TECHNISCHE UNIVERSITÄT DRESDEN

**Faculty of Computer Science** Institute of Computer Engineering, VLSI Design Systems, Diagnostics and Architecture

# Enhancing FPGA Device Capabilities by the Automatic Logic Mapping to Additive Carry Chains

***Thomas B. Preußer***

Milano, FPL 2010

# Itinerary

- Motivation & Goals
- Background on Mapping and Carry Chains
- Mappability Criterion
- Evaluation on MCNC Benchmarks

**The Way Up!**

# Motivation

**Carry chains (CC) may provide**:

- a fast direct signal link between logic blocks, and
- an additional input to the logic block.

→ For addition: Vitoroulis & Al-Khalili, NEWCAS'07.

**Their utilization beyond addition**:

- accelerates critical paths, and
- increases logic density but
- is functionally restricted (Hauck et al., FPGA'98: no "inverting propagate").

# The Idea: Carry Chain – Running Board

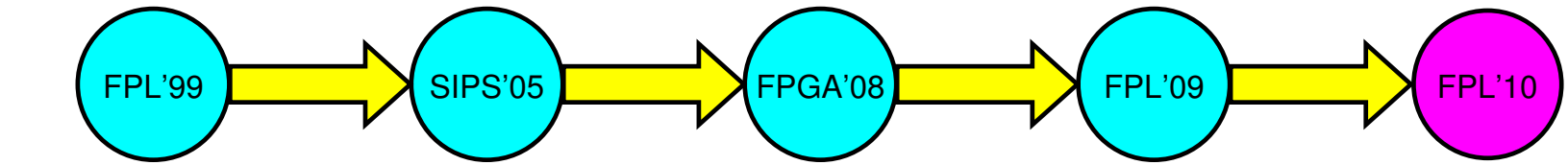
Special-purpose structure *running board*:

- for boarding and
- for exiting.


. . . utilized:

- to increase the transport capacity and
- to speed up boarding and exiting.

# Previous Approaches




**Põldre & Tammemäe**
special–purpose mapping of 4–to–2 adders

**Preußer et al.**
special–purpose mapping of arbiters

**Frederick & Somani**
automatic but for LUTs, no additive chains

**Parandeh–Afshar et al.**
special–purpose mapping of short counters

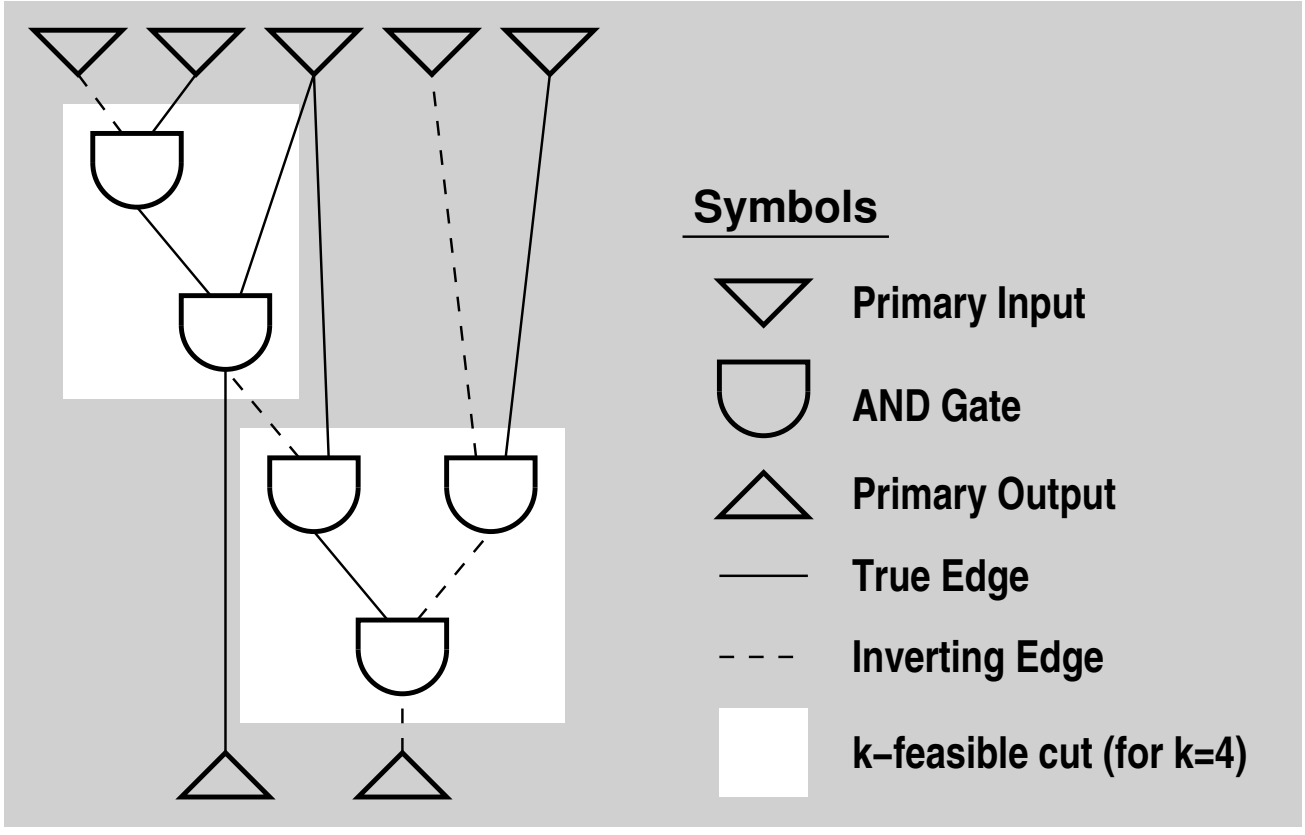
**Preußer & Spallek**
for additive chains, but manual

