Digital Design and Computer Organization

Wallace Tree Multiplier - 2

Reetinder Sidhu

Department of Computer Science and Engineering

Digital Design and Computer Organization

Wallace Tree Multiplier - 2

Reetinder Sidhu

Department of Computer Science and Engineering

Wallace Tree Multiplier - 2

Course Outline

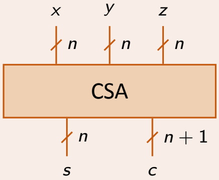
- Digital Design
  - Combinational logic design
  - Sequential logic design
    - ★ **Wallace Tree Multiplier - 2**
- Computer Organization
  - Architecture (microprocessor instruction set)
  - Microarchitecure (microprocessor operation)

Concepts covered

- Wallace Tree Multiplication

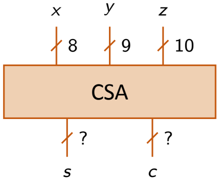
Wallace Tree Multiplier - 2
Carry Save Adder

Basic Carry Save Adder



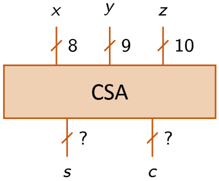
- Contains n full adders
- **Inputs** Three n-bit numbers
- **Outputs**
  - One n-bit number
  - One $(n+1)$-bit number (whose LSB is 0)

Wallace Tree Multiplier - 2
Different Sized Inputs to CSA



Wallace Tree Multiplier - 2

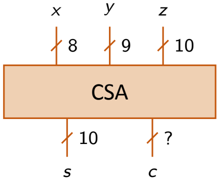
Different Sized Inputs to CSA



- Sum output size will be same as size of largest input

Wallace Tree Multiplier - 2

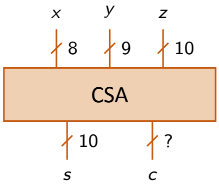
Different Sized Inputs to CSA



- Sum output size will be same as size of largest input

Wallace Tree Multiplier - 2

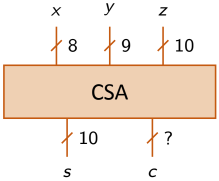
Different Sized Inputs to CSA



- Sum output size will be same as size of largest input
- Computed carry size will be only 9 bits

Wallace Tree Multiplier - 2

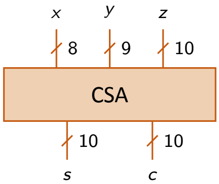
Different Sized Inputs to CSA



- Sum output size will be same as size of largest input
- Computed carry size will be only 9 bits
- So after left shift carry output is 10 bits

Wallace Tree Multiplier - 2

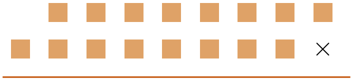
Different Sized Inputs to CSA



- Sum output size will be same as size of largest input
- Computed carry size will be only 9 bits
- So after left shift carry output is 10 bits

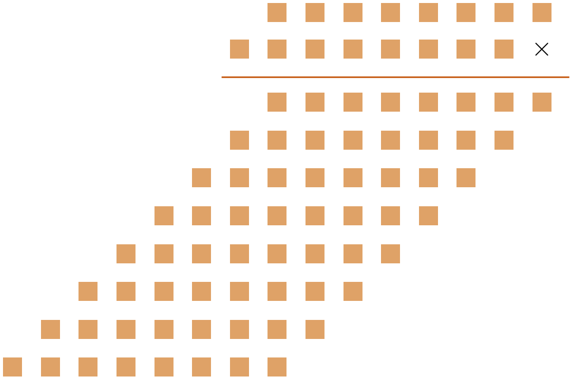
Wallace Tree Multiplier - 2

- Consider the mutliplication of two 8-bit numbers



Wallace Tree Multiplier - 2

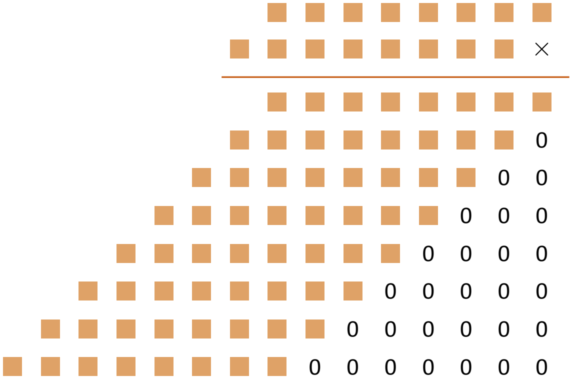
- Consider the mutliplication of two 8-bit numbers



