Automated Formal Synthesis of Wallace Tree Multipliers

Osman Hasan Skander Kort

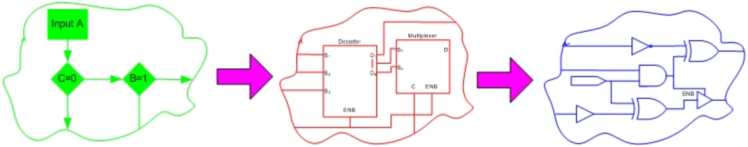
Electrical and Computer Engineering Department
Concordia University, Montreal, Canada

MWSCAS/NEWCAS, 2007

Outline

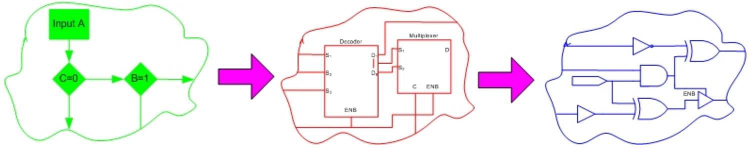
Hardware Synthesis

- Stepwise refinement of circuit descriptions from higher levels of abstraction to lower ones



Hardware Synthesis

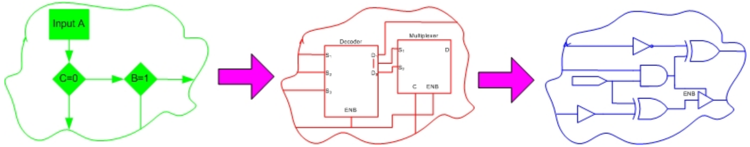
- Stepwise refinement of circuit descriptions from higher levels of abstraction to lower ones



- Automated synthesis tools with sophisticated algorithms
    - Prone to design bugs

Hardware Synthesis

- Stepwise refinement of circuit descriptions from higher levels of abstraction to lower ones



- Automated synthesis tools with sophisticated algorithms
  - Prone to design bugs

- Verification of synthesized results is a necessity
  - 70% of the design time and budget is spent on verification

Synthesis Verification

Synthesis Verification

Pre-Synthesis Verification

Correctness of the synthesis program

- Very tedious
- Impossible for the large synthesis programs

Synthesis Verification

Pre-Synthesis Verification

Correctness of the synthesis program

- Very tedious
- Impossible for the large synthesis programs

Post-Synthesis Verification

Output is verified with respect to the input

- No knowledge about synthesis algorithms
- Abstraction differences complicates the task

Synthesis Verification

Pre-Synthesis Verification

Correctness of the synthesis program

- Very tedious
- Impossible for the large synthesis programs

Post-Synthesis Verification

Output is verified with respect to the input

- No knowledge about synthesis algorithms
- Abstraction differences complicates the task

Formal Synthesis

Synthesis is performed within a formal environment

- Most hardware systems can be handled
- Familiarity with formal semantics and reasoning

Proposed Solution

- Automated Formal Synthesis Approach
  - Formalization and Verification steps are transparent to the user

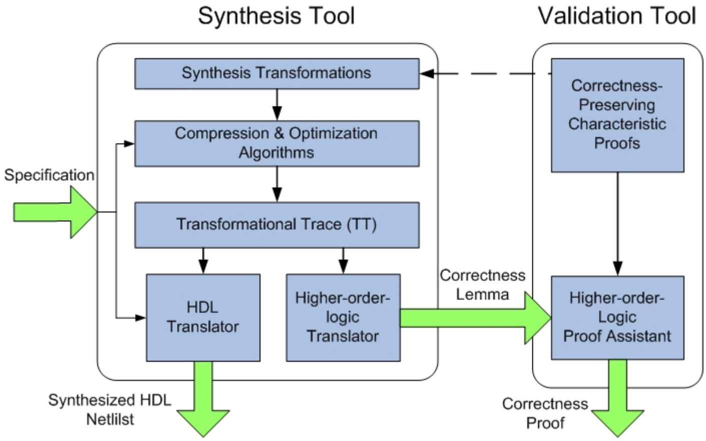
Proposed Solution

- Automated Formal Synthesis Approach
  - Formalization and Verification steps are transparent to the user
- Limit the synthesis process to a specific set of transformations
  - Formally verified to preserve the correctness of the design prior to the actual synthesis process

