



CprE / ComS 583 Reconfigurable Computing

Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

Lecture #5 – FPGA Arithmetic



Quick Points

- HW #1 due Thursday at 12:00pm
 - Any comments?

September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.2



Recap

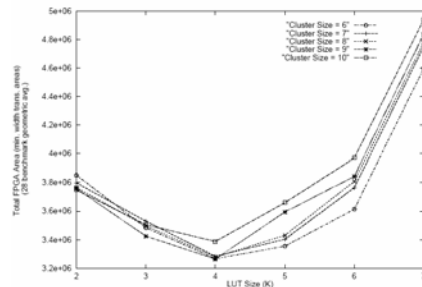


Fig. 8. Total Area for Clusters of Size 6 to 10

- Cluster size of $N = [6-8]$ is good, $K = [4-5]$

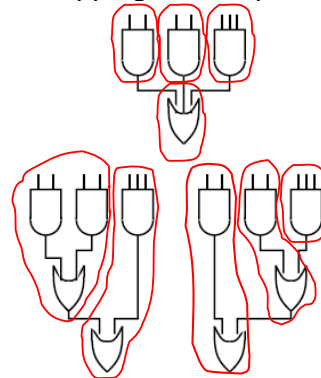
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.3



LUT Mapping Techniques



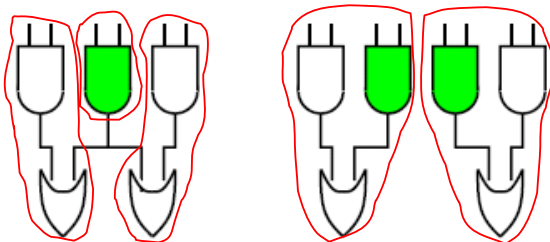
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.4



LUT Mapping Techniques (cont.)



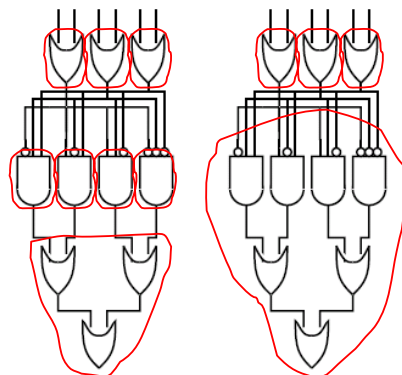
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.5



LUT Mapping Techniques (cont.)



September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.6

Outline

- Recap
- Motivation
- Carry / Cascade Logic
- Addition
 - Ripple Carry
 - Carry Bypass
 - Carry Select
 - Carry Lookahead
- Basic Multipliers

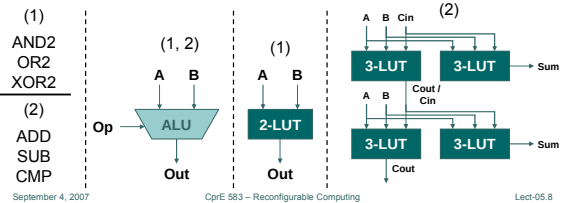
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.7

Motivation

- Traditional microprocessors, DSPs, etc. don't use LUTs
- Instead use a w -bit Arithmetic and Logic Unit (ALU)
 - Carry connections are hard-wired
 - No switches, no stubs, short wires



September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.8

Adder Delays

- Assuming a ripple-carry adder:
 - 32-bit ALU delay – 6ns
 - 32-cascaded 4-LUTs – $32 \times 2.5\text{ns} = 80\text{ns}$

4-LUT delay

Compare: 32-bit ALU (0.6λ)

2.0 ns	Logic delay	16 ns	Area optimized
0.5 ns	Single channel delay	6 ns	Delay optimized
2.5 ns	per bit		

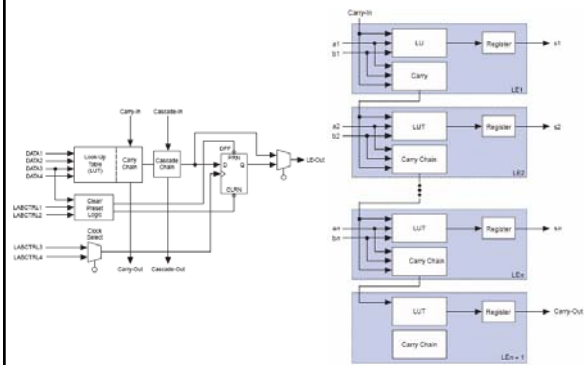
- Motivates extra hardware to accelerate carry operations

