Better Than Worst-Case Computing

Oliver Klöckner

Technische Universität Chemnitz

19. August 2016

Introduction

Motivation

Butler W. Lampson – Hints for Computer System

"Handle normal and worst cases separatly as a rule, because the requirements for the two are quite different:

The normal case must be fast.

The worst case must make some progress."

http://research.microsoft.com/en-us/um/people/blampson/33-Hints/WebPage.html

Inhalt

Was ist das Worst-Case Design?

Wieso wird es angewendet?

Was ist dann Better Than Worst-Case?

- Razor Logic
- Typical Case Optimization
- Approximation

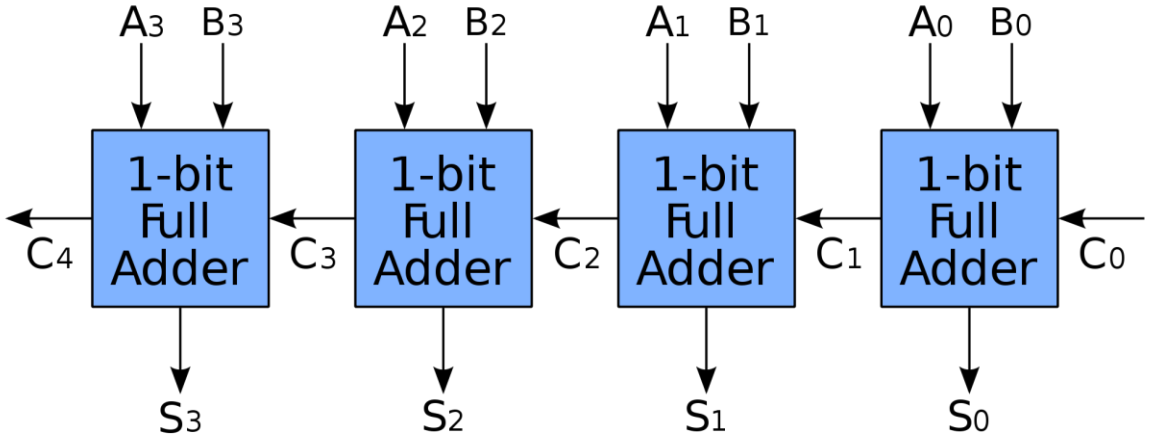
Herausforderungen

Worst-Case Design

Worst-Case Design

Background:

- digitale Systeme clock abhängig
- clock abhängig von Bauelementen durch Verzögerung



https://upload.wikimedia.org/wikipedia/commons/thumb/5/5d/4-bit_ripple_carry_adder.svg/2000px-4-bit_ripple_carry_adder.svg.png

Worst Case Spezifikationen:

- ermitteln/setzen von Rahmenbedingungen(Worst Case)
- Anpassung an Temperatur, Höhe, Spannung, Herstellung, …

Worst-Case Design

Folgen:

- garantierte Funktion unter verschiedenen Bedingungen
- Verzögerung notwendig, da für Worst Case gewartet werden muss
- langsameres System, hoher Energieverbrauch

Worst-Case Design

Sinnvoller Einsatz:

- immer kleinere Schaltkreise, immer komplexere Entwicklung
- inhomogene Herstellung
- unbekannter Verwendungszweck
- 100% sichere Systeme notwendig

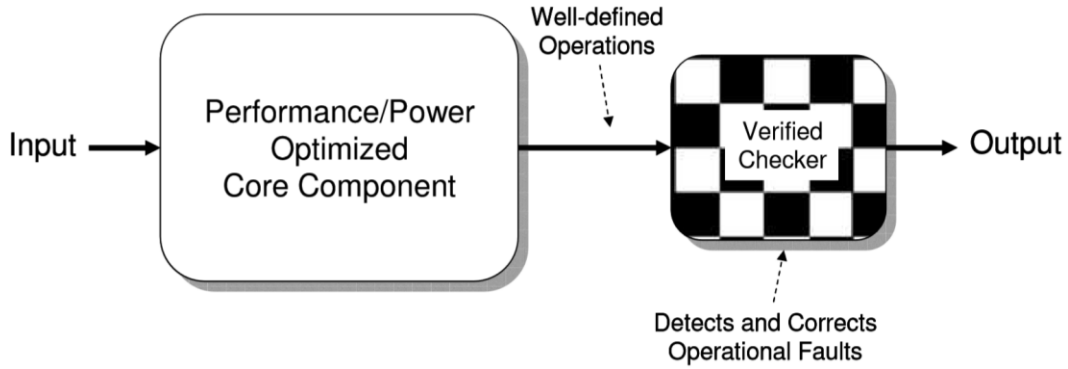
Abweichen erlaubt bei:

