INSTITUTE OF COMPUTER ENGINEERING AND ELECTRONICS
UNIVERSITY OF ZIELONA GÓRA
POLAND

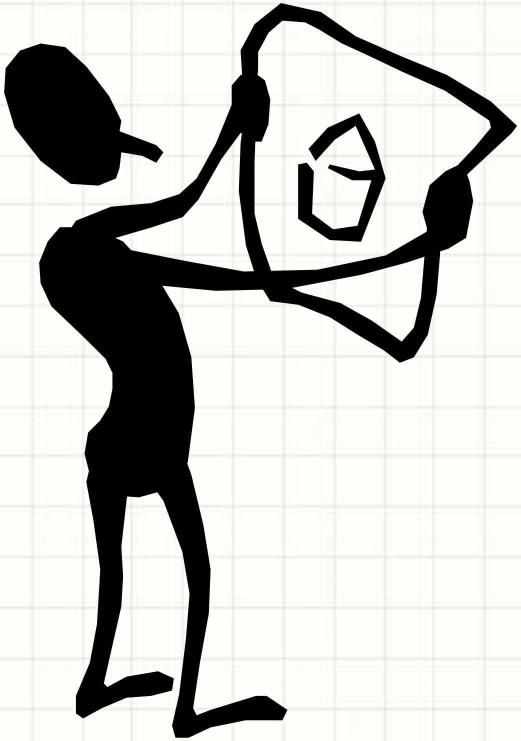
HARDWARE IMPLEMENTATION OF THE CLOUDBUS PROTOCOL USING FPGA

Kazimierz Krzywicki
Grzegorz Andrzejewski

PESW 2014, Prague, 12-13 June 2014

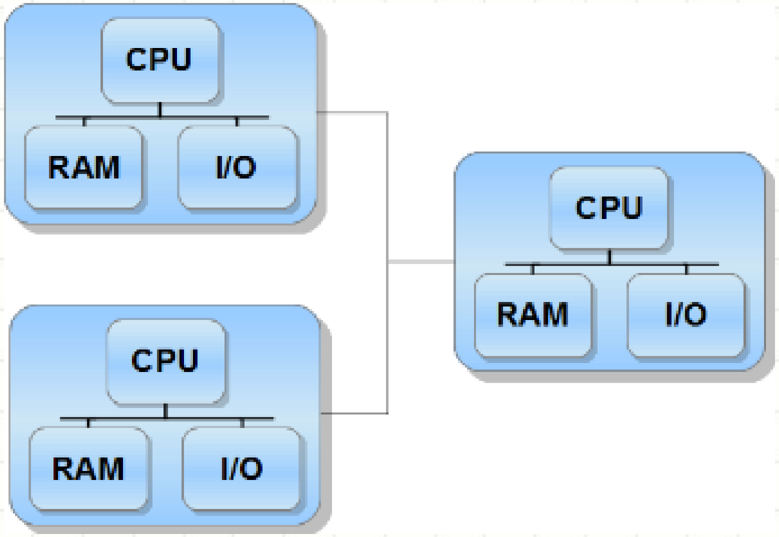
Agenda

1. **Introduction**
2. **CloudBus protocol**
3. **Implementation**
4. **Research results**
5. **Conclusions**



1. Introduction

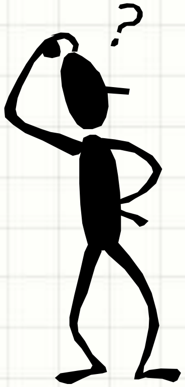
- Data exchange and process synchronization in embedded distributed systems (*CloudBus* protocol)
- Implementation and synthesis of the CloudBus protocol using Altera and Xilinx FPGA devices

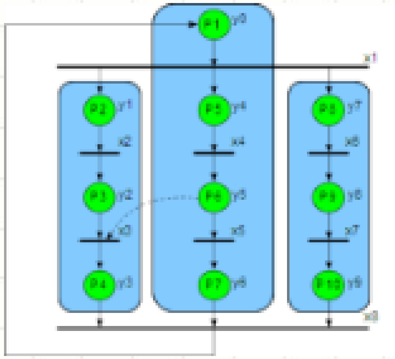


1. Introduction

Distributed embedded systems – main problems:

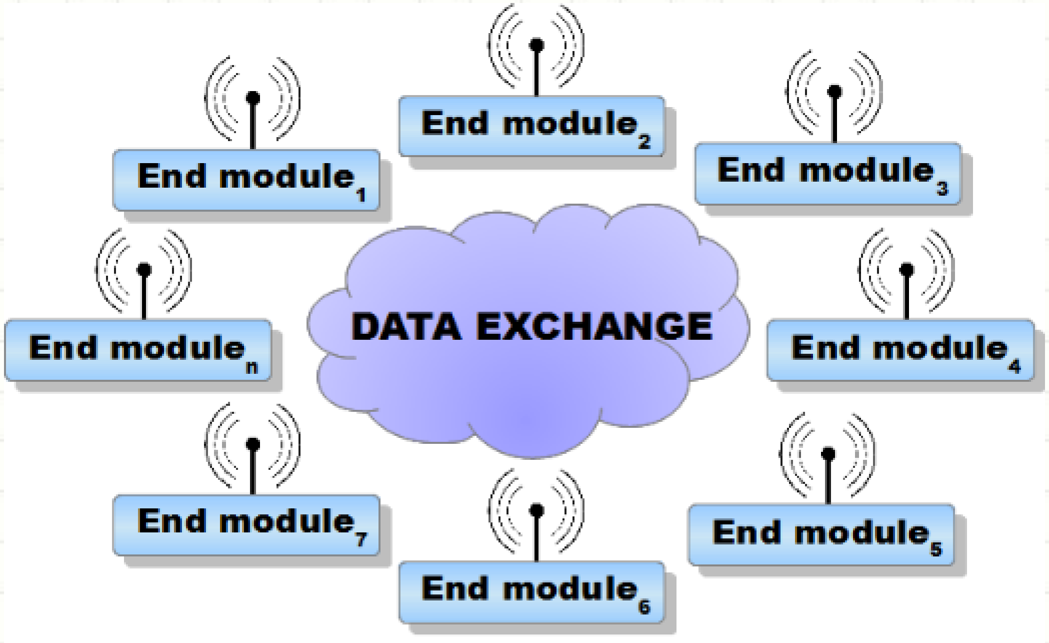
- Large and complex distributed embedded systems are difficult to design and implementation
- High load of the communication interfaces
- High cost of large PLC (Programmable Logic Controller) systems





1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

2. CloudBus protocol



Module sends (broadcasts) question about variable state (e.g. *if x1==1?*)

Module which is responsible for variable – answers when variable reach quested state

Questioner module gets information about variable state

- All end modules are equal to each other
- Each module is responsible for own part of the control algorithm
- Faster from a few to several ten of time when compared to *Modbus-RTU* or *Profibus-DP*

2. CloudBus protocol

CloudBus protocol frame:

CNT	FUNC	VARS	DATA	CRC

Fields of the protocol corresponds to:

CNT – 1 byte for entire frame length;

FUNC – command code (e.g. question about condition or simple answer to other of the modules);