Proposed Solution

- Automated Formal Synthesis Approach
  - Formalization and Verification steps are transparent to the user
- Limit the synthesis process to a specific set of transformations
  - Formally verified to preserve the correctness of the design prior to the actual synthesis process
- Verification: Higher-Order-Logic Theorem Proving
  - Higher-Order-Logic
    - System of deduction with a precise semantics
    - High Expressiveness
  - Theorem Provers
    - Computer based proof tools

Methodology


Synthesis Tool
Validation Tool
Synthesis Transformations
Compression & Optimization Algorithms
Transformational Trace (TT)
HDL Translator
Higher-order-logic Translator
Specification
Synthesized HDL Netlilst
Correctness Lemma
Correctness-Preserving Characteristic Proofs
Higher-order-Logic Proof Assistant
Correctness Proof

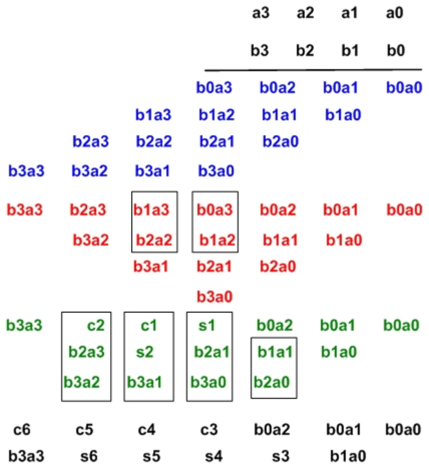
Wallace Tree Multipliers

Wallace Tree Multipliers

- Fast multiplication of wide operands
- Using Full Adder (FA) and Half Adder (HA) cells

Wallace Tree Multipliers

- Fast multiplication of wide operands
- Using Full Adder (FA) and Half Adder (HA) cells



Automated Formal Synthesis of Wallace Tree Multipliers

- FA and HA transformations are correctness preserving transformations

Automated Formal Synthesis of Wallace Tree Multipliers

- FA and HA transformations are correctness preserving transformations
- Synthesis Tool
  - C++
  - Accepts the width of operands
  - Returns the synthesized gate-level netlist and the correctness lemma

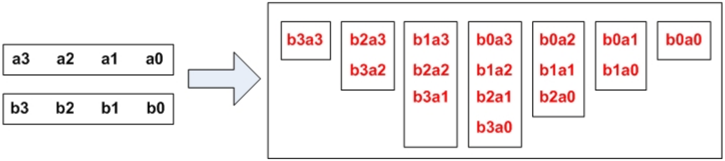
Automated Formal Synthesis of Wallace Tree Multipliers

- FA and HA transformations are correctness preserving transformations
- Synthesis Tool
  - C++
  - Accepts the width of operands
  - Returns the synthesized gate-level netlist and the correctness lemma
- Validation Tool
  - Isabelle/HOL
  - Accepts the correctness lemma
  - Returns True if the synthesis is complete and correct

Formalization of Wallace Tree Multipliers

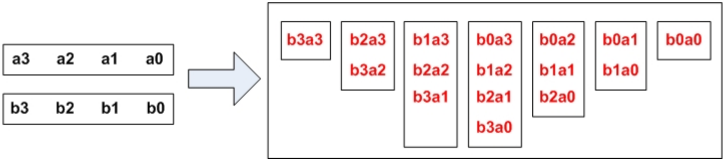
Formalization of Wallace Tree Multipliers

- `w_tree`: Generates a Wallace Tree

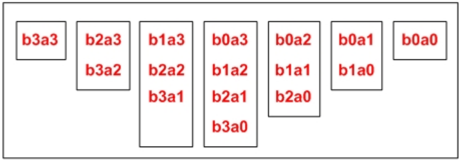


Formalization of Wallace Tree Multipliers

- `w_tree`: Generates a Wallace Tree



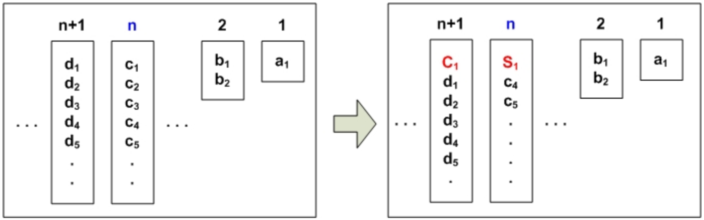
- `eval`: Computes integer value of a Wallace Tree