- bekannten, typischen Bedingungen,
  z.B. in Heimanwendungen, Server
- Ungenauigkeit erlaubt,
  z.B. bei Videoanwendungen, Grafikkarten

Better Than Worst-Case Design

Better Than Worst-Case Design

Design:



Todd Austin et al.:
Opportunities and Challenges for
Better Than Worst-Case Design

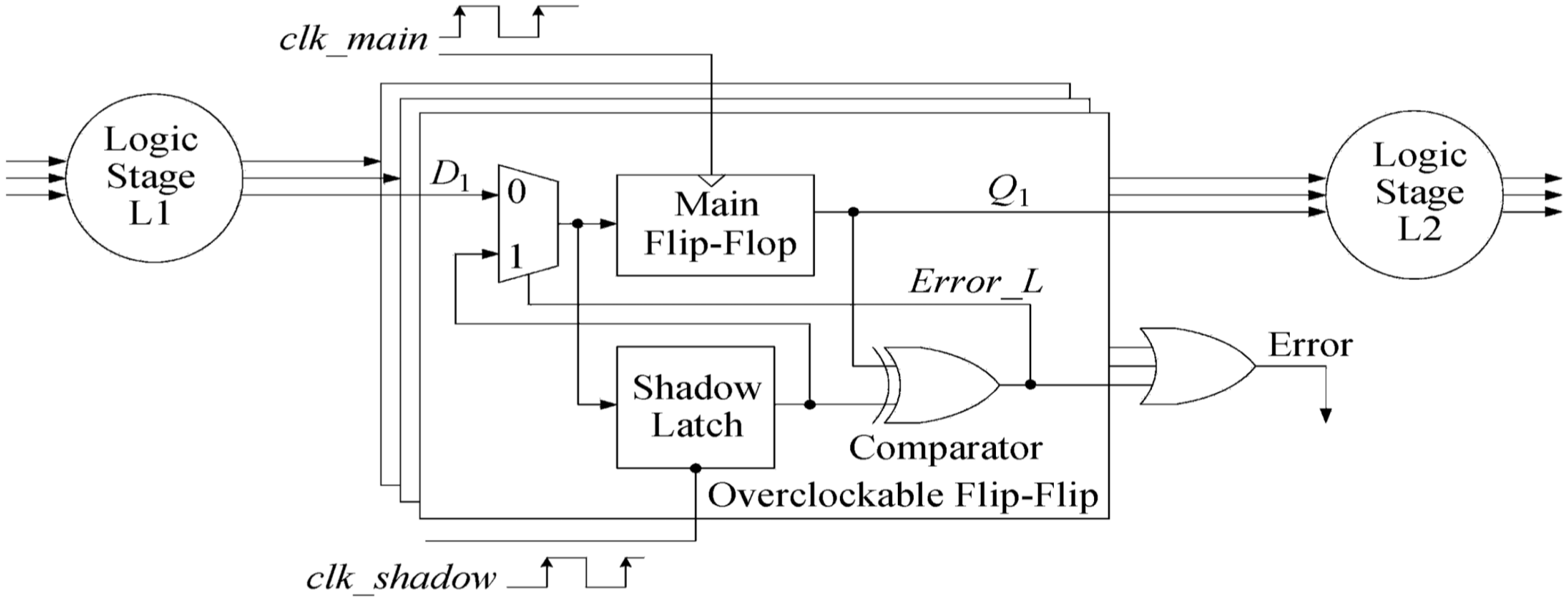
- Umfasst verschiedene Techniken
- Entkopplung von Komponente und Überprüfung
- Anforderungen an Überprüfung:

