

# Development of data compression block, using Huffman coding

AUTHORS:

JANIS DALBINS

GATIS GAIGALS

23.09.2014.

ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY  
EUROPEAN REGIONAL DEVELOPMENT FUND

# Introduction

---

ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY  
EUROPEAN REGIONAL DEVELOPMENT FUND

# Contents

---

Lossless data compression methods;

Huffman coding;

Development methods;

LabVIEW FPGA;

FPGA target device;

Development of UDCB;

Testbench;

Testing results;

Performance results;

Conclusions and suggestions.

# Lossless data compression methods

---

## Entropy type:

- Arithmetic;
- Huffman;
- Golomb;
- Range;
- Shannon-Fano;
- Fibonacci;
- etc.

## Dictionary type:

- DEFLATE;
- Lempel-Ziv;
- Run-length encoding;
- etc.

## Other types:

- Delta;
- Burrows–Wheeler;
- etc.

# Huffman coding

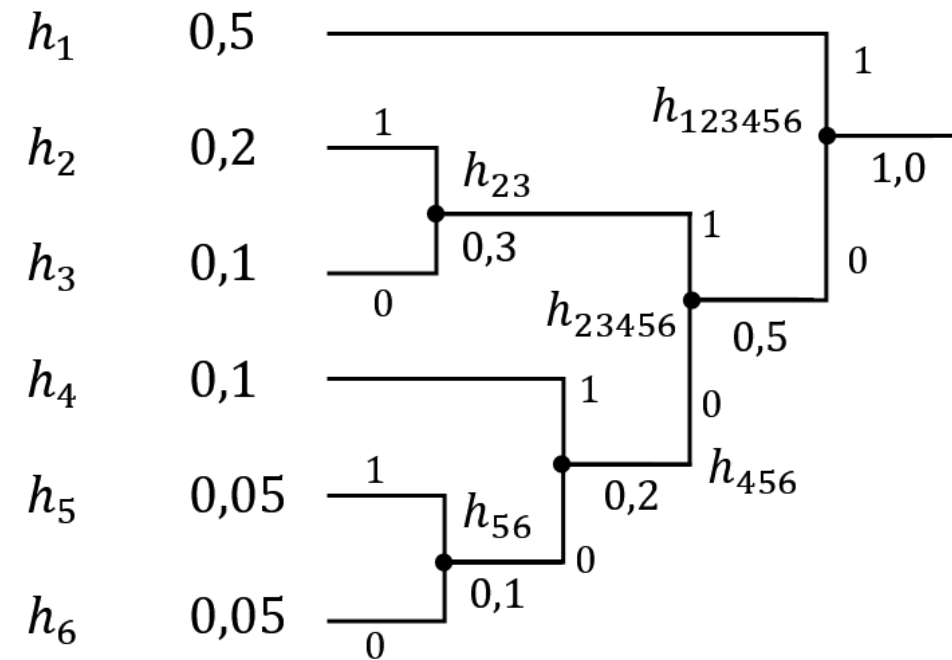
Huffman coding is lossless entropy compression algorithm;

Based on variable-length code table called Huffman tree;

Coding process can be divided in two parts:

- creation of Huffman tree;
- creation of code for each symbol in alphabet;

Universal data compression block (UDCB) is based on a particular type of Huffman coding called Canonical Huffman coding.



# Canonical Huffman coding

Canonical Huffman coding allows to describe Huffman tree;

Description table is created using one formula and simple «for» loop:

*for*  $f := 5$  *downto* 1 *do*  $first[f] :=$   
 $:= [(first[f + 1] + probability[f + 1]/2)]$   
 where  $first[6] := 0$

Nr.	Huffman code	Canonical Huffman code	Nr.	Huffman code	Canonical Huffman code
1	000	011	9	10100	01000
2	001	100	10	101010	000000
3	010	101	11	101011	000001
4	011	110	12	101100	000010
5	10000	00100	13	101101	000011
6	10001	00101	14	101110	000100
7	10010	00110	15	101111	000101
8	10011	00111	16	110000	000110

<b>Code length</b>	1	2	3	4	5	6
<b>Code “probability”</b>	0	0	4	0	5	7
<b>First number</b>	2	4	3	5	4	0

## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

# Development methods

---

Possible UDCB development methods:

- ASIC (Application-specific integrated circuit) or similar;
- Microcontrollers (Modified C or other languages);
- FPGA (VHDL, LabVIEW environment).