$$s_6 2^6 + s_5 2^5 + s_4 2^4 + s_3 2^3 + s_2 2^2 + s_1 2^1 + s_0 2^0$$

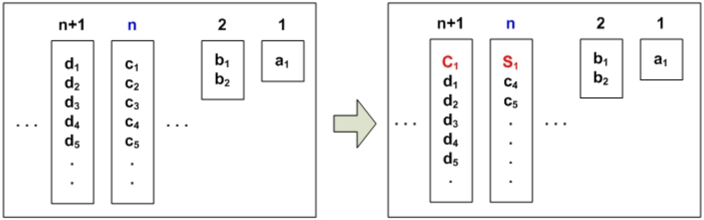
Formalization of Wallace Tree Transformations

- `fa_trans_tree`: FA transformation on a Wallace Tree



Formalization of Wallace Tree Transformations

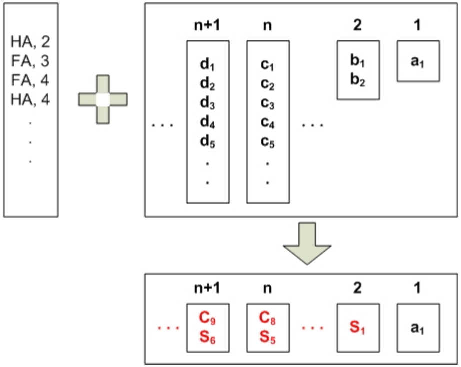
- `fa_trans_tree`: FA transformation on a Wallace Tree



- Similar functions for the HA Transformation

Formalization of Wallace Tree Transformations

- apply_trans: Applies a sequence of FA and HA transformation to a Wallace Tree



Wallace Tree Multiplier Verification

Wallace Tree Multiplier Verification

Theorem: FA transformation is correctness preserving

$\vdash \forall \mathtt{w}, \mathtt{n}.\ \mathtt{eval}(\mathtt{fa\_trans\_tree}\ \mathtt{w}\ \mathtt{n}) = \mathtt{eval}\ \mathtt{w}$

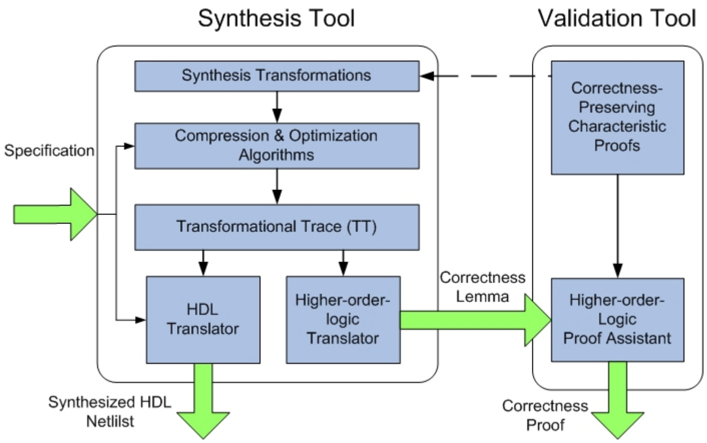
Theorem: HA transformation is correctness preserving

$\vdash \forall \mathtt{w}, \mathtt{n}.\ \mathtt{eval}(\mathtt{ha\_trans\_tree}\ \mathtt{w}\ \mathtt{n}) = \mathtt{eval}\ \mathtt{w}$

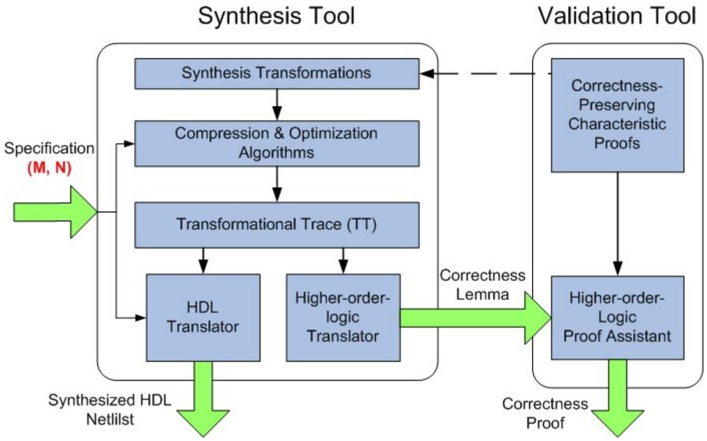
Theorem: Correctness of Wallace Tree synthesis