  Einfach, Schnell, Korrekt
- Ziel:

  höhere Frequenz, niedrigere Spannung

Dynamic Voltage Scaling – Razor Logic

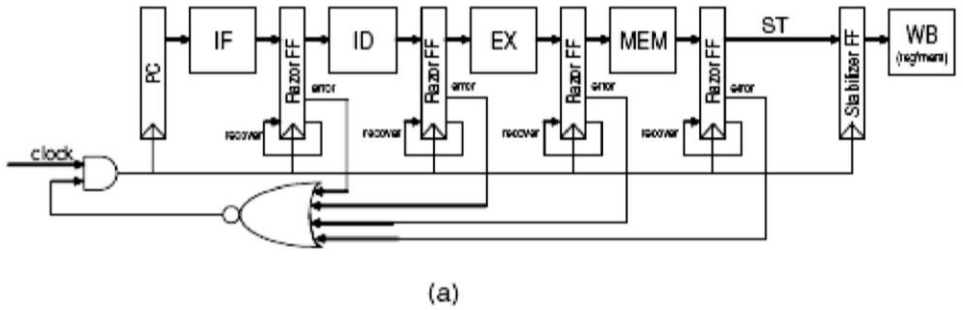
Ziel: Anpassung der Spannung/Frequenz bei geringer Auslastung mit automatischer Fehlererkennung – Energieersparnis



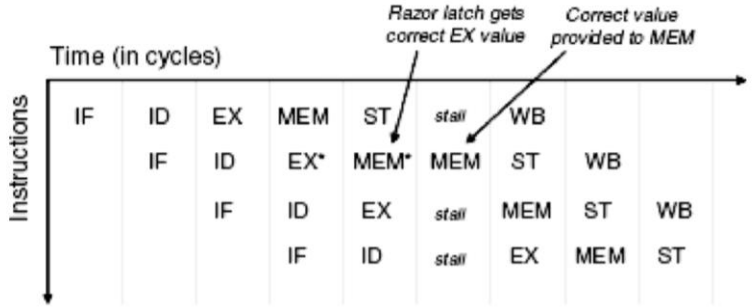
Jason Cong et al. : Better-Than-Worst-Case Design: Progress and Opportunities

Dynamic Voltage Scaling – Razor Logic

Pipeline Stages: Clock Gating

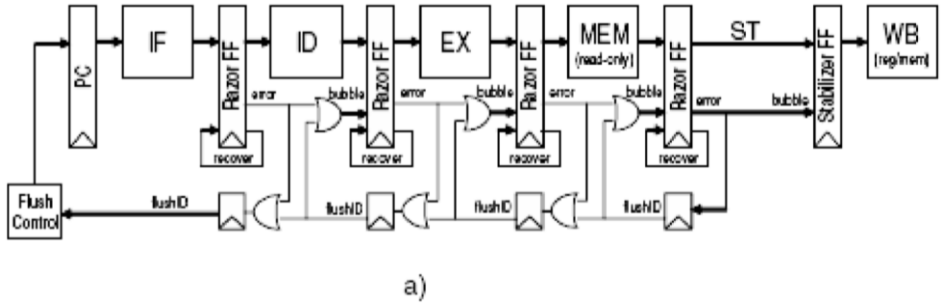


(a)

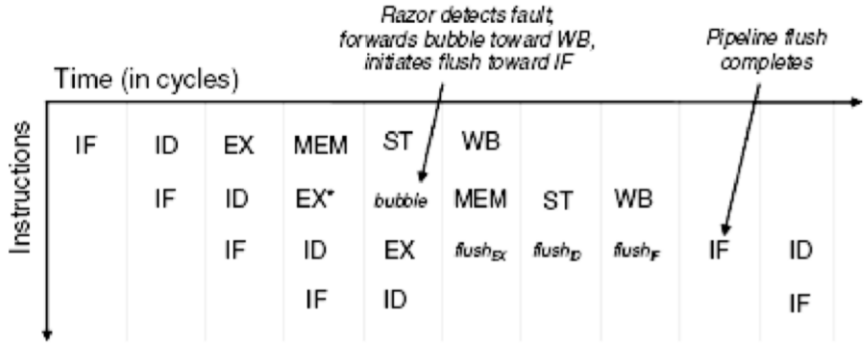


(b)