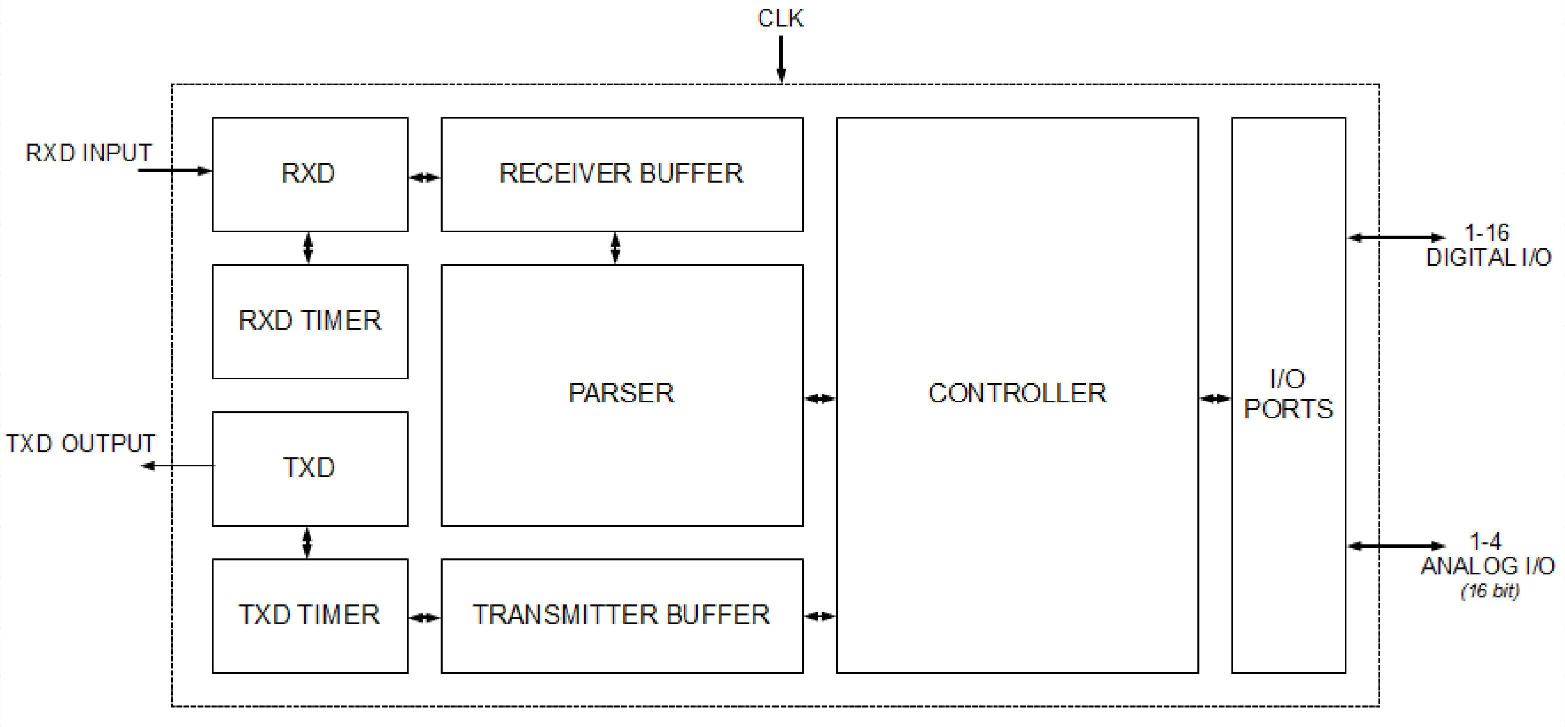
VARS and ***DATA*** – represents binary array of the variables and their states (values);

CRC – 1 byte of the CRC error checksum.

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

3. Implementation

Schematic diagram of the end module unit:



End module features:

- 16 digital inputs/outputs
- 4 analog inputs/outputs (16 bit)
- serial communication interface
- external clock input

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

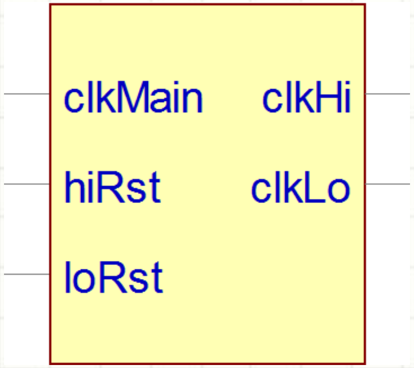
3. Implementation

RXD TIMER module:

RXD TIMER module is double timer/counter.

INPUTS:

- ***clkMain*** – external CLK clock input
- ***hiRst*** and ***loRst*** – counter reset



OUTPUTS:

- ***clkHi*** – high frequency clock
- ***clkLo*** – *low frequency clock*

Operating principle:

clkHi counter is used for the sampling receiving RXD INPUT line,

clkHi – used for the bit read from receiving RXD INPUT line

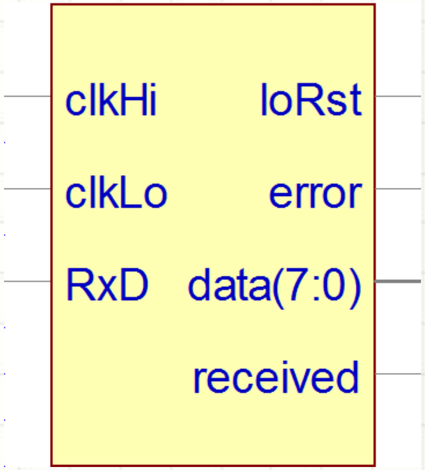
3. Implementation

RXD module:

RXD module is RXD INPUT line data reciver.

