Using Formal Techniques for Design for Verifiability

Rolf Drechsler

University of Bremen
DFKI GmbH
Germany

drechsler@uni-bremen.de

@Rolf_Drechsler

Verification

- It is important
  - Trust me!
- Very powerful tools in the market
  - Formal verification
- For formal tools: little understanding of behavior

How does verification work?

- Circuit is designed

- Handed to verification tool
  - Simulation/emulation
  - Formal techniques

How does verification work?

- Circuit is designed
- Handed to verification tool
  - Simulation/emulation
  - **Formal techniques**

What would we like to have?

- Prediction
  - Run time
  - Memory requirement
- Polynomial

Questions:

- **Can this work for any / all circuits?**
- **How do these circuits look like?**

Example: multiplier verification

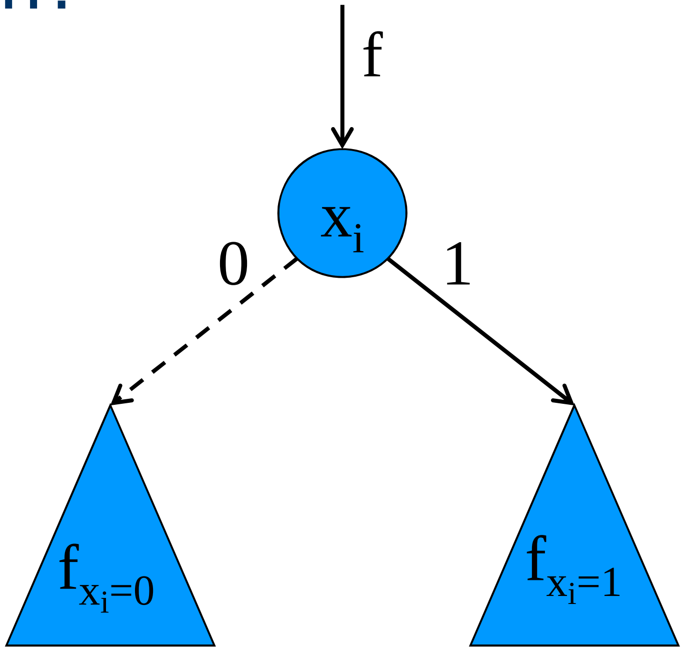
- Formal Verification of Integer Multipliers by Combining Gröbner Basis with Logic Reduction
  (Sayed-Ahmed et al, DATE, 2016)
  - 128-bit multiplier verified
- Polynomial verification of multipliers (Keim et al., Formal Methods in System Design, 2003)
  - Based on *BMDs (difficult DD type)

Design for verifiability

- **Goal:** Design circuits such that
  - *Formally* verifiable
  - *Polynomial* bounds

