

# Formal Verification of a FIFO Component in Design of Network Monitoring Hardware

CESNET 2006

Tomáš Kratochvíla, Vojtěch Řehák, and **David Šafránek**

8th March, 2006



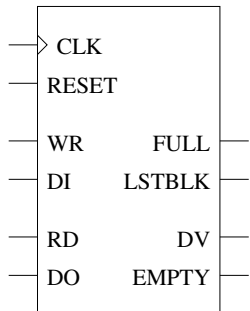
# Content

- 1 Description of FIFO BRAM component
- 2 Verification process
- 3 Parameter settings
- 4 Principles of used formal techniques
- 5 Conclusion

# Purpose of the FIFO BRAM Component

- common FIFO component
  - ▶ reused throughout the project
- implemented in FPGA (Virtex-II)
  - ▶ using internal Virtex-II block RAM memories
- generic component
  - ▶ capacity parametrised — number of `ITEMS`
- parametrised block structure
  - ▶ blocks of constant size `BLOCK_SIZE`

# FIFO BRAM as a Blackbox



## Specification:

[http://www.liberouter.org/cgi-bin2/cvsweb.cgi/liberouter/vhdl\\_design/units/common/fifo\\_bram/doc](http://www.liberouter.org/cgi-bin2/cvsweb.cgi/liberouter/vhdl_design/units/common/fifo_bram/doc)

# FIFO BRAM Specification

## Input ports:

- CLK is a port of the Clock signal
- RESET is a port of the Reset signal
- WR is the Write Request port
- RD is the Read Request port

## Data ports:

- DO is 16bit data output port
- DI is 16bit data input port

# FIFO BRAM Specification

## Output ports:

- EMPTY
  - ▶ active if and only if the FIFO is empty
- FULL
  - ▶ active if and only if the FIFO is full
- LSTBLK
  - ▶ last block detection signal

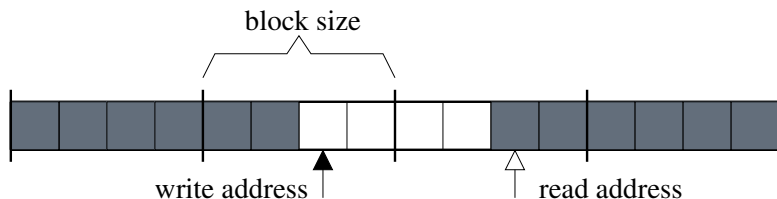
# Last Block Detection

## Specification:

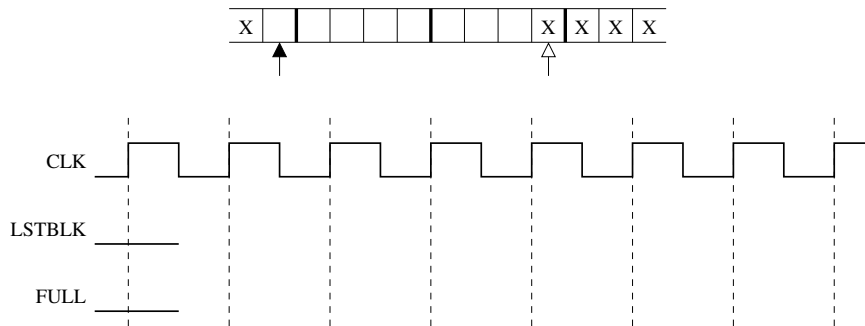
- LSTBLK active if and only if the FIFO contains just `BLOCK_SIZE` or less than `BLOCK_SIZE` number of free items

## Implementation:

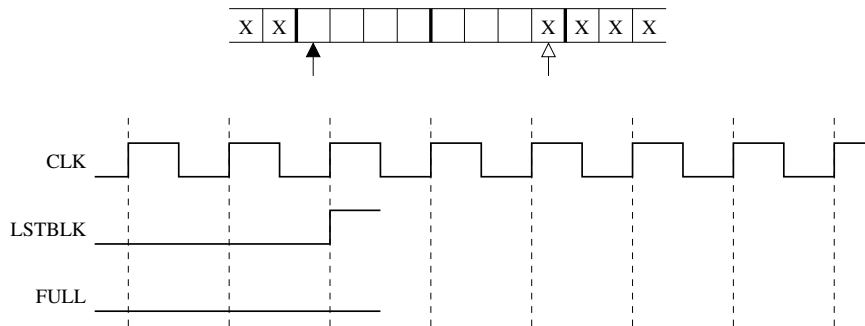
- higher bits of write and read address compared
- LSTBLK active if addresses point to blocks just next to each other