# Goals

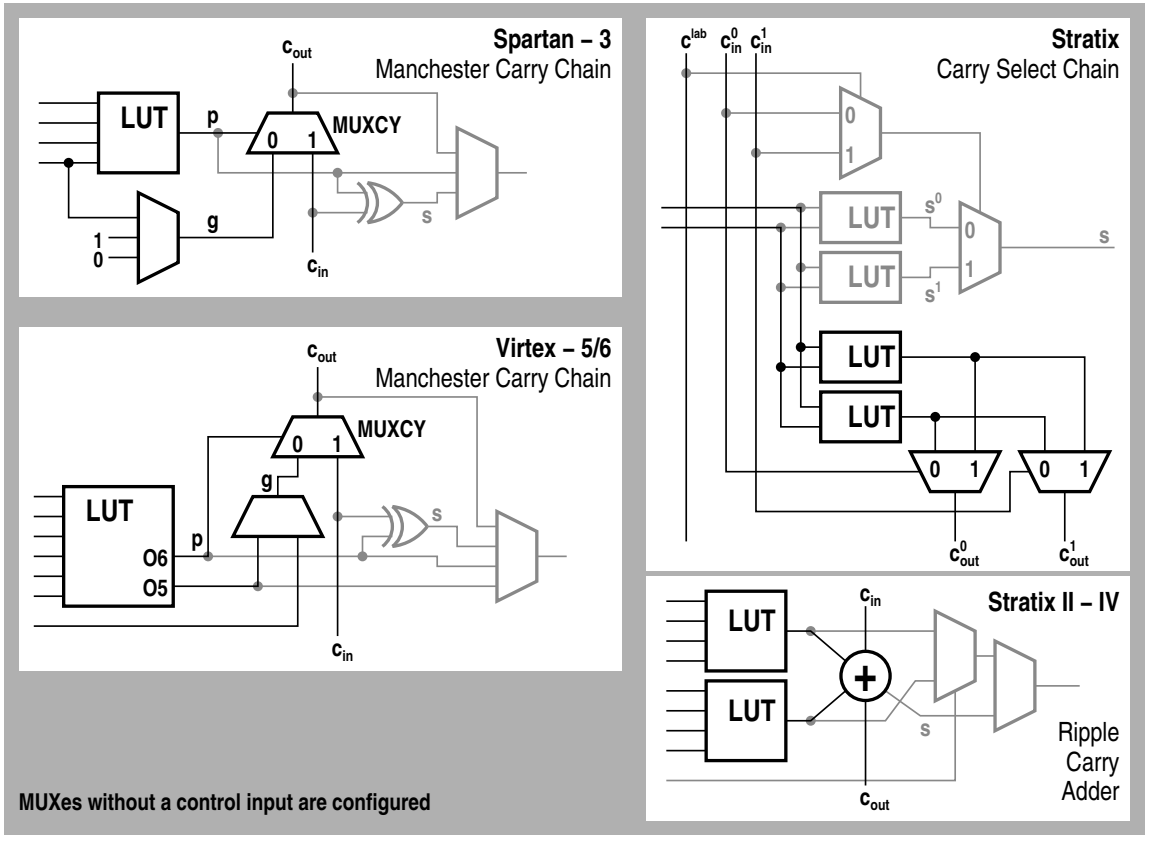
- Utilize carry chains for *general-purpose* logic.
- Provide criteria for the *automated* mapping to CC-structures found in silicon.
- Estimate benefits on a contemporary FPGA architecture.
- Propose architectural enhancements for increased CC-capabilities.