INPUTS:

- ***clkHi*** – high freq. counter for sampling RxD line
- ***clkLo*** – low freq. counter for reading from the RxD line
- ***RxD*** – receiving data line



OUTPUTS:

- ***loRst*** – clock reset
- ***error*** – received data corruption
- ***data*** – received byte
- ***received*** – data successful received

Operating principle:

It retrieves data from RXD INPUT by input ***RxD*** line. Input line is sampled with ***clkHi*** clock. When it detects incoming data, ***clkLo*** clock counter is started for data read from ***RxD*** line. Data bits are saved to ***data*** register output.

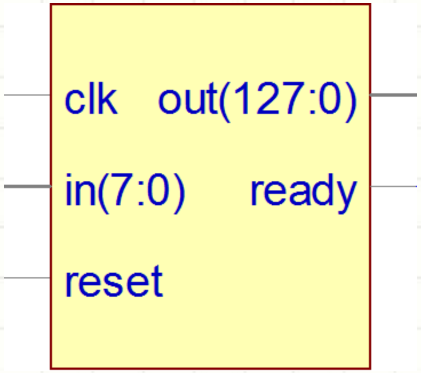
3. Implementation

RECEIVER BUFFER module:

RECEIVER BUFFER module is received data buffer.

INPUTS:

- ***clk*** – clock input
- ***in*** – data byte input
- ***reset*** – buffer reset



OUTPUTS:

- ***out*** – buffered data frame
- ***ready*** – buffer full/data ready

Operating principle:

RECEIVER BUFFER module is 128-bit data buffer for received data by RXD module. It merges all single bytes to entire frame of the CloudBus protocol. Module is synchronized by *clk* clock.

UNIVERSITY OF ZIELONA GÓRA

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

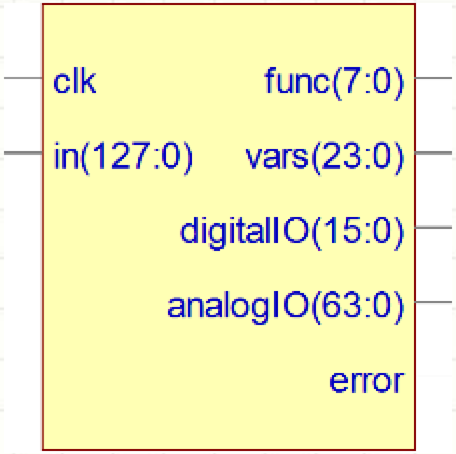
3. Implementation

PARSER module:

PARSER module is parsing CloudBus data frame from the RECEIVER BUFFER.

INPUTS:

- ***clk*** – clock input
- ***in*** *– 128-bit data frame input*



OUTPUTS:

- ***func*** – FUNC field of the CloudBus protocol
- ***vars*** – VARS field of the CloudBus protocol
- ***digitalIO*** and ***analogIO*** – varaibles state
- ***error*** – received frame corruption

Operating principle:

Data are received from RECEIVER BUFFER by 128-bit *in* input. PARSRER module is responsible for parsing received frame of the CloudBus protocol from RECEIVER BUFFER. When parsing is done valid CloudBus data are set to outputs: *func*, *vars*, *digitalIO*, *analogIO* else *error* output is driven high.

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

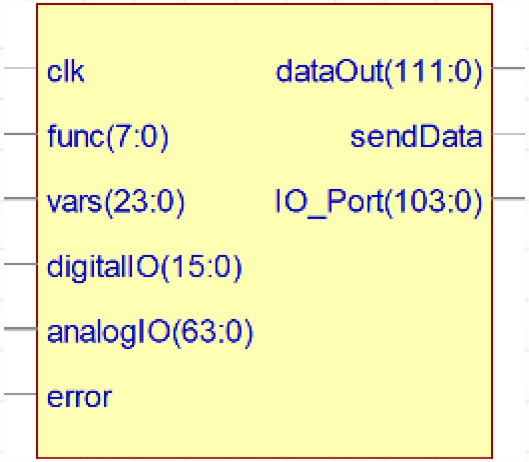
3. Implementation

CONTROLLER module:

CONTROLLER module is the main operating module.