$\vdash \forall \mathtt{a}, \mathtt{b}, \mathtt{t}.\ \mathtt{eval}(\mathtt{apply\_trans}(\mathtt{w\_tree}\ \mathtt{a}\ \mathtt{b})\ \mathtt{t}) = \mathtt{a}_{10} * \mathtt{b}_{10}$

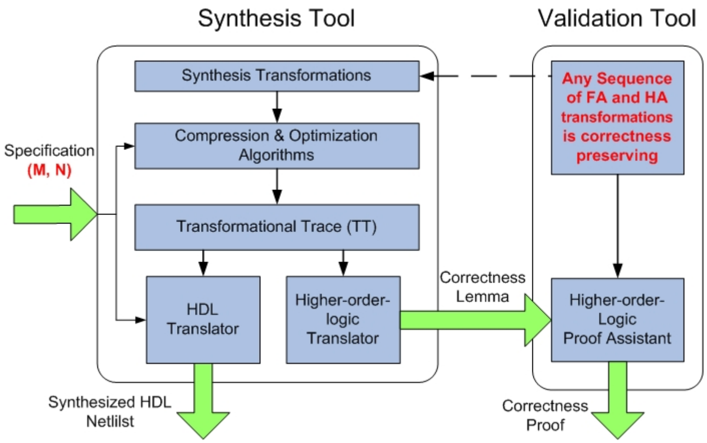
Synthesis of a MxN Multiplier


Synthesis Tool
Validation Tool
Synthesis Transformations
Compression & Optimization Algorithms
Transformational Trace (TT)
HDL Translator
Higher-order-logic Translator
Specification
Synthesized HDL Netlilst
Correctness Lemma
Correctness-Preserving Characteristic Proofs
Higher-order-Logic Proof Assistant
Correctness Proof

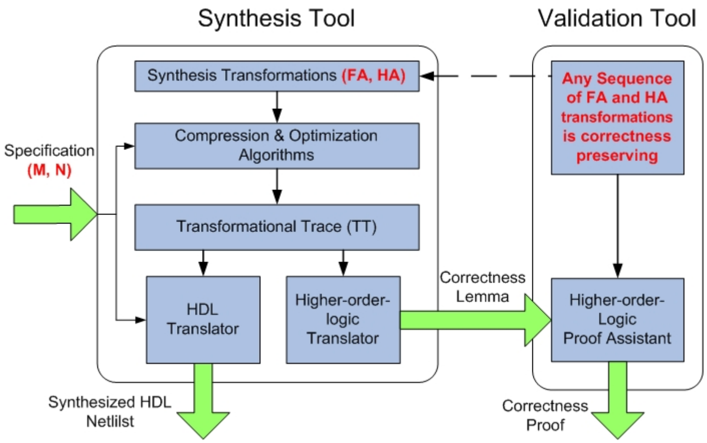
Synthesis of a MxN Multiplier


Synthesis Tool
Validation Tool
Synthesis Transformations
Compression & Optimization Algorithms
Transformational Trace (TT)
HDL Translator
Higher-order-logic Translator
Specification (M, N)
Synthesized HDL Netlilst
Correctness Lemma
Correctness-Preserving Characteristic Proofs
Higher-order-Logic Proof Assistant
Correctness Proof