**Please Follow Me!**

# Cut-based Mapping of AND-Inverter-Graphs



# Example Carry-Chain Architectures



# Carry-Chain Capabilities

**Addition**!:

$$
\begin{aligned}
c_{\text{out}} &= ab + ac + bc \\
&= ab + (a \oplus b)c \\
&= g + pc \\
&= \overline{p}g + pc \qquad (1) \\
&= \overline{p}a + pc \qquad (2)
\end{aligned}
$$

$a$, $b$ operand bits at bit position
$c$ incoming carry input
$c_{\text{out}}$ produced carry output

$g = a \cdot b$
*generate* a carry definitely
$p = a \oplus b$
*propagate* the carry input

*c cannot* be inverted.
Each stage is, indeed, equivalent to a MUX ...

# Mappability Criterion

... turned the *wrong* way:

$$\begin{aligned} f(X) &= f(x_{n-1}, \ldots, x_i, \ldots, x_1, x_0) \\ &= \overline{x_i} \cdot f(x_{n-1}, \ldots, x_i = 0, \ldots, x_1, x_0) + \\ &\quad x_i \cdot f(x_{n-1}, \ldots, x_i = 1, \ldots, x_1, x_0) \\ &= \overline{x_i} \cdot f_0(X \setminus x_i) + x_i \cdot f_1(X \setminus x_i). \end{aligned}$$

Universal capabilities are ensured by a *controlling* input (SHANNON Expansion) but the carry-chain offers a *controlled* input.

$f$ is still mappable if and only if $f_0 \rightarrow f_1$ (see paper for the proof) with:

$$p = f_1 \overline{f_0}, \text{ and}$$
$$g = f_0 \left[+ f_1\right].$$

# Cut Mappability

Special cases that map easily even with constrained computation of $g$:

| Condition | Structure | Mapping | Operation |
|---|---|---|---|
| $f_0 = 0$ | $f = x_i \cdot f_1$ | $g = 0,\ p = f_1$ | AND |
| $f_1 = 1$ | $f = f_0 + x_i$ | $g = 1,\ p = \overline{f_0}$ | OR |
| $\exists x_j.\overline{p} \rightarrow (f_0 \equiv x_j)$ | $f = \overline{p}x_j + px_i$ | $g = x_j,\ p = f_1\overline{f_0}$ | RCA |

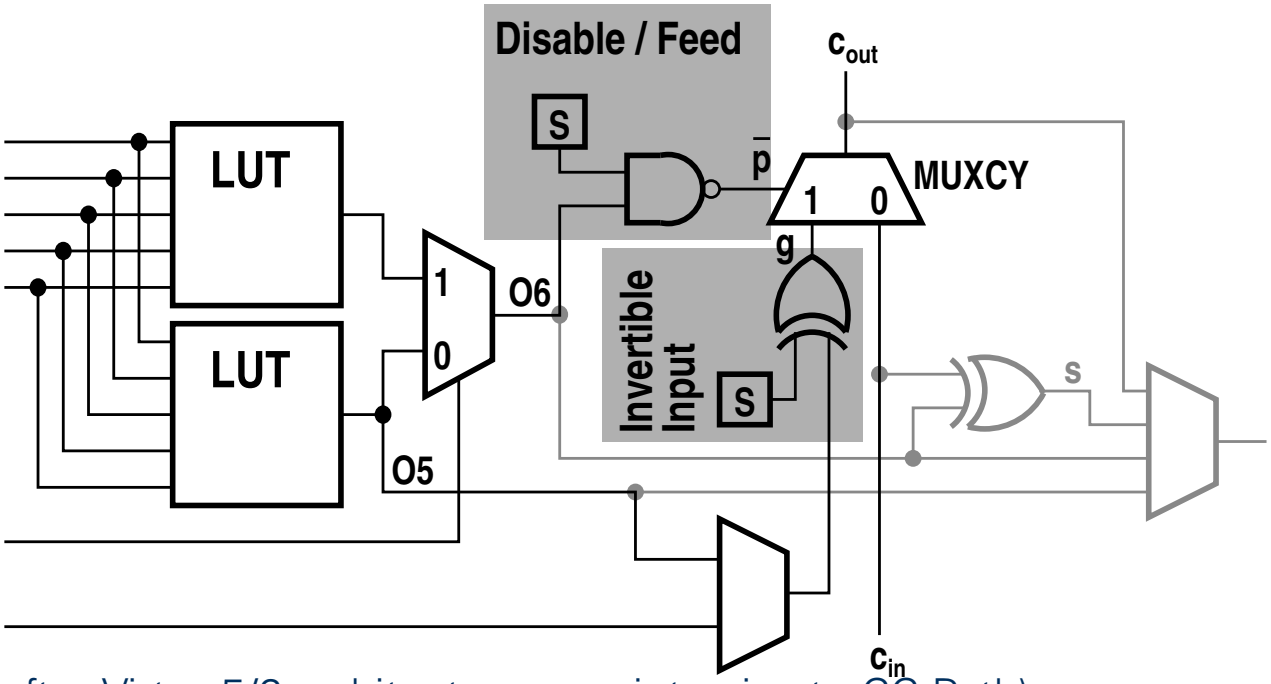
... are frequent (combinational MCNC benchmarks after cut slicing by ABC):