# LabVIEW FPGA

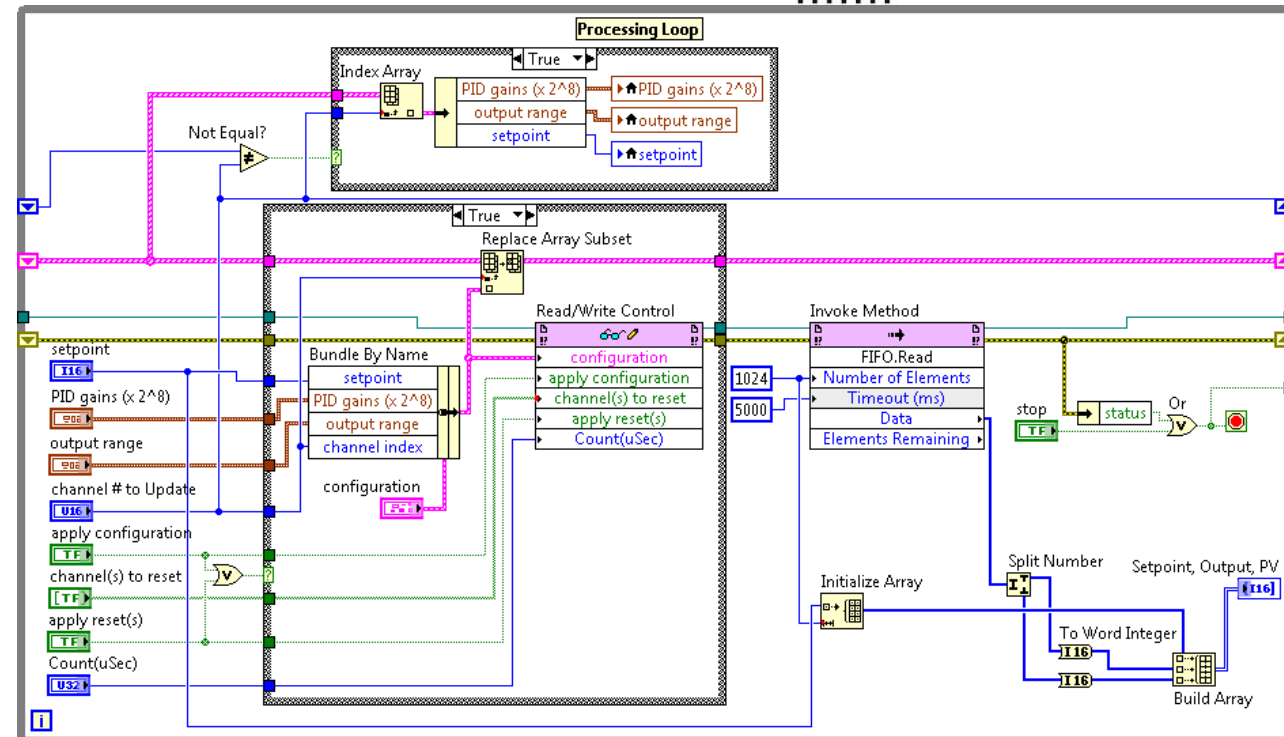
LabVIEW\* is a system-design platform and development environment;

LabVIEW FPGA is an extension module for LabVIEW to target FPGAs on NI\*\* reconfigurable I/O (RIO) hardware

Implements true parallelism for parallel data processing on FPGA chip in real-time

\*Laboratory Virtual Instrument Engineering Workbench

\*\*National Instruments



## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND



# FPGA target device

Xilinx Spartan-3E FPGA Family chip:

- System gates – 500K;
- Equivalent Logic Cells – 10 476;
- Distributed RAM bits – 73K;
- Block RAM bits – 360K;
- Dedicated multipliers – 20;
- User I/O – 232;



ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY  
EUROPEAN REGIONAL DEVELOPMENT FUND

# Development of UDCB

---

The goal of development is a true parallelism for high speed compression of large amount of data;

Possible to change the capacity of the buffer memory;