INPUTS:

- ***clk*** – clock input
- ***func*** – FUNC data input
- ***vars*** – VARS data input
- ***digitalIO*** and ***analogIO*** – varaibles state
- ***error*** – parsing error input



OUTPUTS:

- ***dataOut*** – data to transmit
- ***sendData*** – data ready to send
- ***IO_Port*** – external I/O data port

Operating principle:

It is responsible for implementing designed control algorithm and for the communication with other end modules via CloudBus protocol.

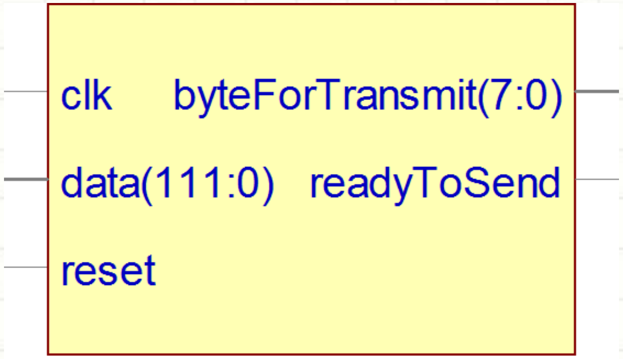
3. Implementation

TRANSMITTER BUFFER module:

TRANSMITTER BUFFER module is transmitter data buffer.

INPUTS:

- ***clk*** – clock input
- ***data*** – CloudBus data frame to send
- ***reset*** – buffer reset



OUTPUTS:

- ***byteForTransmit*** – byte to send
- ***readyToSend*** – data ready to send

Operating principle:

TRANSMITTER BUFFER module is preparing data and encoding entire frame for TXD module. Module also counts frame length and CRC checksum of CloudBus protocol frame.

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

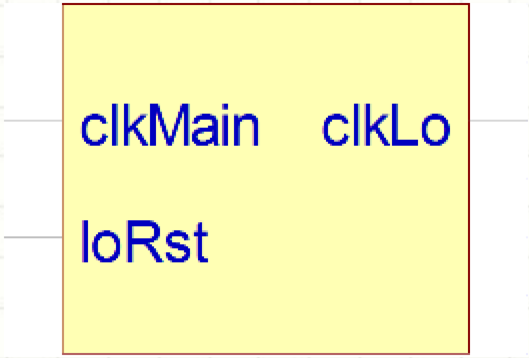
3. Implementation

TXD TIMER module:

TXD TIMER module is sending clock generator for the TXD module

INPUTS:

- ***clkMain*** – external CLK clock input
- ***loRst*** – clock reset



OUTPUTS:

clkLo – data transmit clock

Operating principle:
Module uses external CLK clock for generating transmitting clock (*clkLo*) for the TXD module.

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

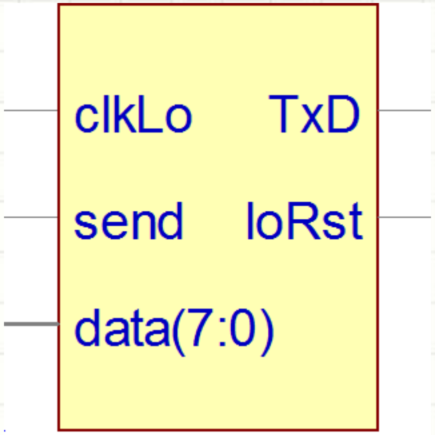
3. Implementation

TXD module:

TXD module is TXD OUTPUT line data sender.

INPUTS:

- ***clkLo*** – clock for data sending
- ***send*** – send data
- ***data*** – byte to send



OUTPUTS:

- ***TxD*** – external transmitting line
- ***loRst*** – sending timer reset

Operating principle:

TXD module sends data outside end module via *TxD* output line. *clkLo* is sending clock, *send* – starts byte transmission from *data* input.

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

4. Research results

The implementation and the synthesis

- Verilog HDL language – same source code for all platforms
- Xilinx ISE Design Suite 14.7:
  - Kintex-7 (XC7K70T);
  - Spartan 3E (XC3S1600E);
  - Virtex-6 (XC6VLX75T)
- Quartus II 13.1:
  - Arria II GX (EP2AGX45DF29C4);
  - Cyclone IV E (EP4CE 115F23I8L);
  - Cyclone V (5CGXFC7D7F27C8).