- 8 × 8 array of values computed by 8 × 8 array of two input AND gates

Wallace Tree Multiplier - 2

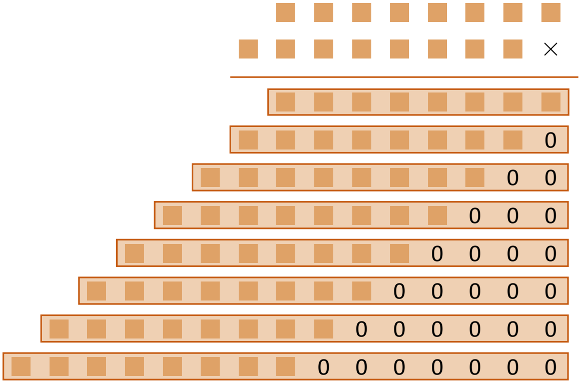
- Consider the mutliplication of two 8-bit numbers



- 8 × 8 array of values computed by 8 × 8 array of two input AND gates

Wallace Tree Multiplier - 2

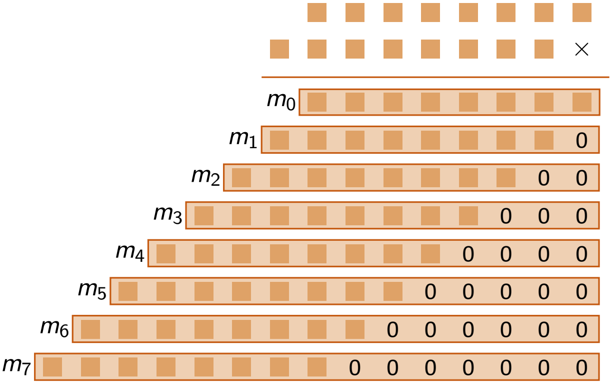
- Consider the mutliplication of two 8-bit numbers



- 8 × 8 array of values computed by 8 × 8 array of two input AND gates

Wallace Tree Multiplier - 2

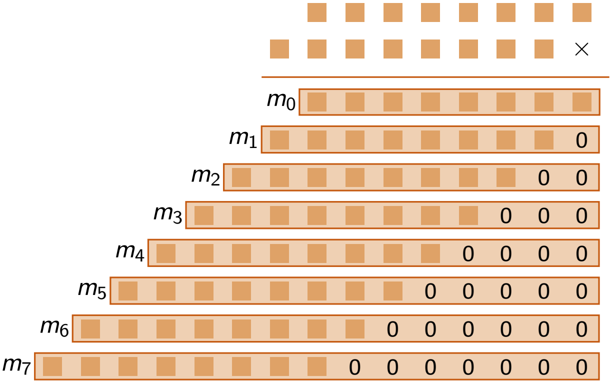
- Consider the mutliplication of two 8-bit numbers



- 8 × 8 array of values computed by 8 × 8 array of two input AND gates

Wallace Tree Multiplier - 2

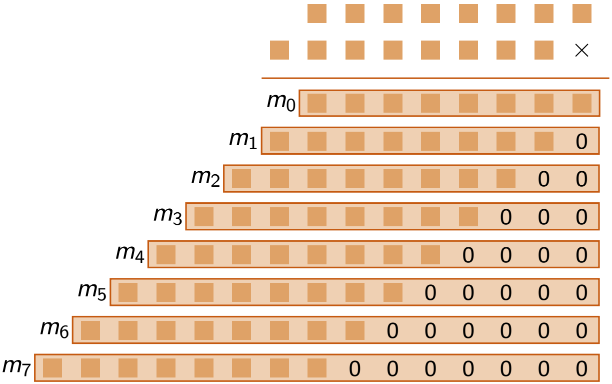
- Consider the mutliplication of two 8-bit numbers