| 4-feasible Cuts: | |
|---|---|
| OR | 32% |
| RCA | 31% |
| AND | 52% |
| $\sum$ special | 85% |
| generic | 13% |
| unmappable | 2% |

| 5-feasible Cuts: | |
|---|---|
| OR | 33% |
| RCA | 23% |
| AND | 49% |
| $\sum$ special | 84% |
| generic | 14% |
| unmappable | 2% |

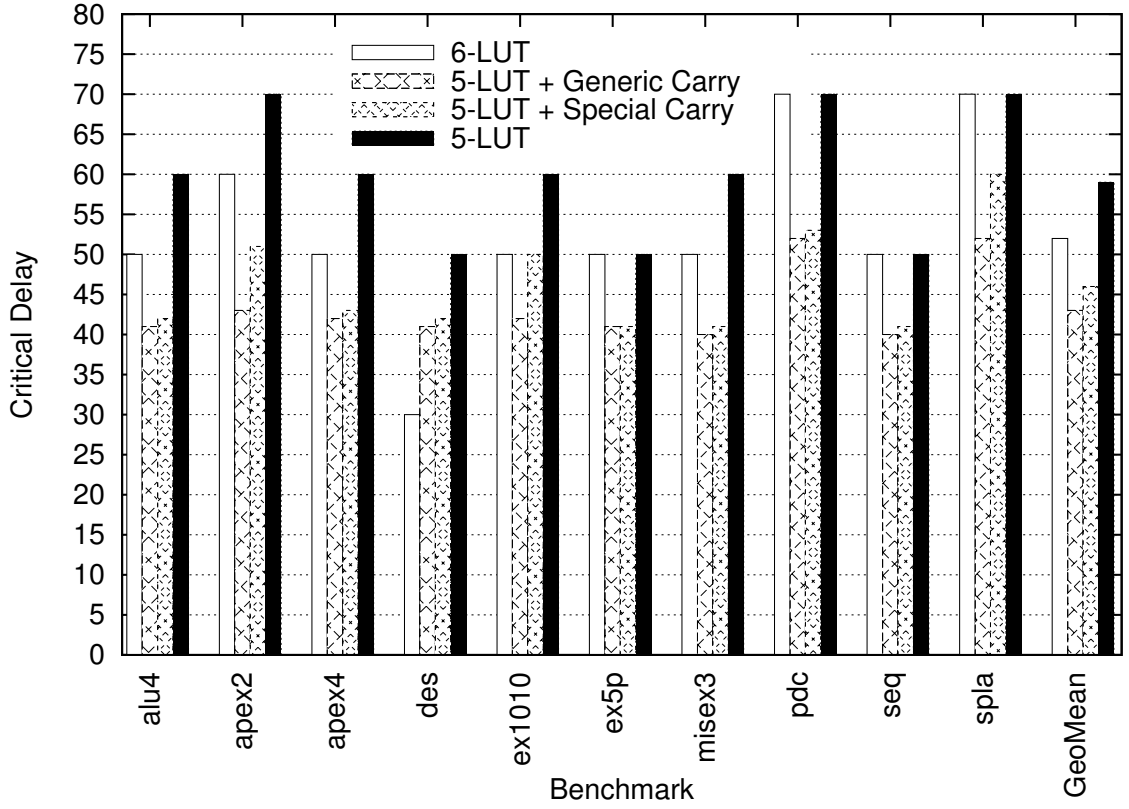
# Improving Carry Chain Accessibility

Allows a majority of the $(k+1)$-input cut functions to be implemented in one block (AND / OR / RCA decompositions):



