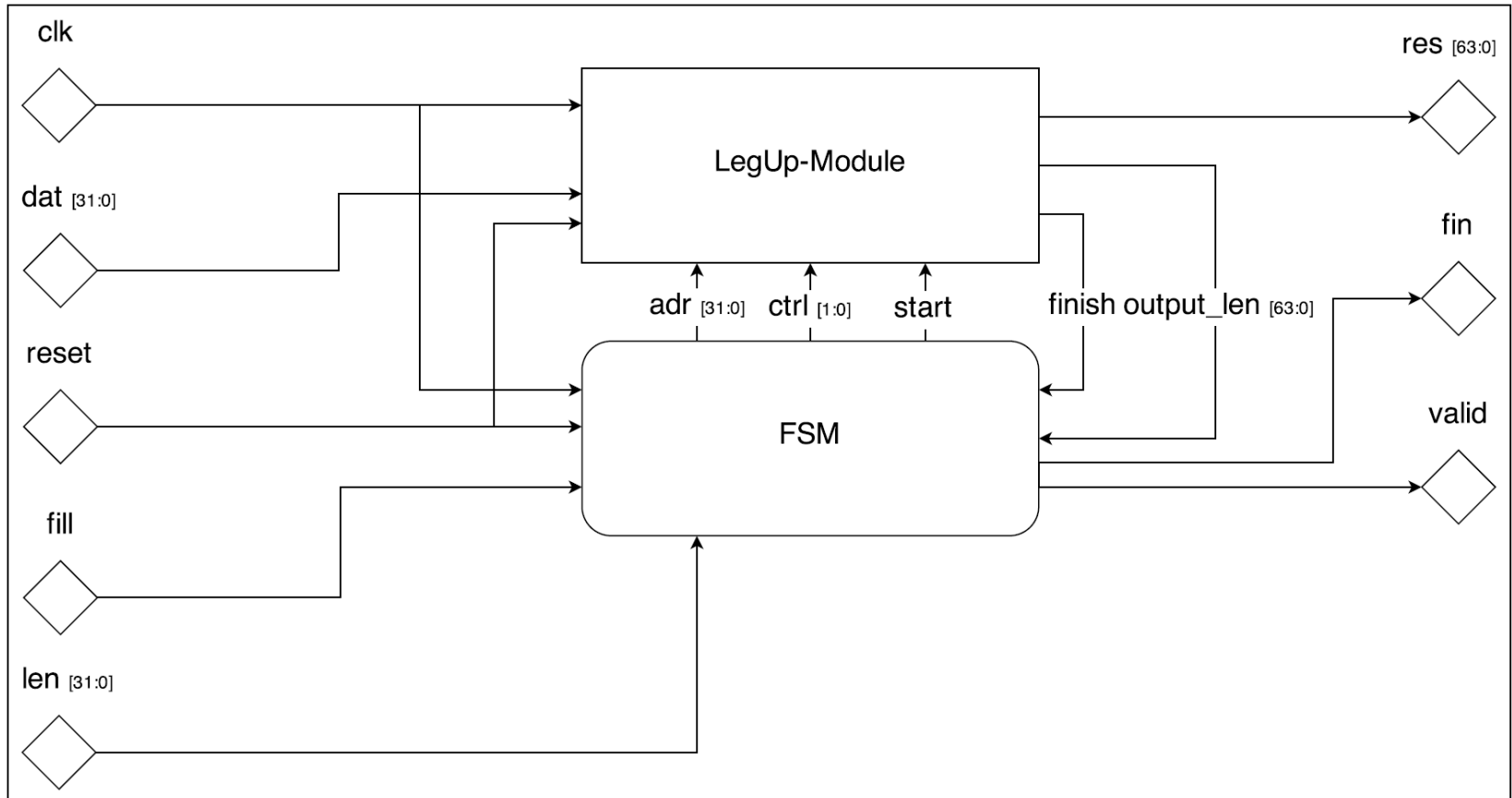
