International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

Vlsi Design of Bit-Stream Compression Based On Run Length Encoding

N.Nandeeswari & S.Deva Karun
M-Tech, Dept. of ECE, Kakinada Institute of science and Technology, korangi.
Asst Prof, Dept. of ECE, Kakinada Institute of science and Technology, korangi.

Abstract

Reconfigurable system uses bit-stream compressionto reduce the bit-stream size and the memory requirement. Thecommunication bandwidth is improved reducing theirconfiguration time. Existing research has explored efficientcompression with slow decompression or fast decompression atthe cost of compression efficiency. This paper proposes a decodeaware compression technique to improve both compression anddecompression efficiencies. The three major contributions of thispaper are: i) Efficient bitmask selection technique that can createa large set of matching patterns; ii) Proposes a bitmask basedcompression using the bitmask and dictionary selection techniquethat can significantly reduce the memory requirement iii)Efficient combination of bitmask-based compression and runlength encoding of repetitive patterns.

Keywords:-Bitmask-based compression; decompression engine, FPGA.

1. INTRODUCTION

FIELD-PROGRAMMABLE GATE ARRAYS (FPGA) are widely used in reconfigurable systems. The description of the logic circuit is entered using a hardware description language such as VHDL or Verilog. The logic design is drawn using a schematic editor. Logic synthesizer program is used to transform HDL or schematic into netlist. A netlist is a description of various logic gates in the design and their interconnections. The implementation tool is used to map the logic gates and interconnections into FPGA. The configurable logic block (CLB) in the FPGA contains look up tables (LUT's) which performs the logic operations. The mapping tool collects netlist gates into groups that fit into the LUTs and then the place & route tool assigns the gate collections to specific CLBs while opening or closing the switches in the routing matrices to connect the gates

together. When the implementation phase is complete, a program extracts the state of the switches in the routing matrices and generates a bit-stream where the ones and zeroes correspond to open or closed switches. Since the configuration information for FPGA has to be stored in internal or external memory as bit-streams, the limited memory size, and access bandwidth become the key factors in determining the different functionalities that a system can be configured and how quickly the configuration can be performed. It is quite costly to employ memory with more capacity and access bandwidth, bit-stream compression technique lessen the memory constraint by reducing the size of the bit-stream. The compressed bit-streams stores more configuration information using the same memory. The efficiency of bit-stream compression is measured using Compression Ratio (CR). It is defined as the ratio between the compressed bit-stream size (CS) and the original bit-stream size (OS) (ie CR=CS/OS). A smaller compression ratio implies a better compression technique. Among various compression techniques that has been proposed compression [5] seems to be attractive for bit-stream compression, because of its good compression ratio and relatively simple decompression scheme. This approach combines the advantages of previous compression techniques with good compression ratio and those with fast decompression.

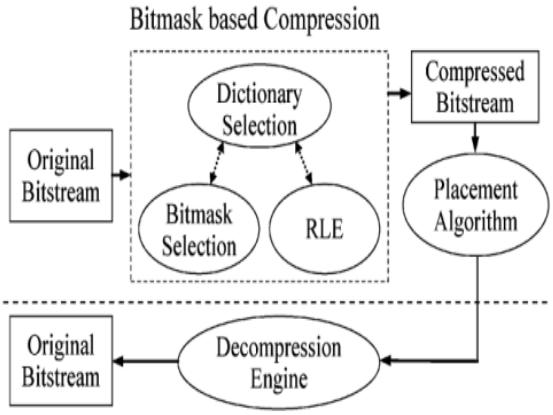

Figure 1. Decode-aware bit-stream compression.

Fig.1 shows decode-aware bit-stream compressionframework. On the compression side, FPGA configuration bitstream is analyzed for selection of profitable dictionary entries and bitmask patterns. The compressed bit-stream is then generateduring bitmask-based compression and run length encoding(RLE). Next, our decode-aware placement algorithm is employed to place the compressed bit-stream in the memory for efficient decompression. During run-time, the compressed bit-stream is transmitted from the memory to the decompression engine, and the original configuration bit-stream is

produced by decompression. Memory and communication bus are designed in multiple of bytes (8 bits), storing dictionaries or transmitting data other than multiple of byte size is not efficient. Therefore, we restrict the symbol length to be multiples of eight in our current implementation. Since the dictionary for bit-stream compression is smaller compared to the size of the bit-stream itself, we use to d=2ito fully utilize the bits for dictionary indexing, where i is the number of indexing bits.