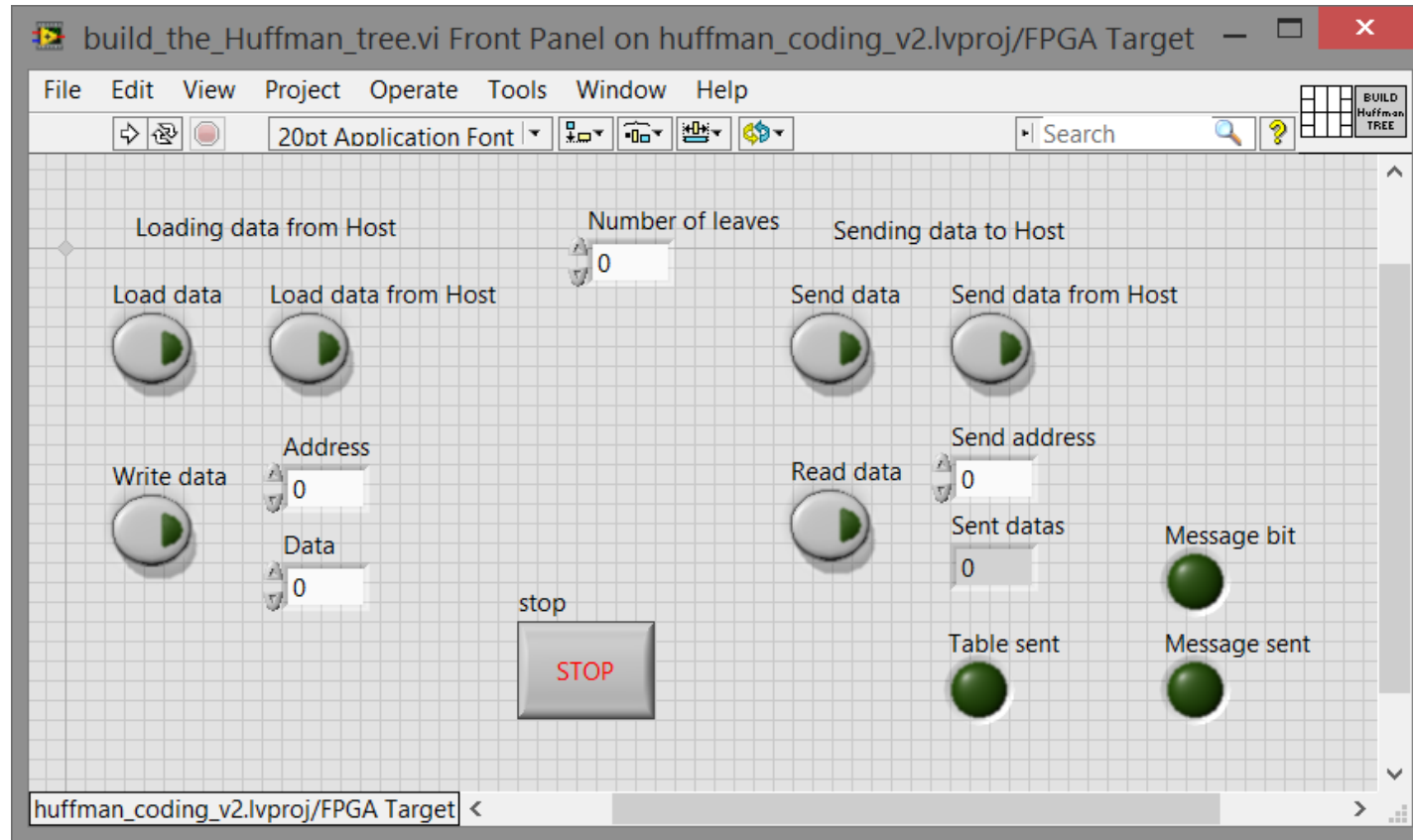
UDCB can compress 8 bit input data;

Despite of Huffman coding sequential nature, it is possible to do parallel data processing:

- Data reading;
- Compression of data;
- Data sending.

Structure of UDCB is divided in units which can be implemented in many ways (add new, edit existing)