- 8×8 array of values computed by 8×8 array of two input AND gates
- $m_0, \ldots, m_7$ are partial products

Wallace Tree Multiplier - 2

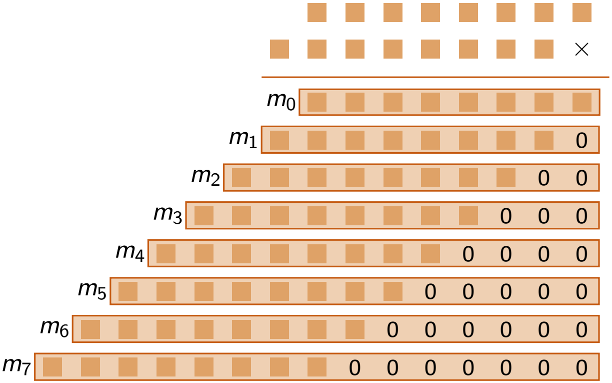
- Consider the mutliplication of two 8-bit numbers



- 8×8 array of values computed by 8×8 array of two input AND gates
- $m_0, \ldots, m_7$ are partial products
- m_i has size $8 + i$ bits

Wallace Tree Multiplier - 2

- Consider the mutliplication of two 8-bit numbers

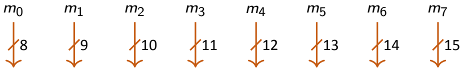


- 8×8 array of values computed by 8×8 array of two input AND gates
- $m_0, \ldots, m_7$ are partial products
- m_i has size $8 + i$ bits

The problem of multiplying two 8-bit numbers has been reduced to the problem of adding the partial products $m_0, \ldots, m_7$

