Improving Bitstream Compression by Modifying FPGA Architecture

Seyyed Ahmad Razavi, Morteza Saheb Zamani
Amirkabir University of Technology, Tehran, Iran
{a.razavi, szamani}@aut.ac.ir

Presented by: Ehsan K. Ardestani
UC Santa Cruz

Introduction

- Configuration Bitstream
  - 400 Mega bits: Virtex 7
  - Problems:
    - Long time to load to the FPGA
    - Larger non-volatile memory chip
- Solution: Bitstream compression

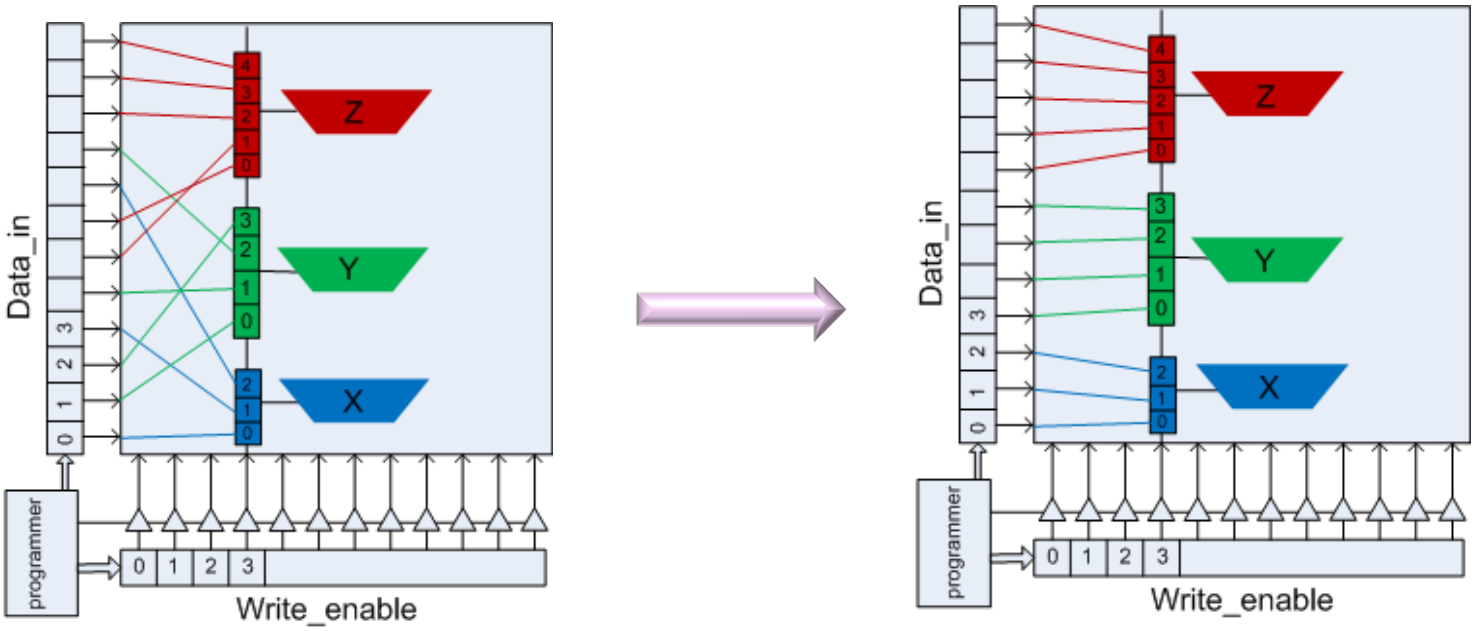
Introduction

- Previous work
  - The effects of routing resource structure on the bitstream compressibility have not been studied.
- Our work:
  - Choose a proper order for configuration bits
  - Choose appropriate bit length for the symbols
  - The order of inputs of switch module MUX

Bitstream format

- The configuration bits of each MUX should be kept together
  - by modifying the *write_enable* and *data_in* lines of SRAM cells
  - => more identical symbols => more compression rate

- Example: Three MUXs: X ($x_0x_1x_2$), Y ($y_0y_1y_2y_3$), Z($z_0z_1z_2z_3z_4$)



"$x_0y_0y_3x_1y_1z_1z_0x_2y_2z_2z_3z_4$"

"$x_0x_1x_2y_0y_1y_2y_3z_0z_1z_2z_3z_4$"