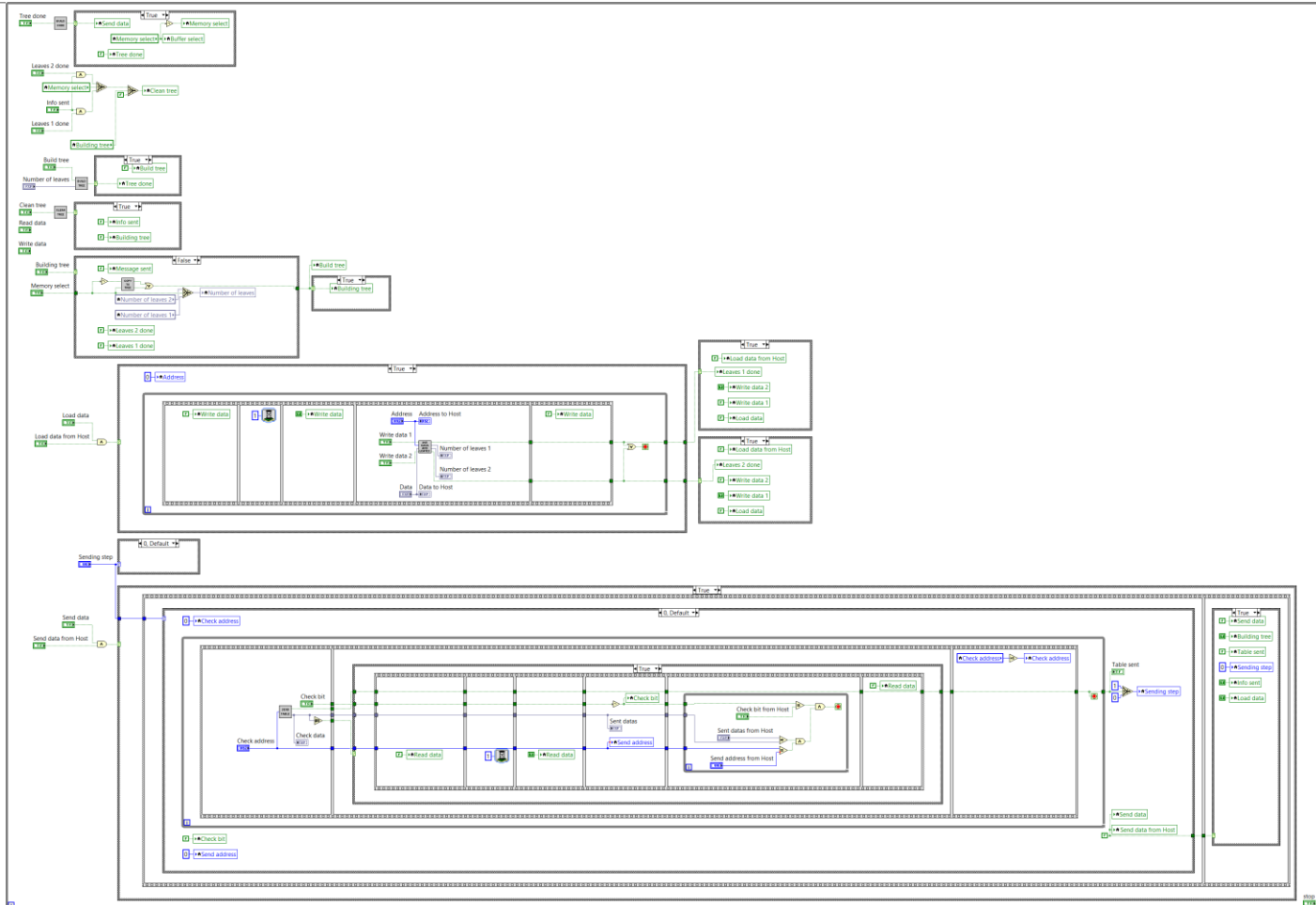
# Development of UDCB



## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

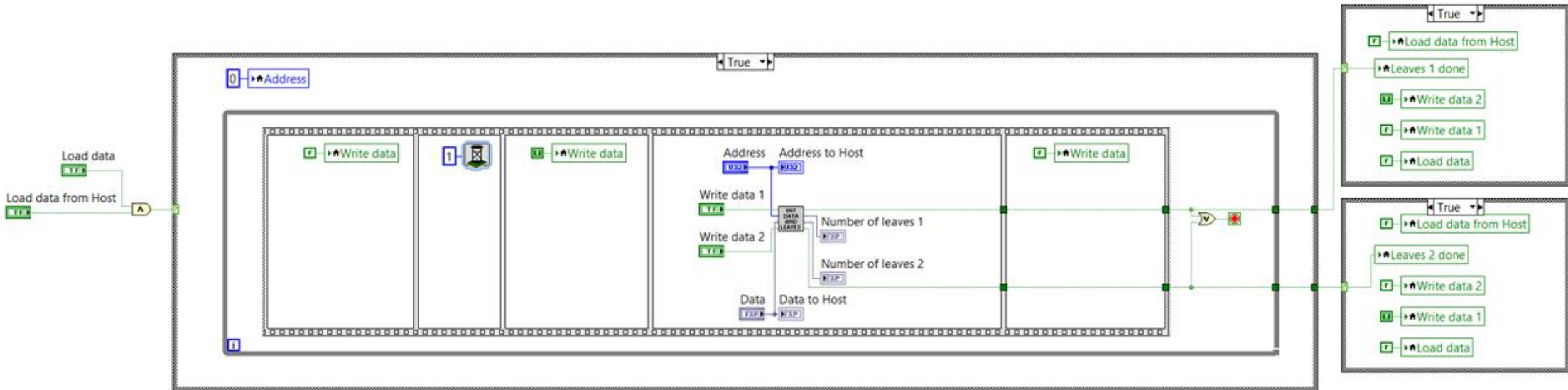
# Development of UDCB



## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

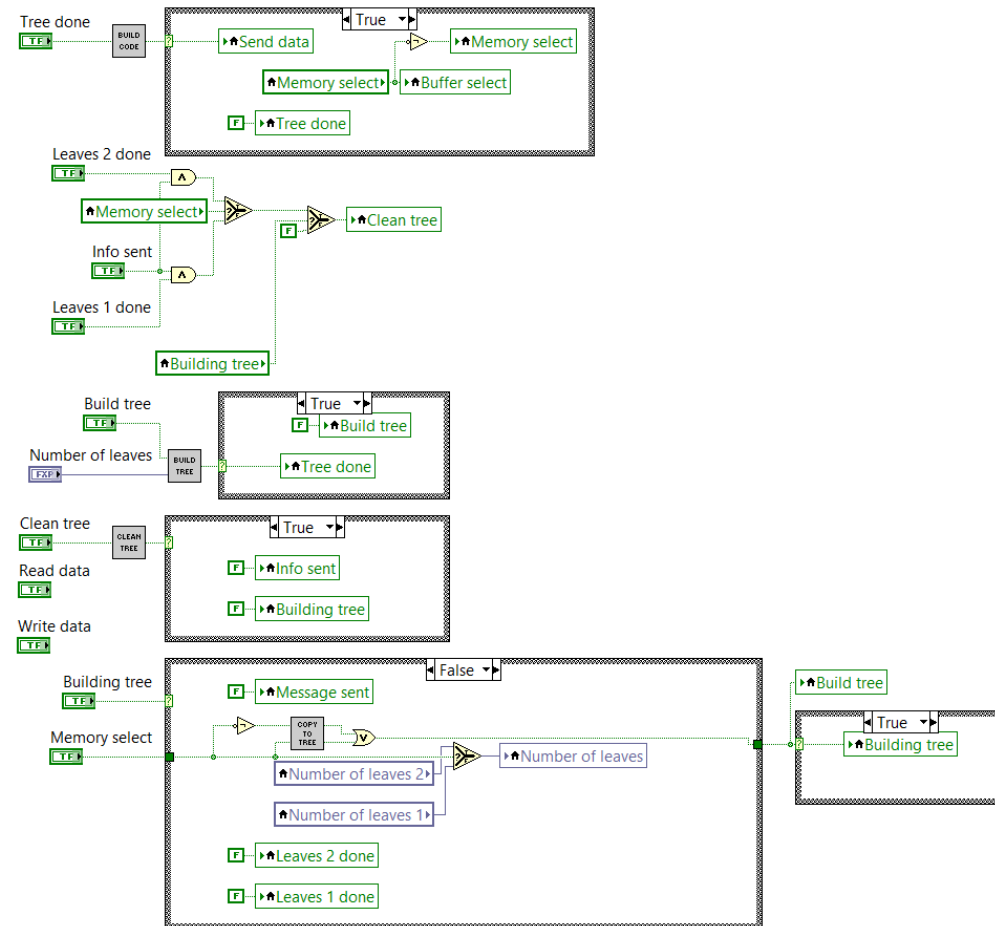
# Development of UDCB



## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

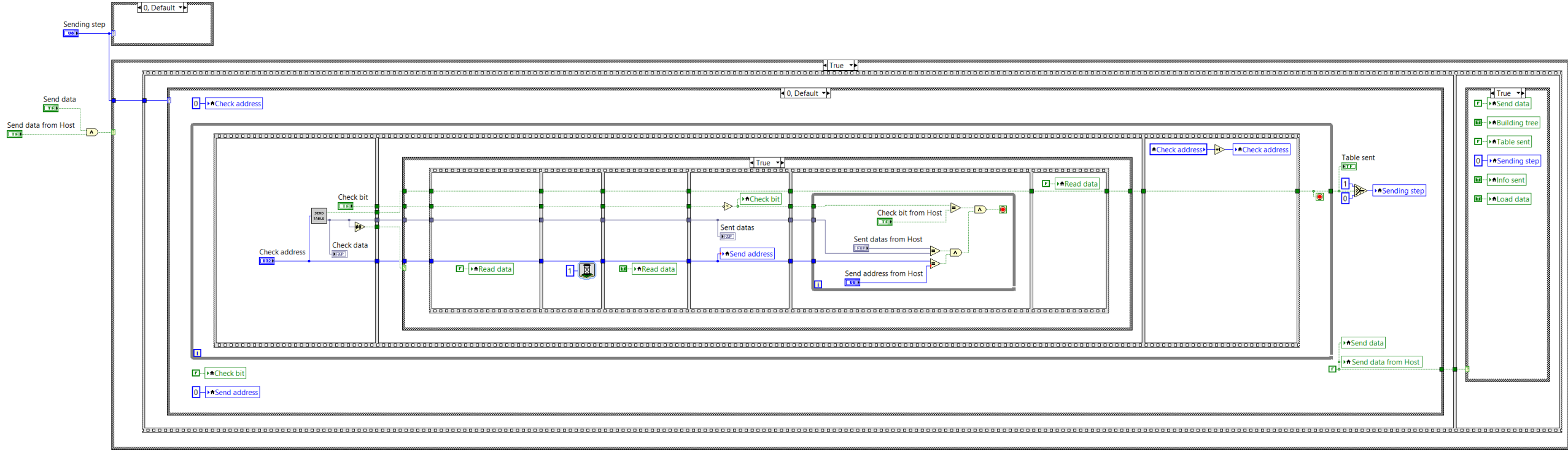
# Development of UDCB



## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

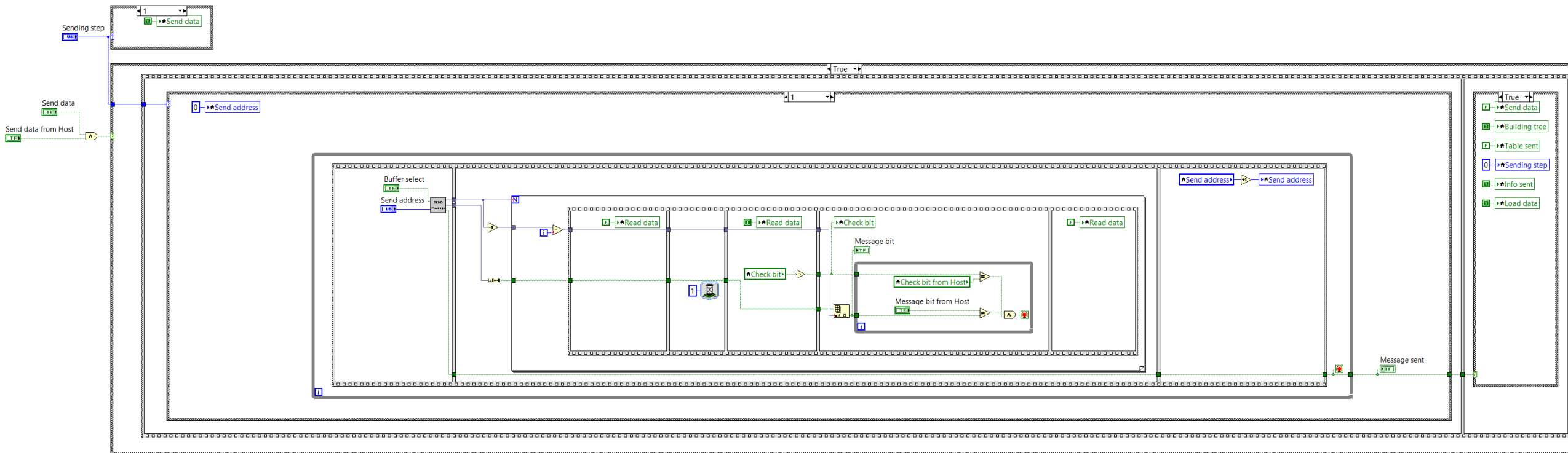
# Development of UDCB



## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