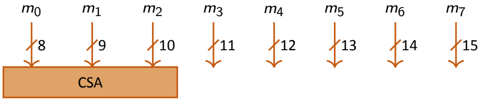
Wallace Tree Multiplier - 2

Carry Save Adders



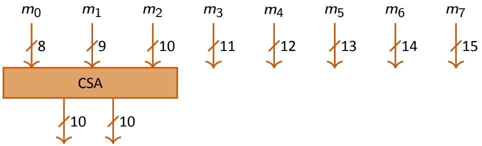
Wallace Tree Multiplier - 2

Carry Save Adders



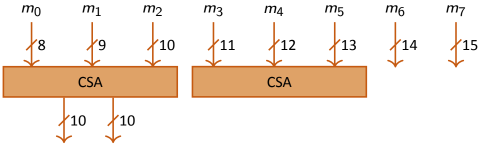
Wallace Tree Multiplier - 2

Carry Save Adders



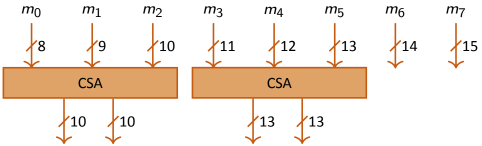
Wallace Tree Multiplier - 2

Carry Save Adders



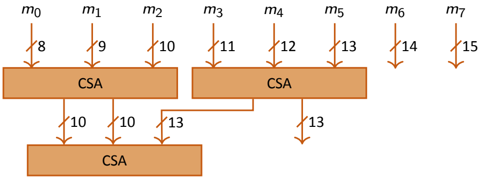
Wallace Tree Multiplier - 2

Carry Save Adders



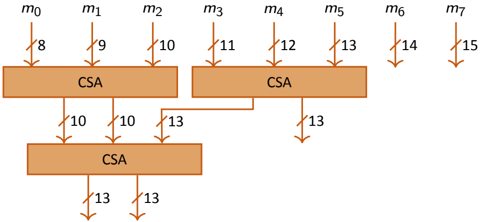
WALLACE TREE MULTIPLIER - 2

Carry Save Adders



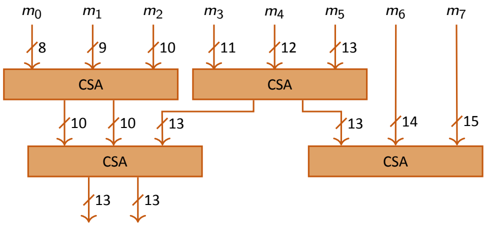
WALLACE TREE MULTIPLIER - 2

Carry Save Adders



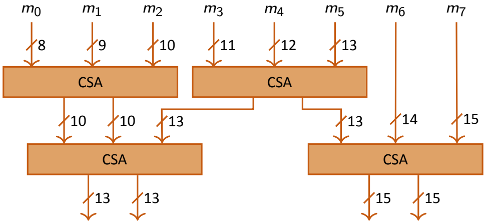
Wallace Tree Multiplier - 2

Carry Save Adders



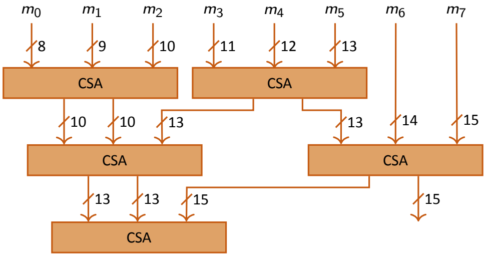
Wallace Tree Multiplier - 2

Carry Save Adders



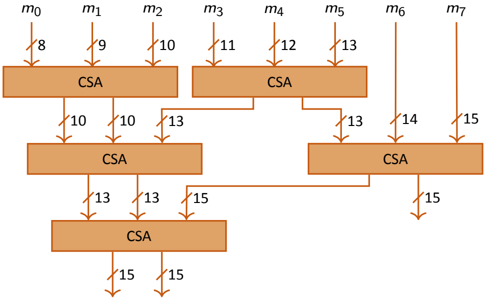
Wallace Tree Multiplier - 2

Carry Save Adders



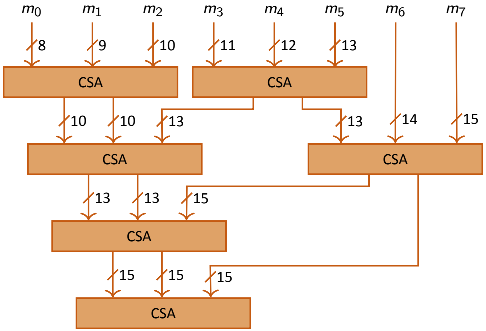
Wallace Tree Multiplier - 2

Carry Save Adders



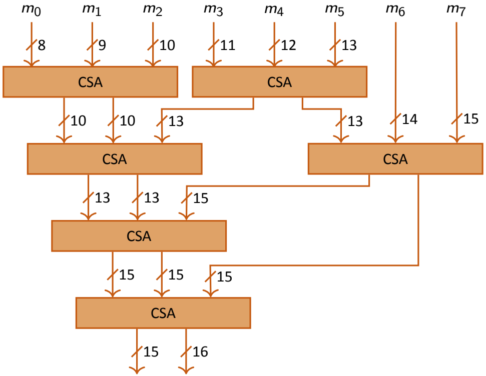
Wallace Tree Multiplier - 2

Carry Save Adders



