



**The Abdus Salam
International Centre for Theoretical Physics**



2065-D4304

**Advanced Training Course on FPGA Design and VHDL for Hardware
Simulation and Synthesis**

26 October - 20 November, 2009

Presentations by Participants

National University of San Luis

San Luis, Argentina

Research and Electronic Services Lab

(Laboratorio de Electrónica, Investigación y Servicios)

L.E.I.S.

Topics:

- FPGA Platform development for Image and Video Processing
- Virtual Instrumentation (RVI)

Andrés Miguel Airabella



Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

- Objectives
- Results
- Drawbacks
- Staff
- What's next?

Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

Objectives

- Prepare a platform capable of Image and Video processing tests, based on parallel algorithms.
- The researcher doesn't need to develop the entire platform.
- The image or video researcher should be able of implement an algorithm in hardware, test it and compare the results with a computer-implemented version.

Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

Objectives

Prepare a platform capable of Image and Video processing tests, based on parallel algorithms

The platform...

... must fit in the Xilinx Multimedia Board FPGA.

... must use the on-board hardware.

... all the available IP cores and modules can, but not necessarily must, be used.

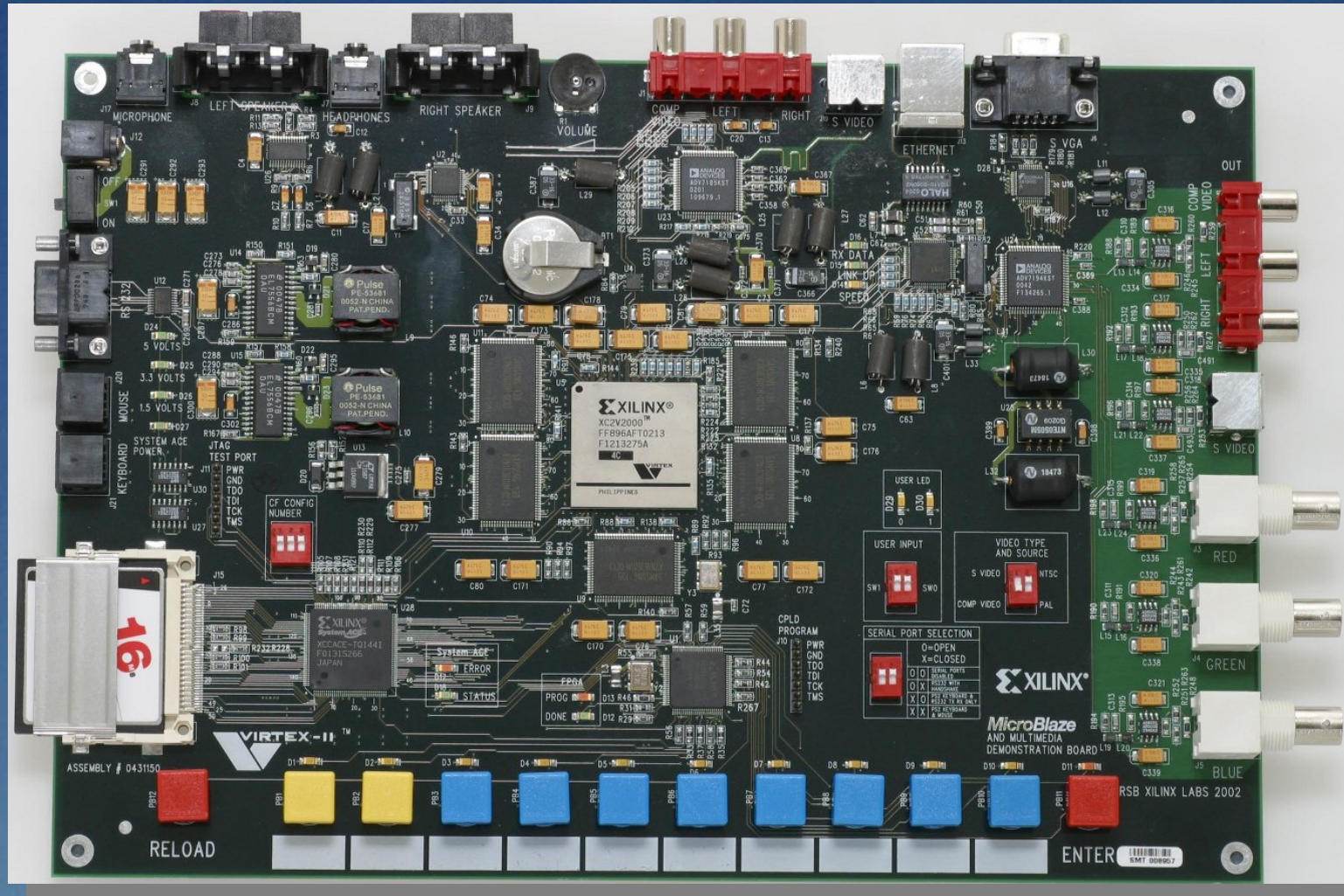
Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing Objectives



Andrés Miguel Airabella



Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing Objectives

The board...

... Video decoder ADV7185.

... VGA DAC FMS3810.

... Memory ZBT RAM K7N163601M.

... FPGA Xilinx Virtex II XC2V2000-6FF896.