Counterflow



a)

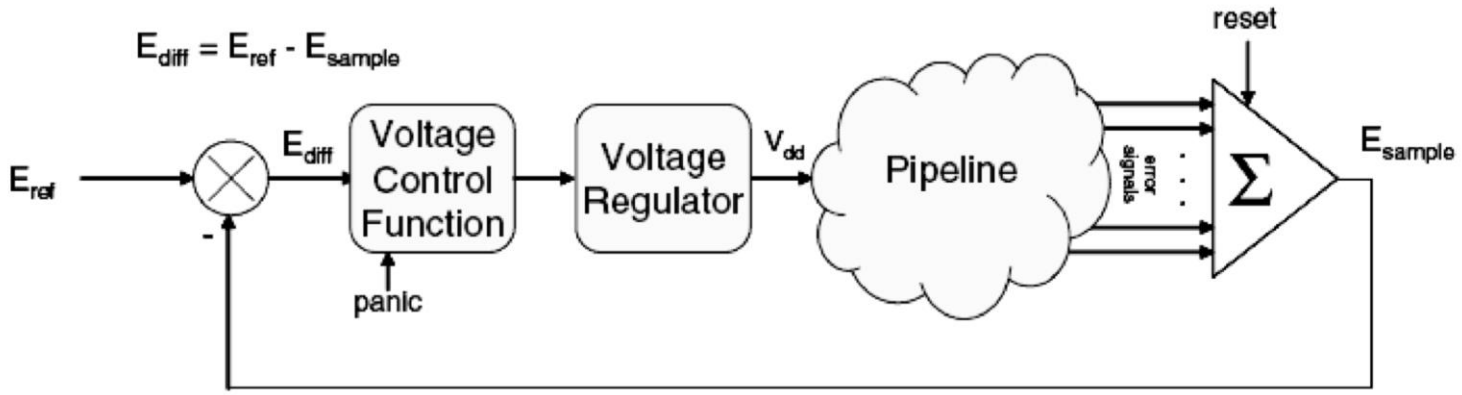


b)

Stefanos Kaxiras, Margaret Martonosi: Computer Architecture Techniques for Power-Efficiency

Dynamic Voltage Scaling – Razor Logic

Voltage scaling:



Stefanos Kaxiras, Margaret Martonosi: Computer Architecture Techniques for Power-Efficiency

Vorteil:

- modifizierte ARM Kerne mit 64% Energieeinsparung bei 3% Overhead
- Anpassung der Spannung an Anwendungsfall

Typical Case Optimization

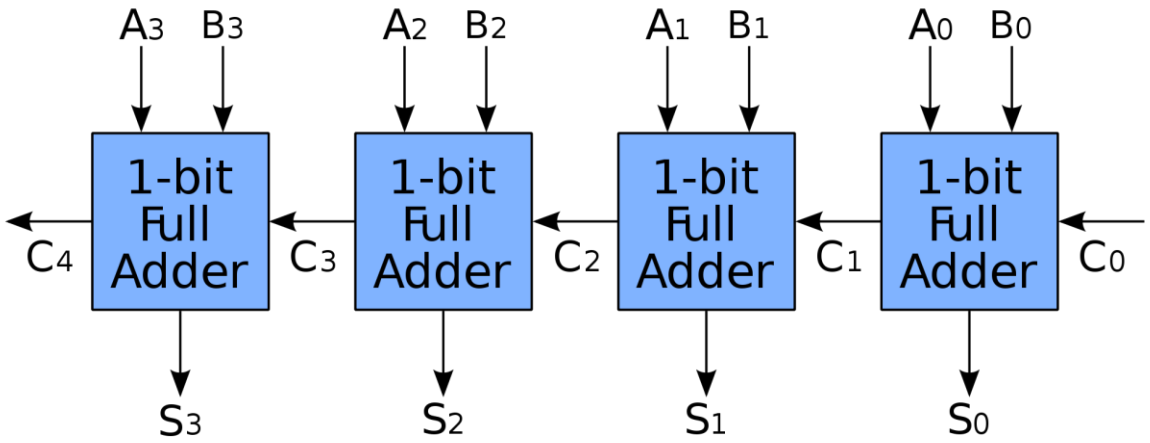
Ziel: Anpassen der Schaltkreise an den typical case, nicht worst case