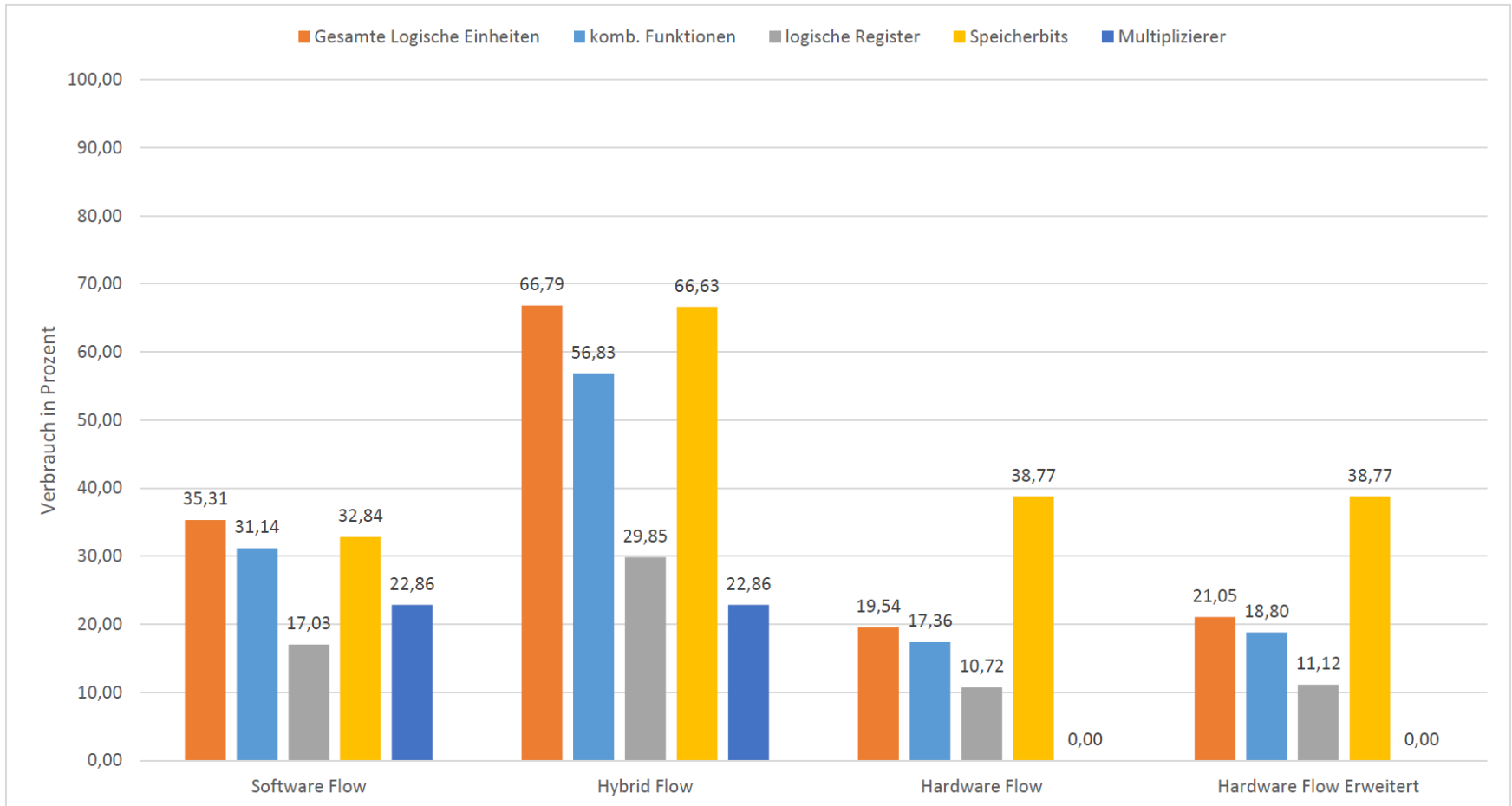


