Formal Verification of a Basic Circuits Library

Christoph Berg, Christian Jacobi, Daniel Kroening

Saarland University, Computer Science Department
66123 Saarbrücken, Germany
{cb,cj,kroening}@cs.uni-sb.de

February 2001

OUTLINE

- Why Verify?
- A Basic Circuits Library
- Related Work
- Leading Zero Counter & Correctness Proof
- Conclusion

Why Verify?

- Why verify at all? Proofs are tedious to find!
- There is a need for correct hardware
  - Cf. the ‘Pentium-Bug’
- Verified hardware is *provably* correct
- Software verification relies on correct hardware

Basic Circuits

- Hardware design is highly modularized
- Some modules frequently reused
- Adders, shifters, multipliers, decoders, encoders, ...
- Correctness of the whole design depends on the correctness of the components

THE LIBRARY

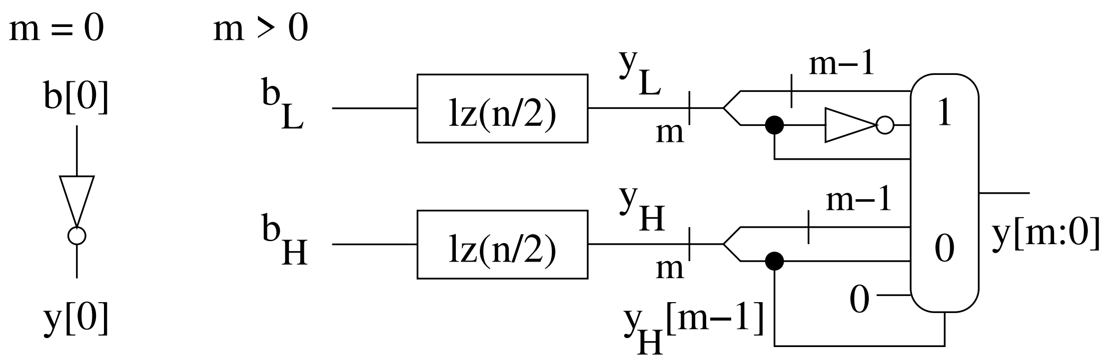
- General purpose modules of basics circuits
- Arbitrary bit width
- Specification in PVS (convertable to Verilog HDL)
- Mathematical formula as correctness condition
- Freely available to researchers

- Designs and proofs are based on:
  S. Mueller, W. J. Paul: *Computer Architecture: Complexity and Correctness*. Springer, 2000.

Related Work

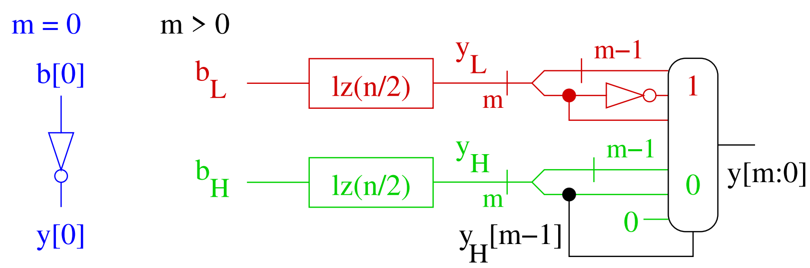
- Library is used in the DLX verification project:
  DLX: 32-bit RISC processor, 5-stage pipeline/Tomasulo scheduler, precise interrupts, fully IEEE-compliant FPU, cache memory interface
- Equivalence checking (Clarke, Bryant)
- ACV (Bryant)

LEADING ZERO COUNTER



- Counts zeros at the beginning of a bit string (of length $n = 2^m$)
- $\overbrace{00000}^{5}\,11001001101 \mapsto 00101 = \text{bin}(5)$
- Used in FP rounding units
- Formalize the above figure as PVS code

Leading Zero Counter (2)



- $m = 0, n = 2^0$: $\qquad lz\_impl(b) := \neg b[0]$
- $m > 0, n = 2^m$:

$$\left.\begin{array}{ll} y_L = lz\_impl(b_L), & b_L = b[2^{m-1}-1, 0] \\ y_H = lz\_impl(b_H), & b_H = b[2^m - 1, 2^{m-1}] \end{array}\right\} \underbrace{00010111}_{b_H}\,\underbrace{00101111}_{b_L}$$

$$lz\_impl(b) := \begin{cases} y_L[m-1] \circ \neg y_L[m-1] \circ y_L[m-2,0] & \\ \qquad\qquad \text{if } y_H[m-1] & \\ \mathbf{0} \circ y_H \qquad \text{otherwise} & \end{cases}$$