Symbol length

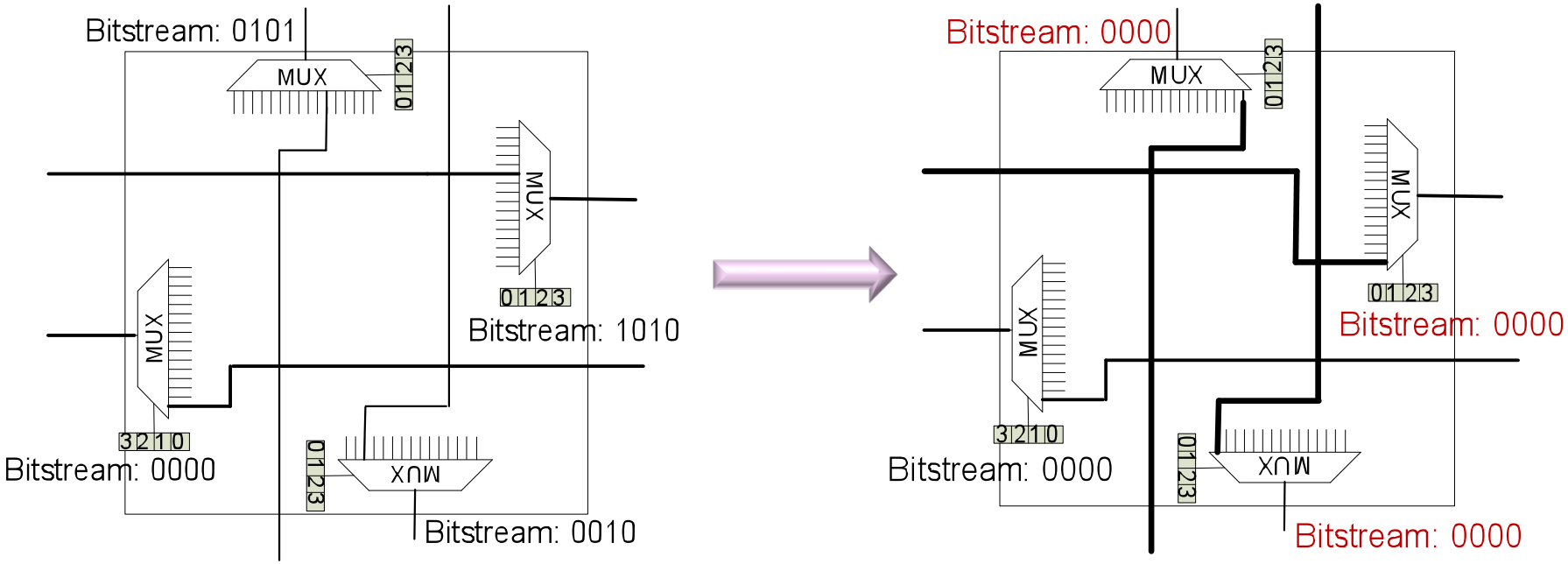
- Symbol length affects the compression rate.

Bitstream	$x_0x_1x_2y_0y_1y_2y_3z_0z_1z_2z_3z_4$
3-bit symbol	$(x_0x_1x_2)$, $(y_0y_1y_2)$, $(y_3z_0z_1)$, $(z_2z_3z_4)$
4-bit symbol	$(x_0x_1x_2y_0)$, $(y_1y_2y_3z_0)$, $(z_1z_2z_3z_4)$
Variable length symbol	$(x_0x_1x_2)$, $(y_0y_1y_2y_3)$, $(z_0z_1z_2z_3z_4)$ Related mask: 3, 4, 5

- Variable symbol length instead of fixed symbol length
  - the configuration bits of each MUX constitute a single symbol.
  - more identical symbols => more compression rate
  - The overhead of mask is low.

The order of Inputs for switch modules' MUXs

- Increasing the number of identical symbols by assigning the same symbols to more frequent routing pattern
  - Some routing patterns in the design are more frequent than the others.
    - E.g. straight routing
  - => Inputs of all the MUXs are ordered in a way that a unique symbol (e.g. "0000") represents all straight patterns.



Our framework

Our framework:

- Integrated with VPR 5
- Generates the HDL code of a tileable homogenous FPGA automatically.
  - Consists of 25 basic tiles
- Supports:
  - Unidirectional routing architecture
  - Any wire segment length, CLB and BLE size, and channel width.
- Generates the required information and scripts for producing the layout
  - e.g. the location of the pins of the tiles which is required for the abutment of the tiles
- Generates the bitstream
- Simulations have verified our FPGA implementation.

Experimental results and Conclusion

- Results:
  - Compression rate is improved by 46% over 20 MCNC benchmarks
- Conclusion and future works:
  - The bitstream format has a significant effect on compression.
  - The FPGA architecture are modified to improve the efficiency of the target compression method.
  - Other architectural modifications for specific compression algorithms