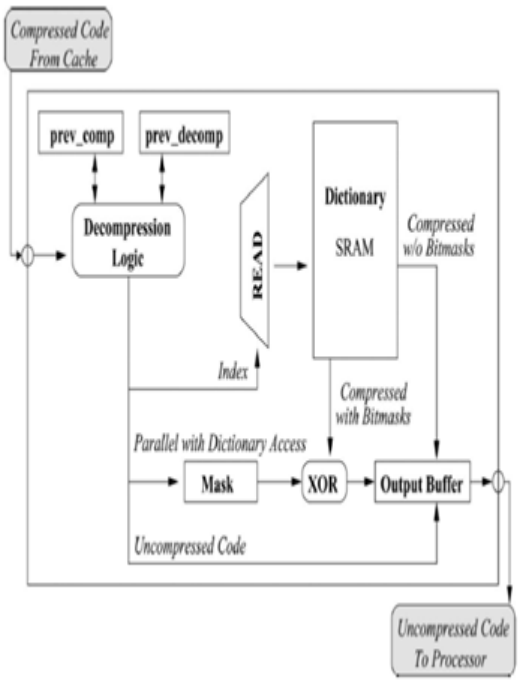

Fig. 2.Decompression Engine.

The decompression engine is a hardware component used to decode the compressed configuration bit-stream and feed the uncompressed bit-stream to the configuration unit in FPGAs. A decompression engine usually has two parts: the buffering circuitry is used to buffer and align codes fetched from the memory, while decoders perform decompression operation to generate original symbols.

2. SIMULATION IMPLEMENTATION

GENERAL

Create a test bench module containing input stimulus to verify the functionality of the counter design. The test bench waveform is a graphical view of the test bench. For a counter; give the clock signal input and the reset input to start the count. These input stimuli are to be specified by writing a test bench module in a new source. The design source should be instantiated in the test bench source and then test bench source is selected for simulate behavioral model. Now this process will take some time and generate a behavioral waveform model as shown in the below figure.

VERILOG

Equipment portrayal dialects, for example, Verilog contrast from programming dialects on the grounds that they incorporate methods for depicting the proliferation of time and flag conditions (affectability).

There are two task administrators, a blocking task (=), and a non-blocking (<=) task. The non-blocking task permits planners to portray a state-machine upgrade without expecting to pronounce and utilize transitory capacity variables (in any broad programming dialect we have to characterize some provisional storage rooms for the operands to be worked on along these lines; those are impermanent capacity variables). Since these ideas are a piece of Verilog's dialect semantics, architects could rapidly compose portrayals of expansive circuits in a generally minimal and succinct structure. At the season of Verilog's presentation (1984), Verilog spoke to a huge efficiency change for circuit originators who were at that point utilizing graphical schematic capture software and uniquely composed programming projects to report and mimic electronic circuits. The originators of Verilog needed a dialect with grammar like the C programming dialect, which was at that point generally utilized as a part of designing programming advancement. Verilog is case-delicate, has a fundamental preprocessor (however less advanced than that of ANSI C/C++), and proportional control stream watchwords (if/else, for, while, case, and so on.), and perfect administrator priority. Syntactic contrasts incorporate variable affirmation (Verilog requires bit-widths on net/regtypes[clarification needed]), boundary of procedural squares (start/end rather than wavy props {}), and numerous other minor contrasts.