September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.9

Altera Flex 8000 Carry Chain

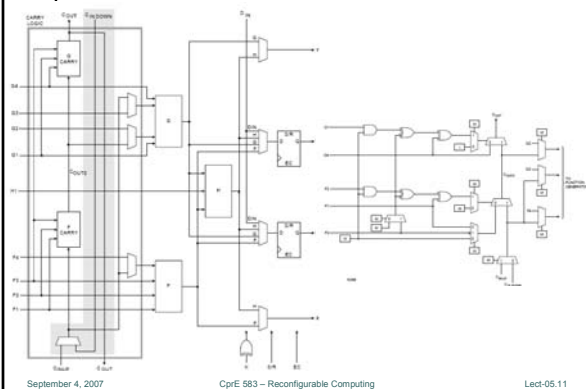


September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.10

Xilinx XC4000 Carry Chain



September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.11

Cascades

- Large fanin operations (reductions):
 - Decoding
 - Matching
 - Completion detection
 - Many-to-one reductions
- Combining logic is simple
- Makes use of dedicated paths

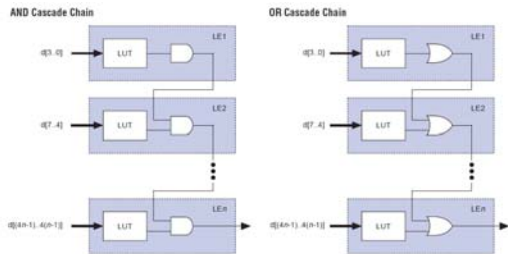
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.12

Altera Cascade Logic

- LE delay – 2.4 ns
- Cascade delay – 0.6 ns



September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.13

Why Look at Arithmetic?

- Parallelization
- Specialization
 - Architecture
 - Size
 - Inputs
- Adder problem – delay grows linearly with bit width
- Solutions for larger adders:
 - Pipelining
 - Carry bypass
 - Carry select

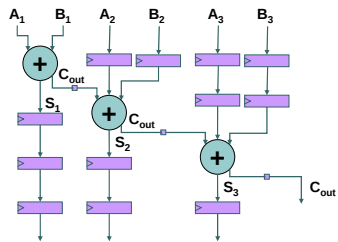
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.14

Adder Pipelining

- Not as practical in ASIC world (registers are expensive)
- Registers essentially “free” in FPGA logic blocks



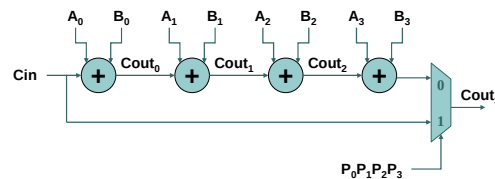
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.15

Carry Bypass Adders

- If all the propagates are 1 ($P_0P_1P_2P_3 = 1$) then $C_{out_3} = C_{in}$
 - $P_i = A_i \text{ xor } B_i$
 - Skip all the carry logic
 - Inexpensive



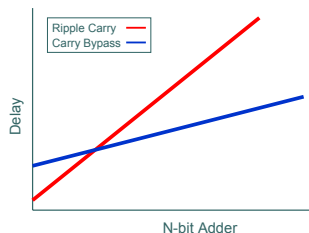
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.16

Carry Bypass Performance

- Small hardware cost:
 - 16-bit add – 4 CLB overhead
 - 32-bit add – 9 CLB overhead
- Delay growth still linear, smaller slope



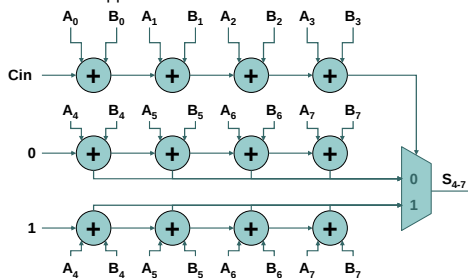
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.17

Carry Select Adders

- Precompute addition value for ($C_{in} = 0$) case and ($C_{in} = 1$) case
- Use mux to select between two with actual C_{in} value
- Cost of this approach?



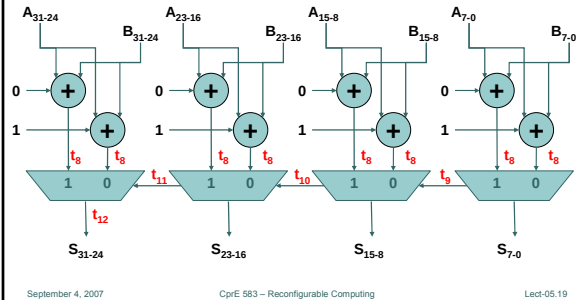
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.18

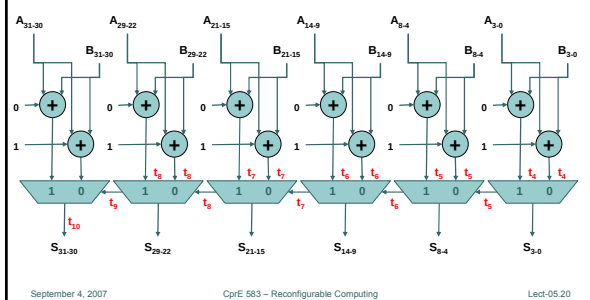
Linear Carry-Select

- Adder delay = w , mux delay = 1



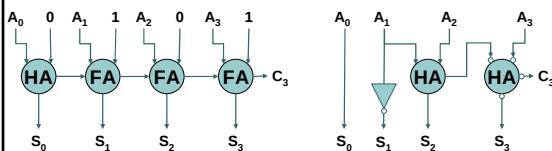
Square-Root Carry Select

- Each carry arrives when the corresponding sum is ready



Constant Addition

- If one operand is constant:
 - More speed?
 - Less hardware?