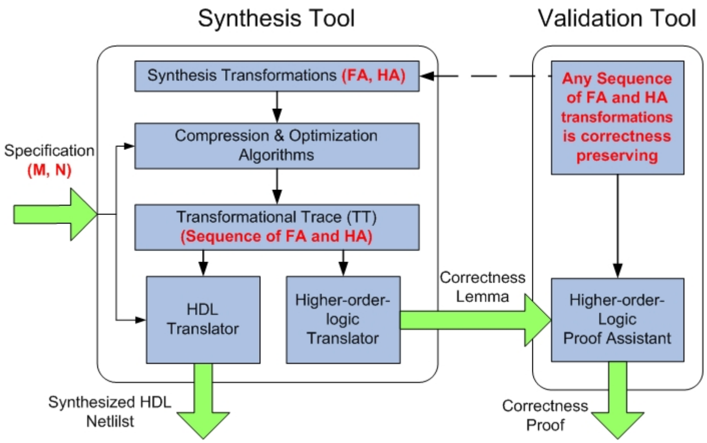
Synthesis of a MxN Multiplier


Synthesis Tool
Validation Tool
Synthesis Transformations
Compression & Optimization Algorithms
Transformational Trace (TT)
HDL Translator
Higher-order-logic Translator
Specification
(M, N)
Correctness Lemma
Any Sequence of FA and HA transformations is correctness preserving
Higher-order-Logic Proof Assistant
Synthesized HDL Netlilst
Correctness Proof

Synthesis of a MxN Multiplier


Synthesis Tool
Validation Tool
Synthesis Transformations (FA, HA)
Compression & Optimization Algorithms
Transformational Trace (TT)
HDL Translator
Higher-order-logic Translator
Specification (M, N)
Synthesized HDL Netlilst
Correctness Lemma
Any Sequence of FA and HA transformations is correctness preserving
Higher-order-Logic Proof Assistant
Correctness Proof

Synthesis of a MxN Multiplier


Synthesis Tool
Validation Tool
Synthesis Transformations (FA, HA)
Compression & Optimization Algorithms
Transformational Trace (TT) (Sequence of FA and HA)
HDL Translator
Higher-order-logic Translator
Specification (M, N)
Synthesized HDL Netlilst
Correctness Lemma
Any Sequence of FA and HA transformations is correctness preserving
Higher-order-Logic Proof Assistant
Correctness Proof

Related Work

Related Work

- Formal Synthesis
  - A formal approach to specify and synthesize at the system level. [Blumenröhr, 1999]
  - Formal synthesis at the algorithmic level. [Blumenröhr et. al, 1999]

Related Work

- Formal Synthesis
  - A formal approach to specify and synthesize at the system level. [Blumenröhr, 1999]
  - Formal synthesis at the algorithmic level. [Blumenröhr et. al, 1999]

None of the existing approaches offers automated synthesis

Related Work

- Formal Synthesis
  - A formal approach to specify and synthesize at the system level. [Blumenröhr, 1999]
  - Formal synthesis at the algorithmic level. [Blumenröhr et. al, 1999]

None of the existing approaches offers automated synthesis

- Multiplier Verification
  - On the complexity of VLSI implementations and graph representations of Boolean functions with applications to integer multiplication. [Bryant, 1991]
  - Mechanically Verifying a family of multiplier circuits [Kapur, 1996]
  - Polynomial formal verification of multipliers. [Keim et. al, 2003]

Related Work

- Formal Synthesis
  - A formal approach to specify and synthesize at the system level. [Blumenröhr, 1999]
  - Formal synthesis at the algorithmic level. [Blumenröhr et. al, 1999]

None of the existing approaches offers automated synthesis

- Multiplier Verification
  - On the complexity of VLSI implementations and graph representations of Boolean functions with applications to integer multiplication. [Bryant, 1991]
  - Mechanically Verifying a family of multiplier circuits [Kapur, 1996]
  - Polynomial formal verification of multipliers. [Keim et. al, 2003]

Formal synthesis has never been used with multipliers before

Conclusions

Conclusions

- Automated Formal Synthesis Methodology
  - User friendly
  - Formally verified synthesized results
  - Speeds up the design process

Conclusions

- Automated Formal Synthesis Methodology
  - User friendly
  - Formally verified synthesized results
  - Speeds up the design process
- Formal Synthesis of Wallace Tree Multipliers
  - Capability to handle synthesis of wide multipliers

Conclusions

- Automated Formal Synthesis Methodology
  - User friendly
  - Formally verified synthesized results
  - Speeds up the design process
- Formal Synthesis of Wallace Tree Multipliers
  - Capability to handle synthesis of wide multipliers
- Future Work
  - Application to other digital circuits

More details and Isabelle/HOL sources

- Contact: o_hasan@ece.concordia.ca

Thank You!