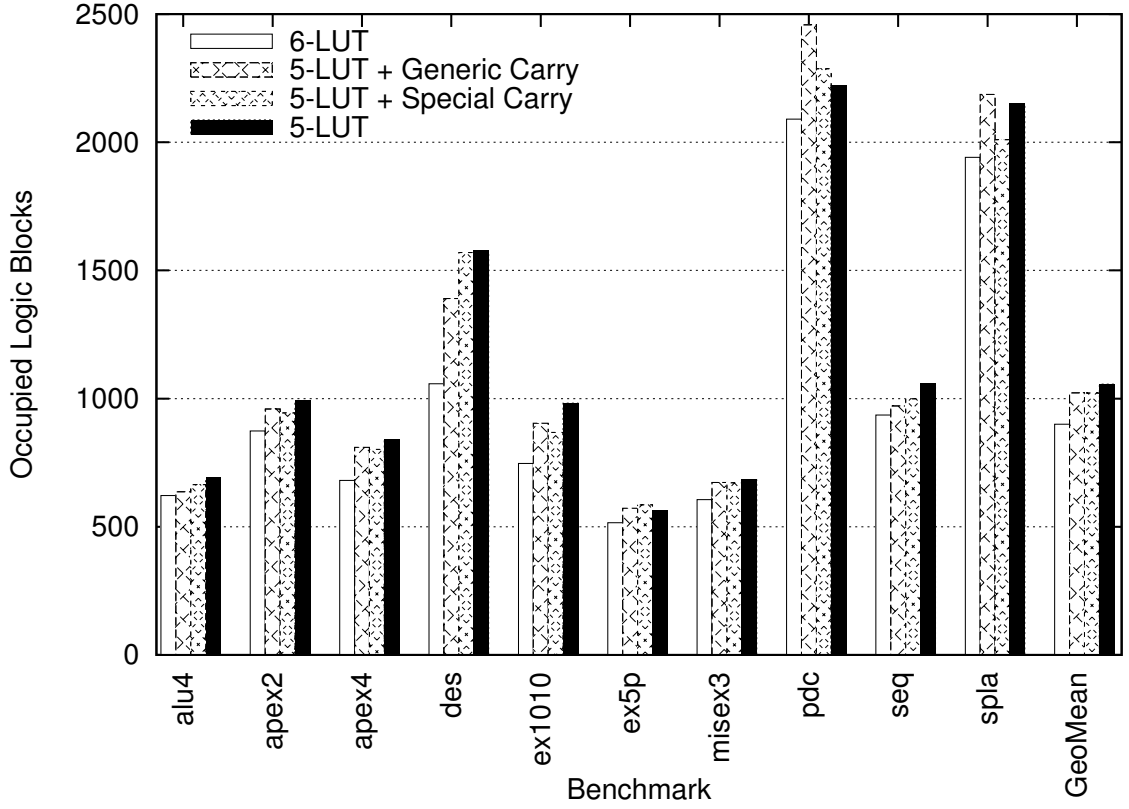

(taylored after Virtex-5/6 architecture, non-intrusive to CC-Path)

# Critical Delay under Carry-Chain Mapping




Optimization Goal: Delay

# Block Count under Carry-Chain Mapping




## Optimization Goal: Delay

# Contributions

- Formal criterion for carry-chain mappability of cut functions.
- Identification of function decomposition classes that map easily.
- Statistical analysis of MCNC benchmarks to support:
  - the mappability of a majority of cut functions, and
  - the expected benefit in delay and block count.
- Suggestion of a powerful logic block enhancement based on an improved access to the carry-chain structure.

# Future Work

- Pave the way into the real world!
  - Enhance mapping tools.
  - Enhance architectures.
- Explore further mapping capabilities:
  - Allow area-speed trade-offs as by logic duplication.
  - Review the balancing (or even de-balancing?) of large logic cones in awareness of the carry paths.

# Thank you!

Cable Car Photos:

- http://en.sevenload.com/photos/11ILiCP-San-Francisco-Cable-Car
- http://en.wikipedia.org/wiki/File:Sf_cable_car.jpg
- http://en.wikipedia.org/wiki/File:SanFrancablecar.JPG