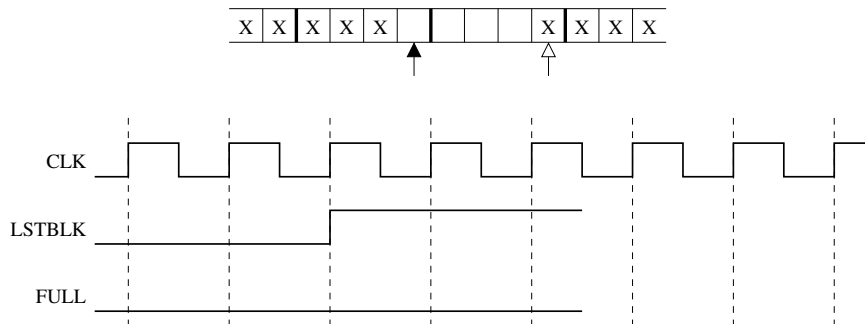
# Last Block Detection



# Last Block Detection



# Last Block Detection



# Generic Assertions

- **FIFO is always eventually full**
- FIFO is always eventually empty
- FIFO is full if and only if the `FULL` signal is active
- FIFO is empty if and only if the `EMPTY` signal is active
- FIFO does not overflow
- FIFO does not underflow
- It is not possible to read from and write to the same address at the same time
- all these assertions verified successfully

# Generic Assertions

- FIFO is always eventually full
- FIFO is always eventually empty
- FIFO is full if and only if the `FULL` signal is active
- FIFO is empty if and only if the `EMPTY` signal is active
- FIFO does not overflow
- FIFO does not underflow
- It is not possible to read from and write to the same address at the same time
- all these assertions verified successfully



# Generic Assertions

- FIFO is always eventually full
- FIFO is always eventually empty
- FIFO is full if and only if the `FULL` signal is active
- FIFO is empty if and only if the `EMPTY` signal is active
- FIFO does not overflow
- FIFO does not underflow
- It is not possible to read from and write to the same address at the same time
- all these assertions verified successfully

# Generic Assertions

- FIFO is always eventually full
- FIFO is always eventually empty
- FIFO is full if and only if the `FULL` signal is active
- FIFO is empty if and only if the `EMPTY` signal is active
- FIFO does not overflow
- FIFO does not underflow
- It is not possible to read from and write to the same address at the same time
- all these assertions verified successfully

# Generic Assertions

- FIFO is always eventually full
- FIFO is always eventually empty
- FIFO is full if and only if the `FULL` signal is active
- FIFO is empty if and only if the `EMPTY` signal is active
- FIFO does not overflow
- FIFO does not underflow
- It is not possible to read from and write to the same address at the same time
- all these assertions verified successfully

# Generic Assertions

- FIFO is always eventually full
- FIFO is always eventually empty
- FIFO is full if and only if the `FULL` signal is active
- FIFO is empty if and only if the `EMPTY` signal is active
- FIFO does not overflow
- FIFO does not underflow
- It is not possible to read from and write to the same address at the same time
- all these assertions verified successfully

# Generic Assertions

- FIFO is always eventually full
- FIFO is always eventually empty
- FIFO is full if and only if the `FULL` signal is active
- FIFO is empty if and only if the `EMPTY` signal is active
- FIFO does not overflow
- FIFO does not underflow
- It is not possible to read from and write to the same address at the same time
  
- all these assertions verified successfully

# Last Block Detection Assertion

In every possible execution of the system, the number of free items in the FIFO is less or equal than the size of a block if and only if the LSTBLK signal is active.

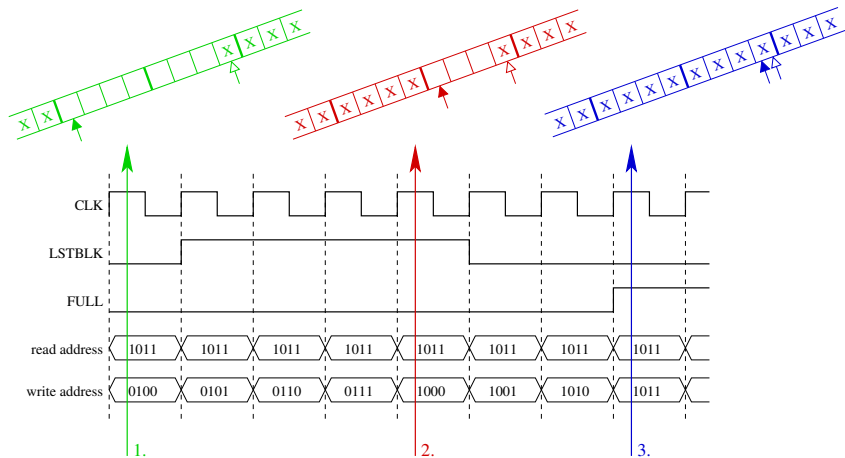
- a critical bug found!