4. Research results

Xilinx ISE Design Suite 14.7 research results

Implemented modules	Device								
	Kintex-7 (XC7K70T)			Spartan 3E (XC3S1600E)			Virtex-6 (XC6VLX75T)		
	Reg.	Slices	LUTs	Reg.	Slices	LUTs	Reg.	Slices	LUTs
RXD TIMER	21	14	41	21	17	29	21	14	47
RXD	31	14	28	32	33	41	31	13	28
RECEIVER BUFFER	266	93	289	266	199	264	265	127	289
PARSER	112	170	364	0	37	46	0	18	28
CONTROLLER	0	1	2	0	2	3	0	1	2
TRANSMITTER BUFFER	9	2	8	9	6	9	9	3	8
TXD TIMER	12	8	29	12	10	16	12	11	29
TXD	8	9	15	8	12	19	8	7	17
Total used	459	311	776	348	316	427	346	194	448
Total available	82000	10250	41000	29504	14752	29504	93120	11640	46560
Usage [%]	**0,56**	**3,03**	**1,89**	**1,18**	**2,14**	**1,45**	**0,37**	**1,67**	**0,96**

Legend:

- RED – maximum value
- GREEN – minimum value

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

4. Research results

Quartus II 13.1 research results

Modules	Device					
	Arria II GX (EP2AGX 45DF29C4)		Cyclone IV E (EP4CE 115F23I8L)		Cyclone V (5CGXF C7D7F27C8)	
	Reg.	LUTs	Reg.	LUTs	Reg.	ALMs
RXD TIMER	21	28	21	28	21	18
RXD	36	35	32	47	36	23
RECEIVER BUFFER	262	152	262	275	263	140
PARSER	113	32	113	155	113	73
CONTROLLER	3	2	3	3	3	2
TRANSMITTER BUFFER	9	2	9	9	9	5
TXD TIMER	12	16	12	16	12	11
TXD	8	15	8	17	8	10
Total used	464	282	460	550	465	282
Total available	36100	36100	114480	114480	225920	56480
Usage [%]	**1,29**	**0,78**	**0,40**	**0,48**	**0,21**	**0,50**

Legend:

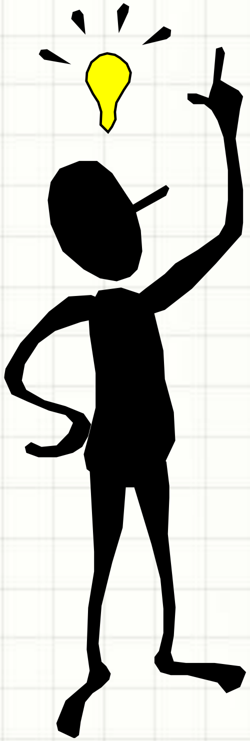
- RED – maximum value
- GREEN – minimum value

UNIVERSITY OF ZIELONA GÓRA

1. Introduction
2. CloudBus protocol
3. Implementation
4. Research results
5. Conclusions

5. Conclusions

- Total avarge resource for both platforms usage is about 1-2%
- Average resource usage for the:
  - Xilinx devices:
    - used registers is 0,7%
    - occupied slices is 2,28%
    - used LUTs is 1,43%.
  - Altera devices:
    - used registers is 0,63%
    - used LUTs is 0,59%.



5. Conclusions

- Implementing CloudBus protocol on the FPGA platform gives negligibly small resource usage
- CloudBus protocol almost dosen’t limit the implementation of the other control algorithms

Further research:

- Implementation and comparison performance of the real working industry distributed embedded systems in the term of protcol type
- further development of the CloudBus protocol

INSTITUTE OF COMPUTER ENGINEERING AND ELECTRONICS
UNIVERSITY OF ZIELONA GÓRA
POLAND

THANK YOU FOR YOUR ATTENTION!

HARDWARE IMPLEMENTATION OF THE CLOUDBUS PROTOCOL USING FPGA

Kazimierz Krzywicki
Grzegorz Andrzejewski

PESW 2014, Prague, 12-13 June 2014