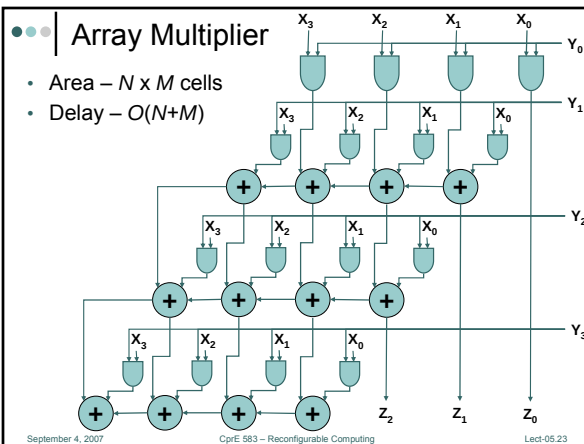
Multiplication

- Shift and add operations
- Need N bit adder, M cycles

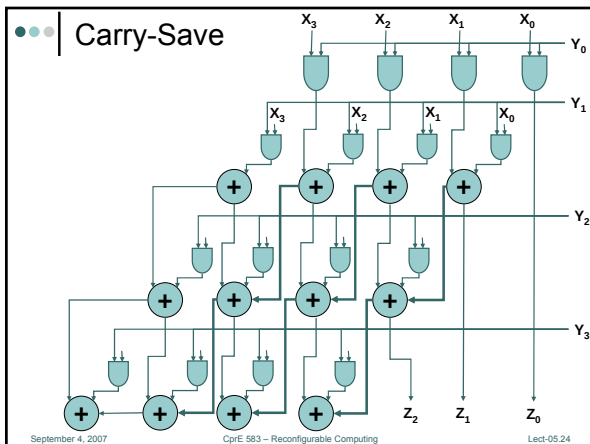
$$\begin{array}{r}
 42 = 101010 \text{ Multiplicand (N bits)} \\
 \times 10 = 1010 \text{ Multiplier (M bits)} \\
 \hline
 000000 \\
 101010 \text{ Partial products} \\
 000000 \\
 + 101010 \\
 \hline
 420 = 111001110 \text{ Product (N+M bits)}
 \end{array}$$

Array Multiplier

- Area - $N \times M$ cells
- Delay - $O(N+M)$



Carry-Save



Multiplier Pipelining

- Register cost:
 - Multiplicand – $(N \text{ bits/stage} \times M \text{ stages})$
 - Multiplier – $(M^2 + M) / 2 \text{ bits}$
 - Early output values – $(M^2 + M) / 2 \text{ bits}$
 - Total – $M \times (N + M + 1) \text{ bits}$
- Critical path = max:
 - DFF + FA + setup
 - Bottom-level adder

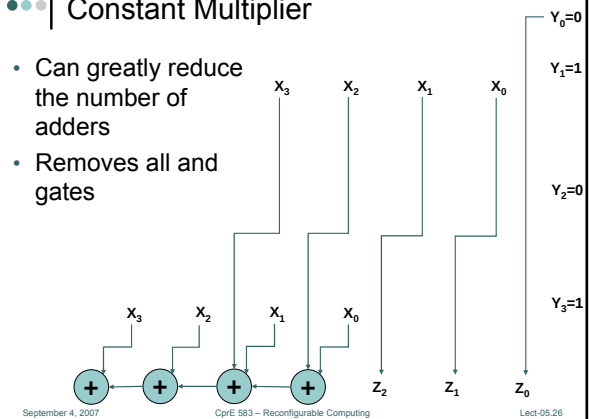
September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.25

Constant Multiplier

- Can greatly reduce the number of adders
- Removes all and gates



September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.26

LUT-based Constant Multipliers

- k -LUT can perform constant multiply of k -bit number
- Break operand into k -bit quantities
- Example: 8-bit x 8-bit constant, $k=4$

$$\begin{array}{r}
 10101011 \\
 \times \text{NNNNNNNN} \\
 \hline
 \text{AAAAAAAAAA} \quad (N * 1011 \text{ (LSN)}) \\
 + \text{BBBBBBBBBBBB} \quad (N * 1010 \text{ (MSN)}) \\
 \hline
 \text{SSSSSSSSSSSS} \quad \text{Product}
 \end{array}$$

September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.27

Summary

- Latency overhead of programmable logic
- Several approaches to reducing design latency:
 - Fast carry
 - Cascade
 - Hardwired connections
- Multiplier optimization goals different from adder
- Other techniques:
 - Logarithmic v. linear (Wallace Tree multiplier)
 - Data encoding (Booth's multiplier)

September 4, 2007

CprE 583 – Reconfigurable Computing

Lect-05.28