Abb.: finaler Wrapper-Entwurf

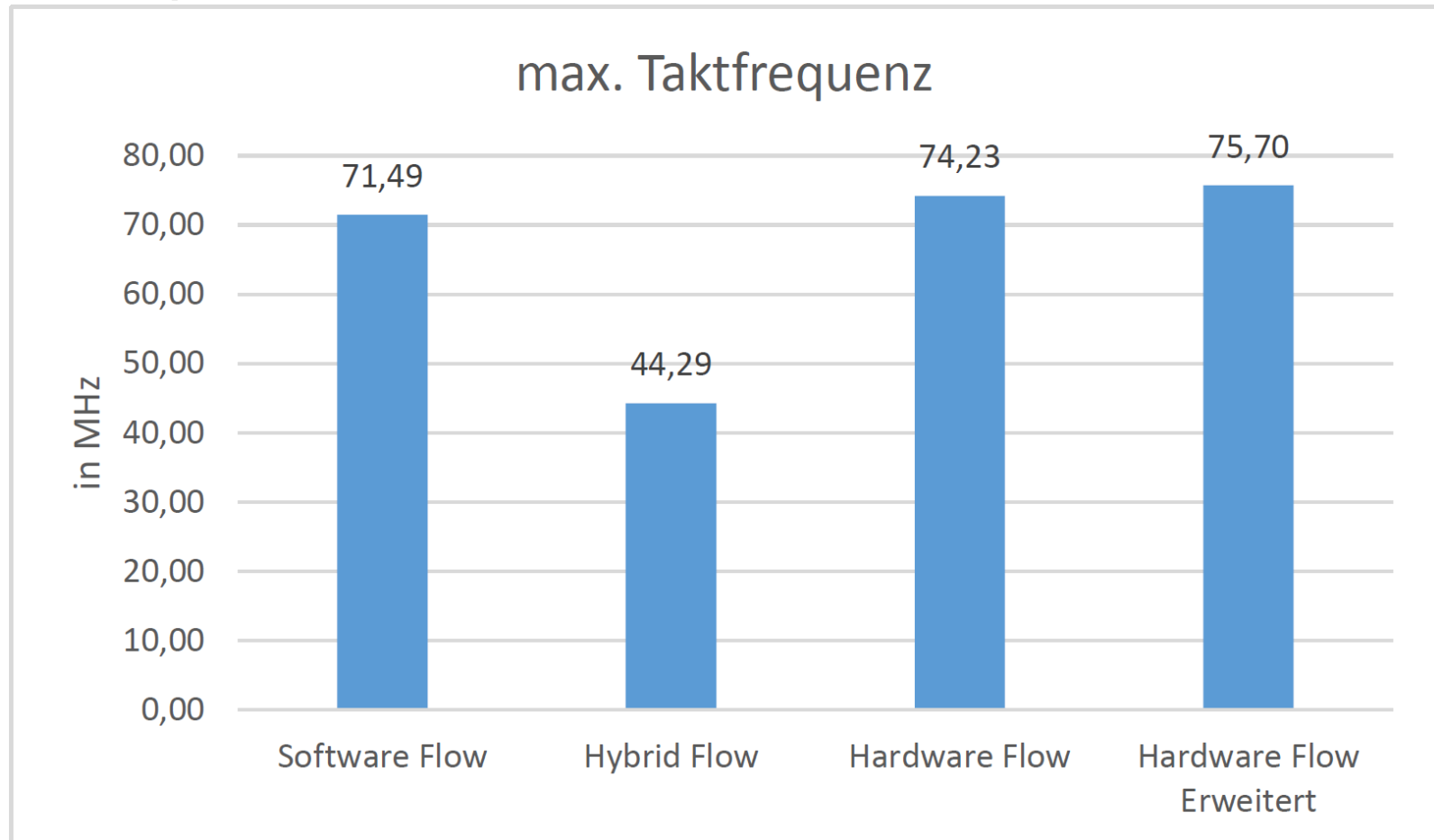
Analyse

- FPGA: Altera Cyclone II (EP2C35F672C6):
 - max. 33.216 logische Einheiten
 - max. 483.860 Speicherbits
- Ergebnisse aus „Compilation-Reports“
- nur Synthese durchgeführt
- keine Umsetzung auf konkreter Hardware

Analyse



Analyse



Analyse

Als marshallische Nachfragefunktion (auch walrasianische Nachfragefunktion), benannt nach dem Ökonomen Alfred Marshall (bzw. Léon Walras), bezeichnet man in der Mikroökonomik und dort speziell in der Haushaltstheorie eine mathematische Funktion, die für gegebene Güterpreise und ein gegebenes Vermögen angibt, welche Menge von jedem einzelnen Gut konsumiert werden sollte, wenn man den größtmöglichen Nutzen realisieren möchte. Ausgangspunkt der Überlegungen, die zur marshallischen Nachfragefunktion führen, ist das Prinzip der Nutzenmaximierung:

Abb.: Beispielttext, vgl. [7]

- originaler Text: 515 Zeichen
- Einlesevorgang: 517 Takte
- Kompressionsvorgang: 11.277 Takte
- Auslesevorgang: 454 Takte
- komprimierter Text: 451 Zeichen

Fazit

- mächtiges Framework
- es können schnell erste Ergebnisse erzielt werden
- sinnvolle Schaltung erst durch zusätzliche Arbeit
- Ziel noch nicht erreicht
- momentan auf Altera FPGA beschränkt
- generierter Verilog-Code schwer zu lesen/warten
- händischer Entwurf ist evtl. zu bevorzugen

Quellenverzeichnis

- [1] ALTERA CORPORATION: *Cyclone II Device Handbook, Volume 1*, 2008. <http://www.altera.com/literature/lit-cyc2.jsp>.
- [2] A. CANIS, J. CHOI, M. ALDHAM, V. ZHANG, A. KAMMOONA, J.H. ANDERSON, S. BROWN, T. CZAJKOWSKI: *LegUp: High-level synthesis for FPGA-based processor/accelerator systems*. ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), 33–36, February 2011.
- [3] A. CANIS, J. CHOI, J.H. ANDERSON, S. BROWN: *Lab 1: Using the LegUp High-level Synthesis Framework*. University of Toronto, July/August 2014. <http://legup.eecg.utoronto.ca/tutorials.php>.
- [4] D. SALOMON, G. MOTTA: *Handbook of Data Compression*. Springer Verlag, 2010.
- [5] UNIVERSITY OF TORONTO: *LegUp Documentation*, Release 3.0, January 2013.
- [6] WIKIBOOKS: *Datenkompression: Verlustfreie Verfahren: Wörterbuchbasierte Verfahren: LZ77 — Wikibooks, Die freie Bibliothek*, 2013. [Online; abgerufen am 17. Januar 2015].
- [7] WIKIPEDIA: *Marshallsche Nachfragefunktion — Wikipedia, Die freie Enzyklopädie*, 2015. [Online; Stand 6. Februar 2015].
- [8] WILLIAMS, ROSS N.: *An Extremely Fast Ziv-Lempel Data Compression Algorithm*. *Data Compression Conference 1991 (DCC'91)*, 8–11, April 1991.
- [9] WILLIAMS, ROSS N.: *lzw1*, 1991. source code: http://ross.net/compression/download/original/old_lzrw1.c.



»Wissen schafft Brücken.«