Counting Zeros

- What does ‘counting leading zeros’ mean formally?
- $lzero(b) = \max\{i \in \mathbb{N} \mid i = 0 \vee (i \leq n \wedge b[n-1, n-i] = \mathbf{0}^i)\}$

$$\underbrace{\overbrace{000}^{i}\;00}_{\max}\,10010$$

- ```
  lzero(b):nat = max({ i:nat | i=0 OR
     (i<=n AND b^(n-1, n-i) = fill[i](FALSE)) })
  ```
  ```

PROOF

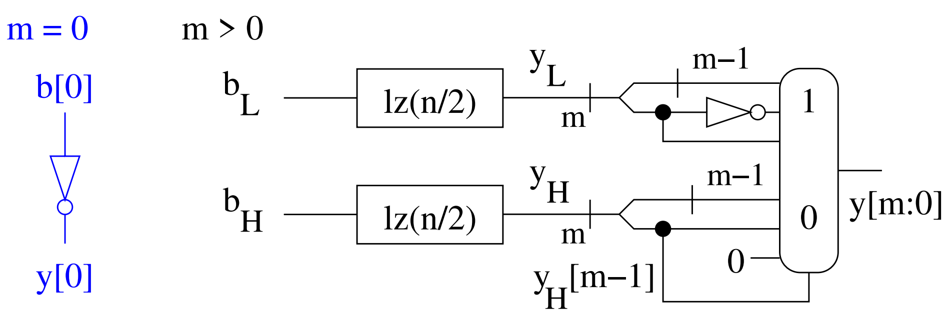
- Show that for every input the circuits counts correctly the number of leading zeros
- **Theorem 1** *For all* $m \in \mathbb{N}, b \in \{\mathbf{0}, \mathbf{1}\}^{2^m}$:

$$\langle lz\_impl(b) \rangle = lzero(b)$$

- The induction base $m = 0, n = 1$ is trivial:

$$lz\_impl(0) = \neg 0 = 1$$

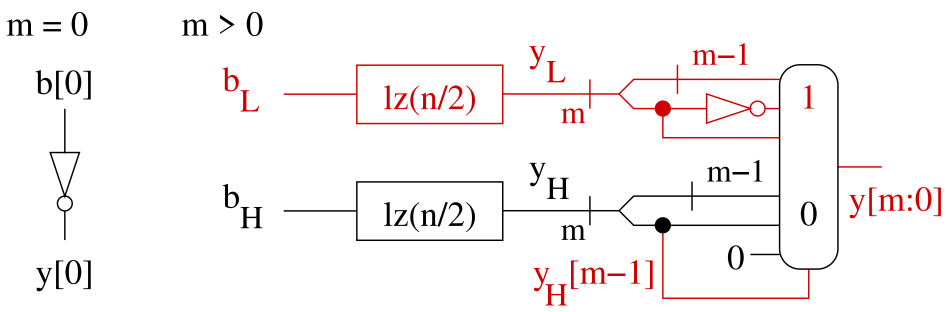
$$lz\_impl(1) = \neg 1 = 0$$

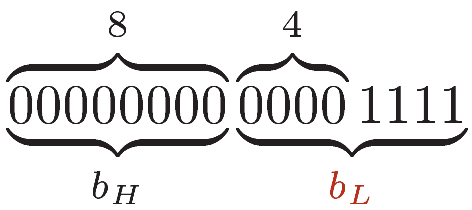


PROOF (2)

- Case $lzero(b) \geq \frac{2^m}{2} \implies y_H[m-1] = 1$:

$$\begin{aligned}\langle lz\_impl(b)\rangle &= \langle y_L[m-1] \circ \neg y_L[m-1] \circ y_L[m-2,0]\rangle \\ &= \frac{2^m}{2} + \langle y_L\rangle \\ &=_{IH} \frac{2^m}{2} + lzero(b_L) = lzero(b) \qquad \square\end{aligned}$$





Conclusion

- Library consists of adders, incrementers, shifters, decoder, encoder, zero tester, absolute value, multiplier
- Library is successfully used in the DLX project (March 2001)
- Future work: implement the verified DLX on a Xilinx FPGA (mid-summer 2001)

PROOF (3)

- Case $lzero(b) < \frac{2^m}{2} \implies y_H[m-1] = 0$:

$$\langle lz\_impl(b)\rangle = \langle lz\_impl(b_H)\rangle =_{IH} lzero(b_H) = lzero(b)$$

$$\underbrace{\overbrace{000}^{3}\,10111}_{b_H}\,\underbrace{00101111}_{b_L}$$