3. SIMULATION RESULTS

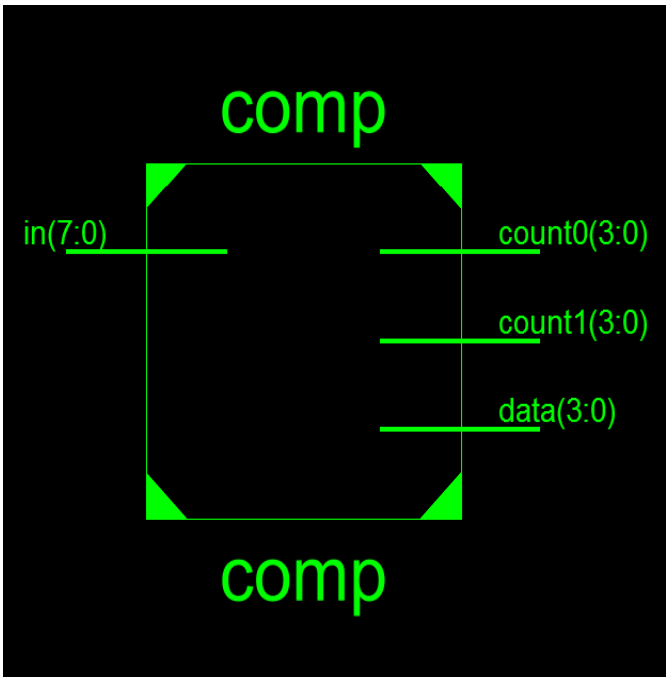

Fig:-3 De-Compressor Block Diagram

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

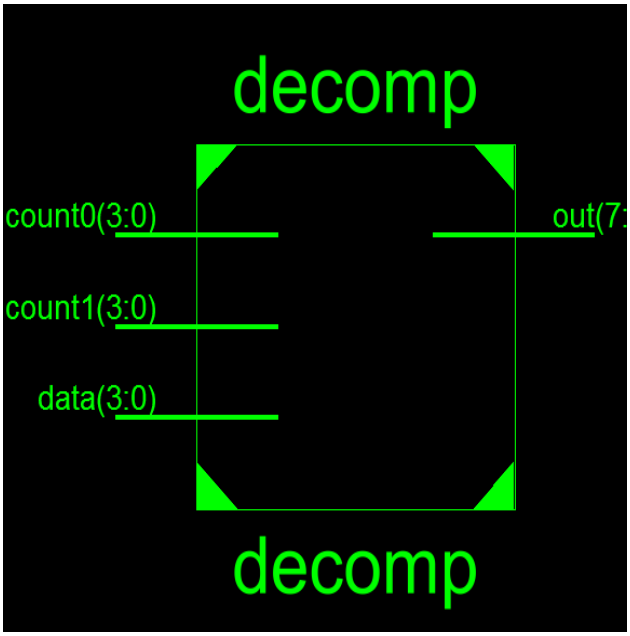

Fig:-4 De-Compressor Block Diagram

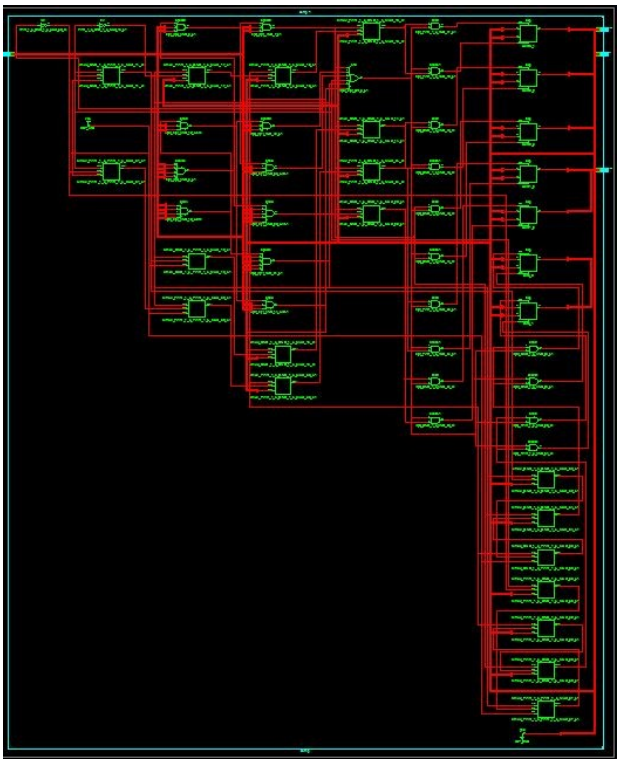

Fig:-5Compressor RTL

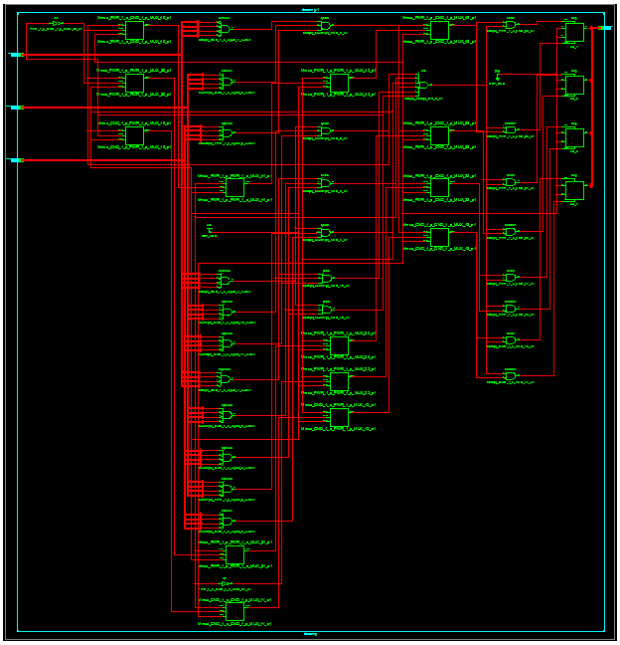

Fig:-6De-Compressor RTL

4. CONCLUSION

The existing compression algorithms either provide good compression with slow decompression or fast decompression at the cost of compression efficiency. In this paper, we

proposed a decoding-aware compression technique that tries to obtain both best possible compression and fast decompression performance. The proposed compression technique analyzes the effect of parameters on compression ratio and chooses the optimal ones automatically. We also exploit run length encoding of consecutive repetitive patterns efficiently combined with bitmask-based compression

to further improve both compression ratio and decompression efficiency.

5. REFERENCES

[1] J. H. Pan, T. Mitra, and W. F. Wong, "Configuration bit-stream compression for dynamically reconfigurable FPGAs," in Proc. Int. Conf. Comput.-Aided Des., 2004, pp. 766–773.

[2] S. Hauck and W. D. Wilson, "Run length compression techniques for FPGA configurations," in Proc. IEEE Symp.Field-Program.Custom Comput. Mach., 1999, pp. 286–287.

[3] A. Dandalis and V. K. Prasanna, "Configuration compression for FPGAbased embedded systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 12, pp. 1394–1398, Dec. 2005.

[4] D. Koch, C. Beckhoff, and J. Teich, "Bit-stream decompression for high speed FPGA configuration from slow memories," in Proc. Int. Conf. Field-Program. Technol., 2007, pp. 161–168.