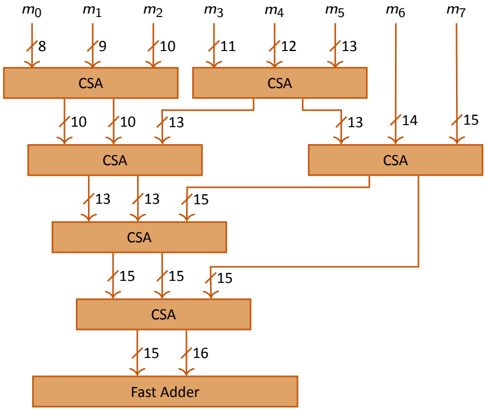
Wallace Tree Multiplier - 2

Carry Save Adders



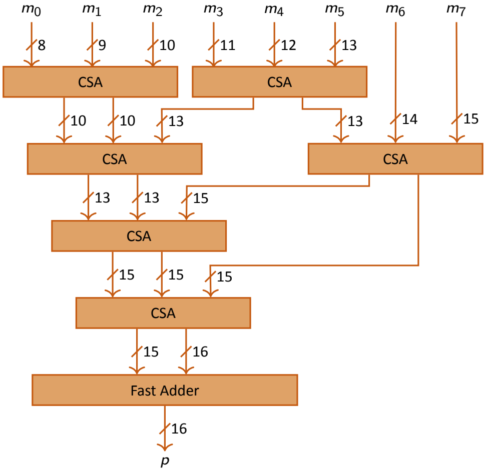
Wallace Tree Multiplier - 2

Carry Save Adders



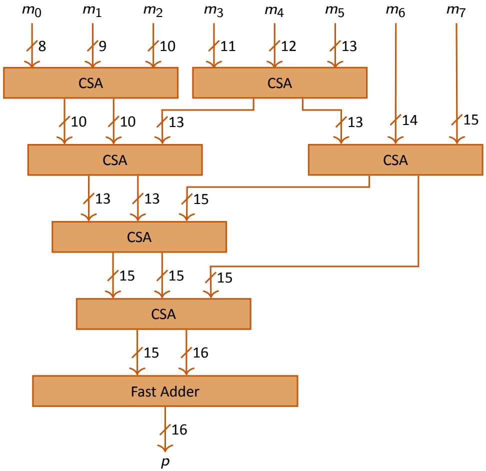
Wallace Tree Multiplier - 2

Carry Save Adders



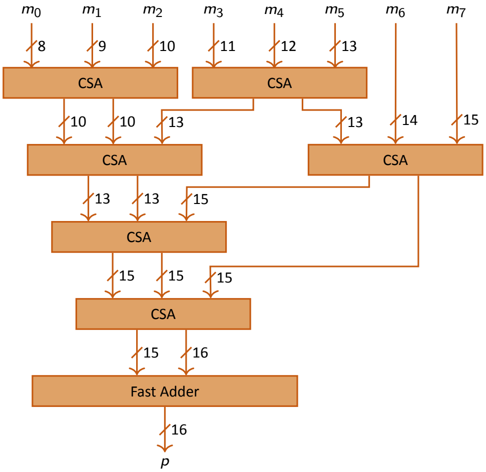
Wallace Tree Multiplier - 2

Carry Save Adders



- Let t_{Adder} be critical path delay of fast adder

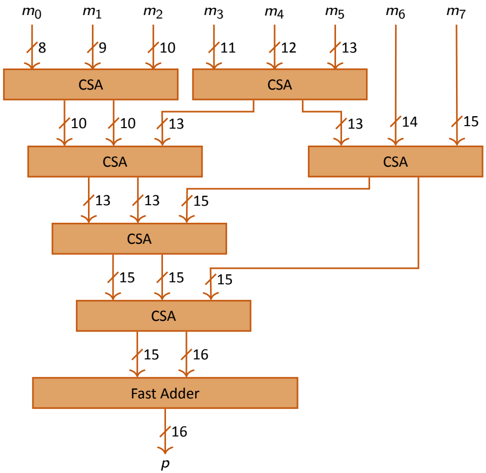
WALLACE TREE MULTIPLIER - 2
Carry Save Adders



- Let t_{Adder} be critical path delay of fast adder
- The critical path delay of the Wallace tree above the adder is just $4t_{FA}$

Wallace Tree Multiplier - 2

Carry Save Adders



- Let t_{Adder} be critical path delay of fast adder
- The critical path delay of the Wallace tree above the adder is just $4t_{FA}$
- Critical path delay of 8×8 Wallace tree multiplier is thus $t_{AND} + 4t_{FA} + t_{Adder}$

Wallace Tree Multiplier - 2

Think About It

- Compare the area and time performance of the Wallace Tree Multiplier with that of the Shift-Add Multiplier
  - Try ripple carry and parallel prefix adders in both