# Last Block Detection Assertion

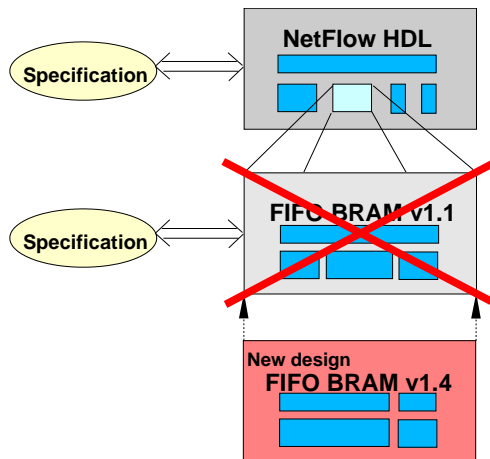
In every possible execution of the system, the number of free items in the FIFO is less or equal than the size of a block if and only if the LSTBLK signal is active.

- **a critical bug found!**

# Bug in Last Block Detection



# New Version







# FIFO BRAM Instances Verified

- generic parameters:

ITEMS, BLOCK\_SIZE, BRAM\_TYPE, and DATA\_WIDTH

- Verified and satisfied:

$2 \leq \text{ITEMS} \leq 128$

$\text{BLOCK\_SIZE} \leq \text{ITEMS}$

$\text{BRAM\_TYPE} \in \{1, 2, 4, 9, 18, 36\}$

$\text{DATA\_WIDTH} \bmod \text{BRAM\_TYPE} = 0 \wedge \text{DATA\_WIDTH} \leq 72$

All parameters found in each Liberouter project.

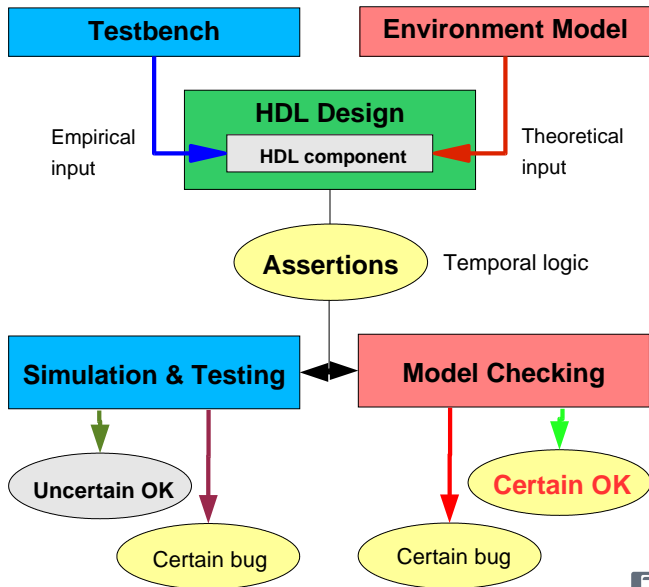
- Unable to verify:  $\text{ITEMS} = 2048 \wedge \text{BLOCK\_SIZE} = 256$



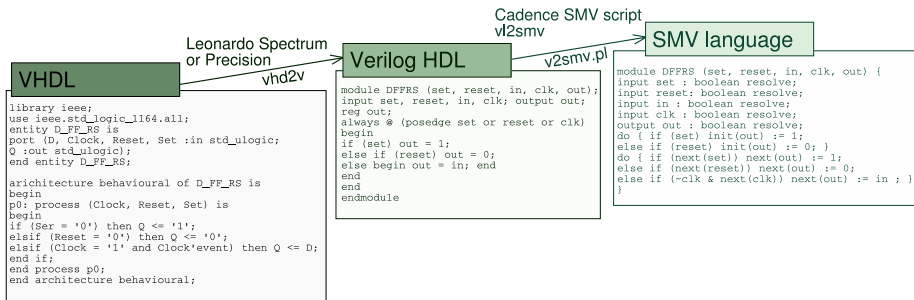
# Formal Techniques Used

- special kind of assertion-based verification
- temporal logic formulae taken as assertions
- model checking method
- direct formal verification of VHDL sources
- VHDL -> Cadence SMV translation

# Model Checking



# VHDL -> SMV Translation



# Conclusion

- formal verification contribution
  - ▶ correctness of the design maximised
  - ▶ conformity of the code w.r.t. the specification achieved
- verification framework reuse
  - ▶ The same set of assertions and verification methodology is being successfully applied in other FIFO components verification:  
[http://www.liberouter.org/cgi-bin2/cvsweb.cgi/liberouter/vhdl\\_design/units/common/fifo](http://www.liberouter.org/cgi-bin2/cvsweb.cgi/liberouter/vhdl_design/units/common/fifo)