[5] S. Seong and P. Mishra, "Bitmask-based code compression for embedded systems," IEEE Trans. Comput.-Aided Des.Integr. Circuits Syst., vol. 27, no. 4, pp. 673–685, Apr. 2008.

[6] S. Hauck, Z. Li, and E. Schwabe, "Configuration compression for the Xilinx XC6200 FPGA," IEEE Trans. Comput.-Aided Des.Integr. Circuits Syst., vol. 18, no. 8, pp. 1107–1113, Aug. 1999.

[7] D. A. Huffman, "A method for the construction of minimumredundancy codes," Proc. IRE, vol. 40, no. 9, pp. 1098–1101, 1952.

[8] A. Moffat, R. Neal, and I. H. Witten, "Arithmetic coding revisited," in Proc. Data Compression Conf., 1995, pp. 202–211.

[9] Xiaoke Qin, ChetanMuthry, and Prabhat Mishra, "Decoding Aware Compression of FPGA Bit-streams," in Proc. Data Compression Conf., 2011, pp. 411–419.

Authors Profile

I Nandeeswari was born in Kakinada, Andrapradesh on October 18, 1993. I graduated from the Sri SaiAditya Institute of Science and Technology (JNTU), Surempalem, Kakinada. Now I am pursuingM.Tech in Kakinada Institute of science and Technology, korangi.

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

Mr. S.DEVA KARUN was born Rajahmundry, AP, on FEBRUARY 02 1987. He graduated from the Jawaharlal Nehru Technological University, Kakinada, Post-graduated from the Jawaharlal Nehru Technological University Kakinada. Presently He is working as Asst Prof in Kakinada Institute of Engineering & Technology, Korangi. So far he is having 3 Years of Teaching Experience in various reputed engineering colleges. His special fields of interest included VLSI-Signal Processing, Embedded Systems.