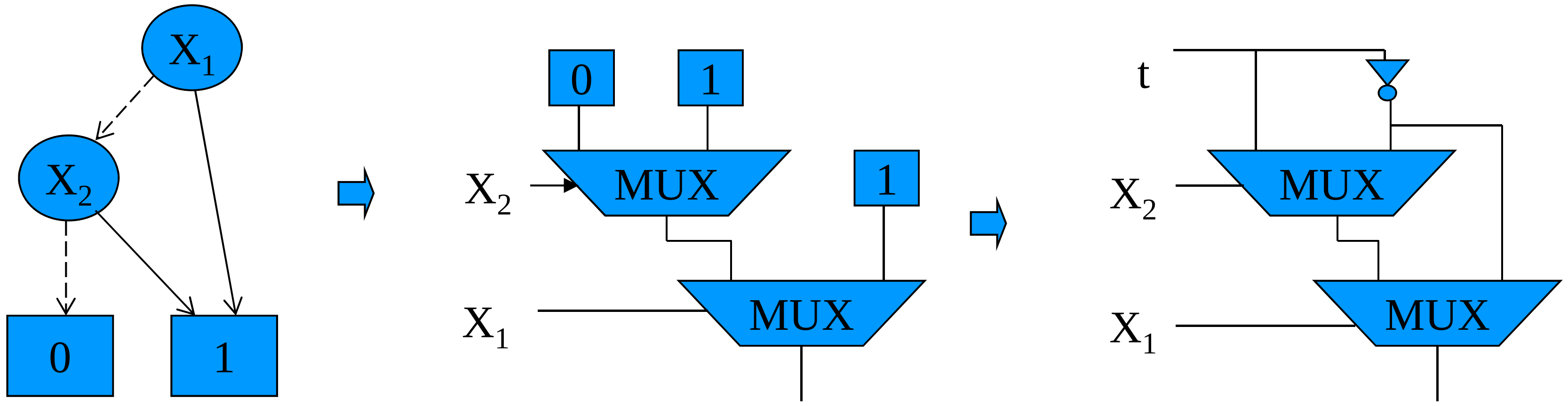
Binary Decision Diagrams

- Shannon decomposition:
  $$f = \overline{x}_i \cdot f_{x_i=0} + x_i \cdot f_{x_i=1}$$
- Terminals: ‘0‘, ‘1‘
- Ordered and reduced BDDs
- Canonical data structure



Derive circuits from BDDs

- Synthesis of fully testable circuits from BDDs (Drechsler et al, TCAD, 2004)
- Each node is substituted by a multiplexor
- Example: $f(x_1, x_2) = x_1 + x_2$

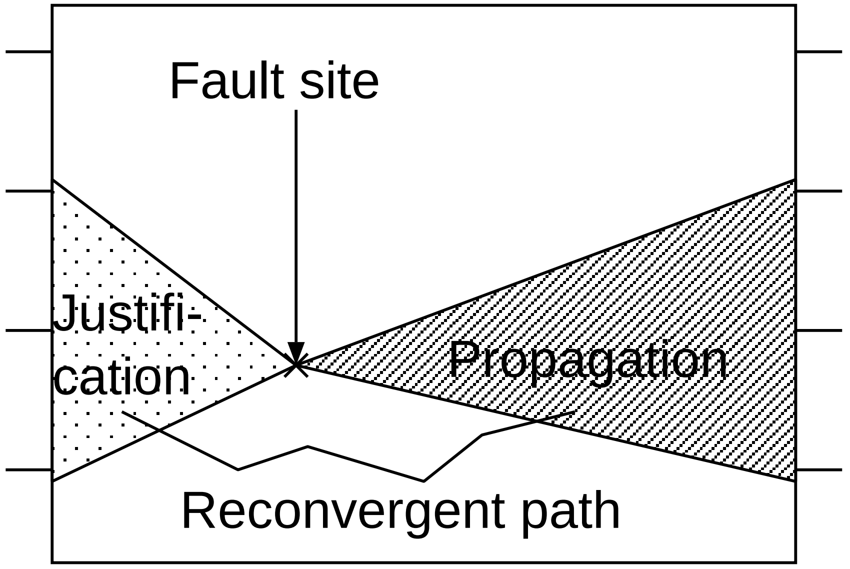


Consider Construction

- Small BDD does not imply small BDD during construction!
  - Otherwise: *tautology checking* would be trivial

- But, interesting to look at **BDD results**: Bern et al: Global rebuilding of OBDDs Avoiding Memory Requirement Maxima. CAV 1995

What makes verification hard?



- Similar to test generation
- Circuit structure
- Tree-like -> polynomial verification (e.g. by BDDs)

- But how about reconvergent paths?

Reverse engineer formal tools

- E.g.: what makes SAT solvers efficient?
  - Implication graphs
  - Learning
  - Non-chronological backtracking
  - ...

- How do these circuits look like?

Conclusions

- **Today:** very powerful formal verification tools
  - But: little understanding
- **Research goal:**
  - Designing circuits that are by construction **provably** formally verifiable
- Works for BDDs, but **not trivial**!
- Future work: extension to KFDDs, SAT, SMT,...

Using Formal Techniques for Design for Verifiability

Rolf Drechsler

University of Bremen
DFKI GmbH
Germany

drechsler@uni-bremen.de

@Rolf_Drechsler