# Development of UDCB

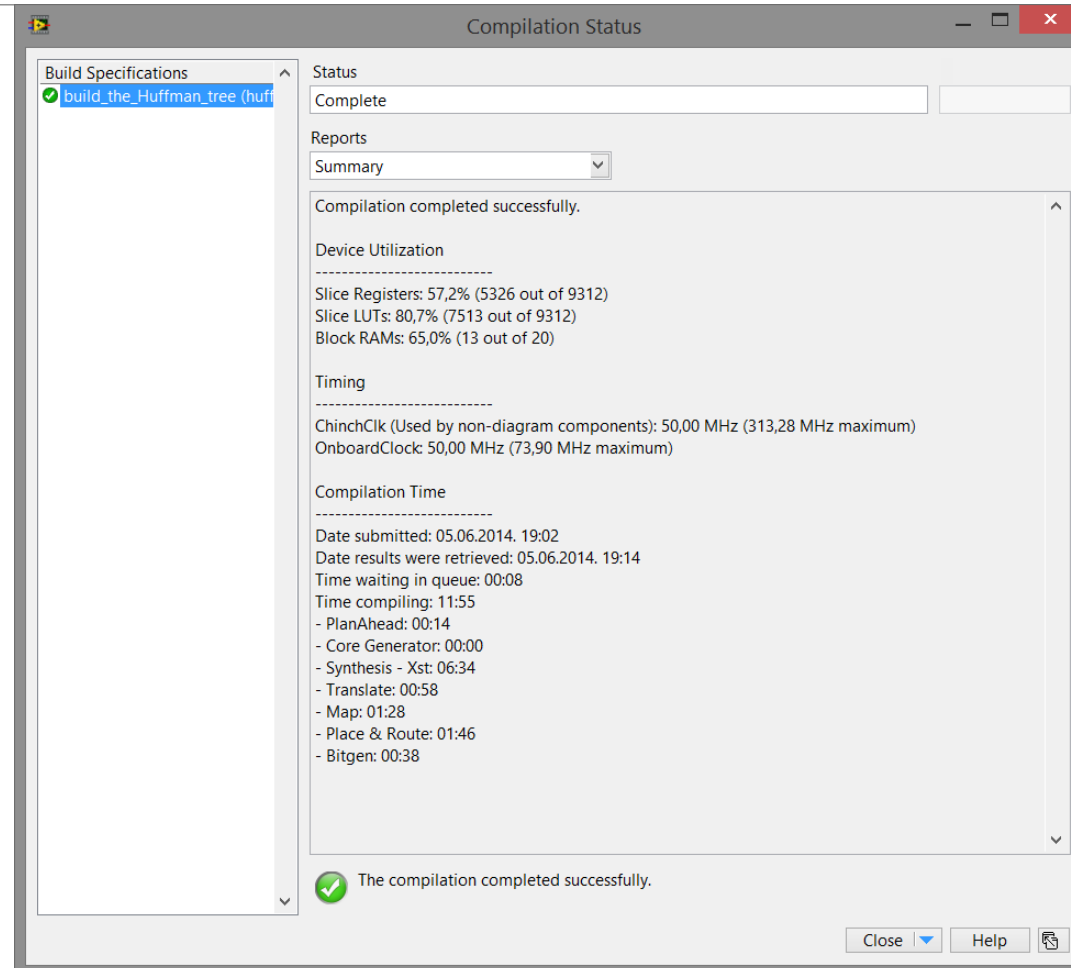


## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND



# Development of UDCB



## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

# Testbench

The screenshot shows a LabVIEW front panel titled "Test(Host).vi Front Panel on huffman\_coding\_v2.lvproj/My Computer". The interface is organized into several sections:

- Control:** Includes a "Case value" knob set to 1, a "Stop" button with a red "STOP" label, and an "error IO" section with a "status" indicator (a green checkmark) and a "code" field showing "0".