Erfolgt durch:

- Ermitteln der Dauer unter Programm spezifischen Bedingungen
- Vergleich mit allgemeinen (zufälligen) Fall
- Optimierung durchführen

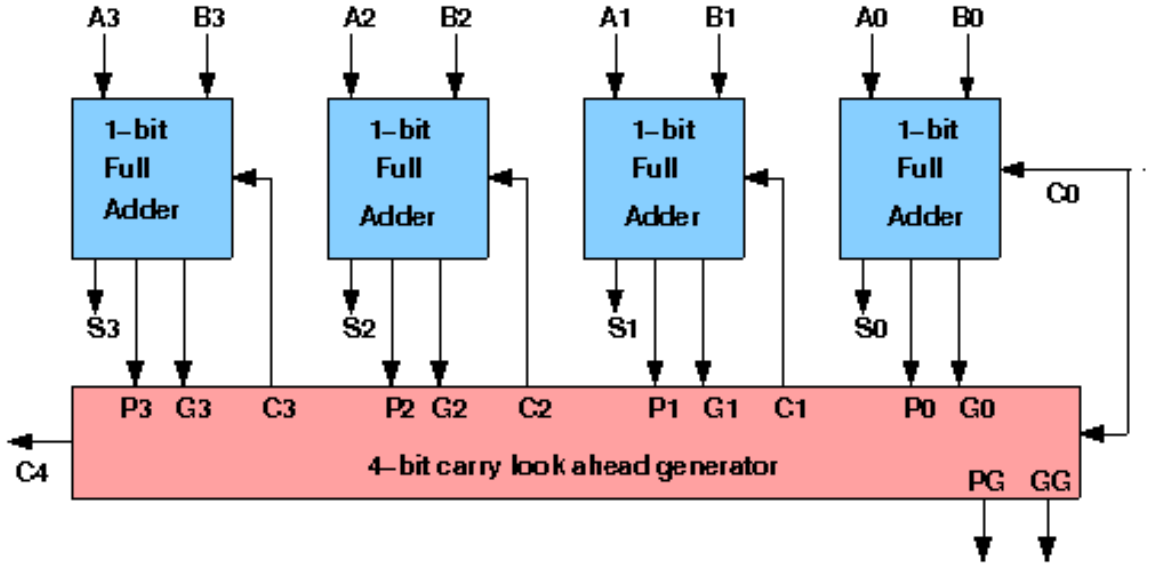
Typical Case Optimization - Adder

Ripple carry Addierer



https://upload.wikimedia.org/wikipedia/commons/thumb/5/5d/4-bit_ripple_carry_adder.svg/2000px-4-bit_ripple_carry_adder.svg.png

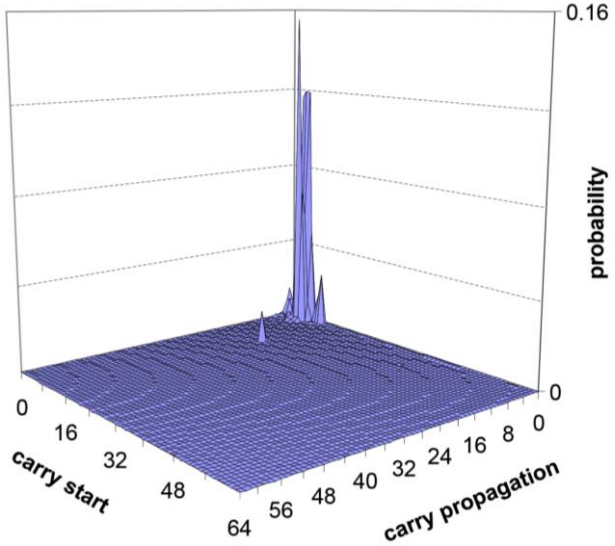
Carry Look Ahead Addierer



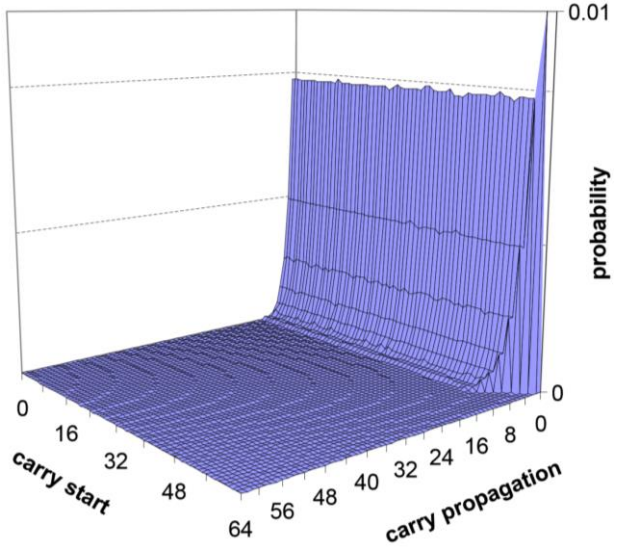
http://sit.iitkgp.ernet.in/~coavl/images/carrylookahead.png

Typical Case Optimization - Adder

Auswertung



Zufallswerte



Vergleich

Adder	**Latency** (in gate delays)		
Topology	Worst-Case	Typical-Case	Random
Kogge-Stone	8	5.08	7.09
TCO Adder	128	3.03	3.69

Todd Austin et al.: Opportunities and Challenges for Better Than Worst-Case Design

Folgen:

- schlechte Performance im worst case
- schneller in typical und random case

Approximate Computing Designs

Ziel: Performance Erhöhung durch Fehlerrate >0% (meist angegeben)

Fehler Verteilung:

- Ripple-Carry Adder: mäßige Fehlerzunahme unter kritischer Spannung
- Kogge-Stone Adder: niedrigere kritische Spannung, schnelle Fehlerzunahme

- während Laufzeit: für niedrige Fehlerrate KSA; sonst RCA
- z.B. in Kernel von H.264 Codec mit Einsparung von 20%-60% gegenüber statischem Adder

Approximate Computing Designs

Reduktion weniger aktiver Teile:

- Reduzierung um Schaltkreisteile mit wahrscheinlich geringer Aktivität
- Wahrscheinlichkeit mit Hilfe von Simulationen oder math. Modellen
- Berechnung der Fehlerrate; falls zu hoch: Rückgängig, sonst weiter

Konfigurierbarer Adder:

- dynamische Anpassung der Fehlerrate
- Vorhersagen von Carry-In für MSB aufgrund von Carry-Out vorangegangener Bits
- geringer Einfluss weit entfernter Bits
- z.B. bei JPEG Komprimierung 21% höherer Durchsatz

Worst-Case Design

Für Heimanwender

Übertaktung, Unterspannung möglich,

da nicht Worst-Case Bedingungen vorherrschen!

Aber keine Gewährleistung!

Conclusion - Herausforderungen

- Direkt bei Entwicklung Typical-Case beachten, nicht immer nur Worst-Case
- Energieeinsparungen auf Hardware Ebene
- Spezielle Analysetools
- Energieeinsparungen auf Hardware Ebene
- Dynamische Anpassung an Laufzeit
- Außerhalb kritischer Bereiche Approximierungen zulassen; Anpassen von Algorithmen
- General-Purpose Prozessoren schwierig zu approximieren

Fragen?

Ohne Anpassung geht es nicht schneller - Danke für eure Aufmerksamkeit?!