- Sending values:** Features an "Address" knob set to 256 and a "Generated values" list containing the numbers 72, 117, 102, 102, 109, 97, and 110.
- Receiving values:** Includes a "Send address" knob set to 121 and a "Sent datas" knob set to 8.
- Received table:** A 2x12 grid of numeric indicators, all currently showing 0.
- Received message:** A "Received message" knob set to 197, followed by a horizontal row of 24 green circular indicators.
- Tree path and Message path:** Two text fields displaying file paths: "% E:\Janks\Ventspils Augst...\1\test\_text\_tree.csv" and "% E:\Janks\Ventspils ...\1\test\_text\_message.csv".

The bottom status bar shows the project path: "huffman\_coding\_v2.lvproj/My Computer".

## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

# Testing results

Decode.vi Front Panel on huffman\_coding\_v2.lvproj/My Computer

File Edit View Project Operate Tools Window Help

20pt Application Font

Value case: 0

Number of leaves: 3

Generated values: 3, 3, 3, 3, 3, 3, 3, 3

Decoded message: 3, 3, 3, 3, 3, 3, 3, 3

Tree path: % E:\Janks\Ventspils Augst...\0\test\_text\_tree.csv

**Decoding table**

0	0	2	3	0	5	0	0
0	0	2	2	0	1	0	0

Message path: % E:\Janks\Ventspils ...\0\test\_text\_message.csv

Message: 0 1 0 1 0 1 0 1

Decoding table size in bits: 36

Message size in bits: 384

Compression correct:

Compression size in bits: 420

Compression ratio: 4,876

Entropy: 1,379

Average bit count: 1,5

Efficiency, %: 91,929

huffman\_coding\_v2.lvproj/My Computer

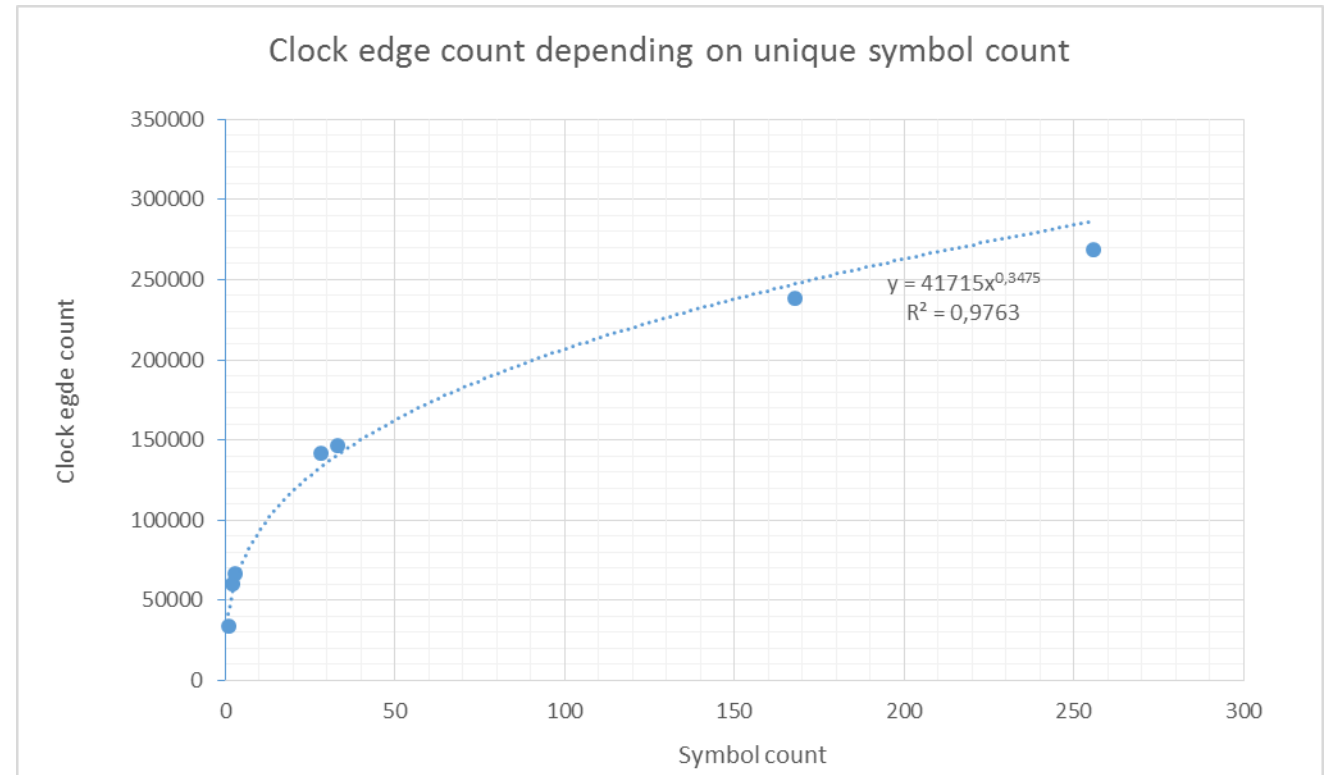
## ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY EUROPEAN REGIONAL DEVELOPMENT FUND

# Performance results

Testing for each case – 500 runs:

Symbol count	Average clock counts	Average time, ms
3	66809	1,34
28	141479	2,83
33	146498	2,93
168	238394	4,77
2	60354	1,21
1	33422	0,67
256	268958	5,38



ACKNOWLEDGEMENTS

PRESENTED RESEARCH AND DEVELOPMENT AS PART OF THE PROJECT NO. L-KC-11-0006 ARE FUNDED BY  
EUROPEAN REGIONAL DEVELOPMENT FUND

# Conclusions and suggestions

---

UDCB is developed successfully and it is compressing data correctly;

UDCB can be implemented in low power FPGA in data storing subsystem on nano-satellite CubeSAT type satellite;

Implement another Huffman coding type algorithms better performance comparing;

Implementation in other FPGA targets for more accurate and close to real-time testing;

Create adaptive table sending algorithm:

- Unique symbol count  $<86$  then send only coded symbols and their code length;
- Unique symbol count  $\geq 86$  then send whole table but only each symbol code length;

Upgrade UDCB to compress another length data (ex. 16 or 32 bit data);

Upgrade UDCB buffer memory for larger amount of compressible data (ex. 1024 or 2048 symbols).

*“The laws of probability, so true in general, so fallacious in particular.”*

*- Edward Gibbon*