Andrés Miguel Airabella



Universidad Nacional de San Luis

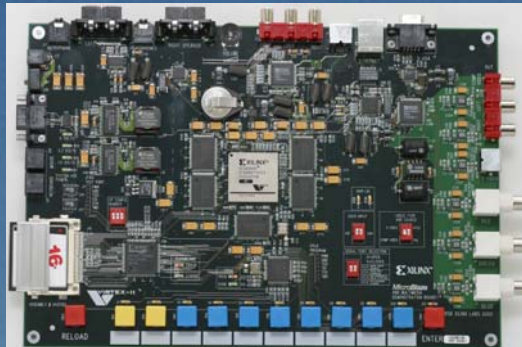


FPGA Platform development for Image and Video Processing

Objectives



⋮



Andrés Miguel Airabella



Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

- Objectives
- Results
- Drawbacks
- Staff
- What's next?

Andrés Miguel Airabella

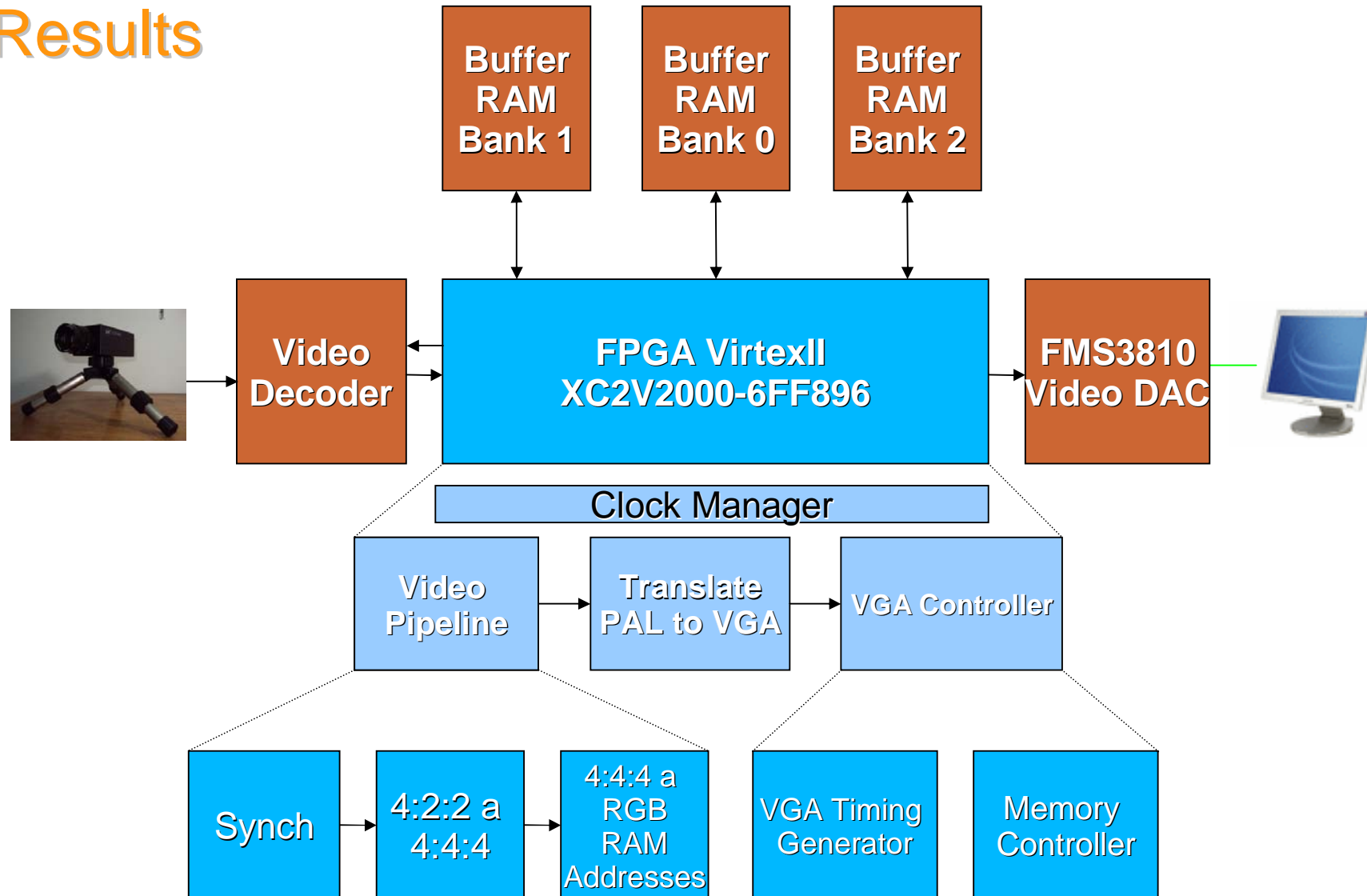


Universidad Nacional de San Luis



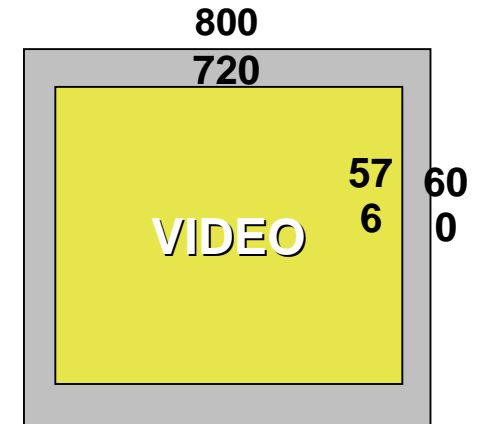
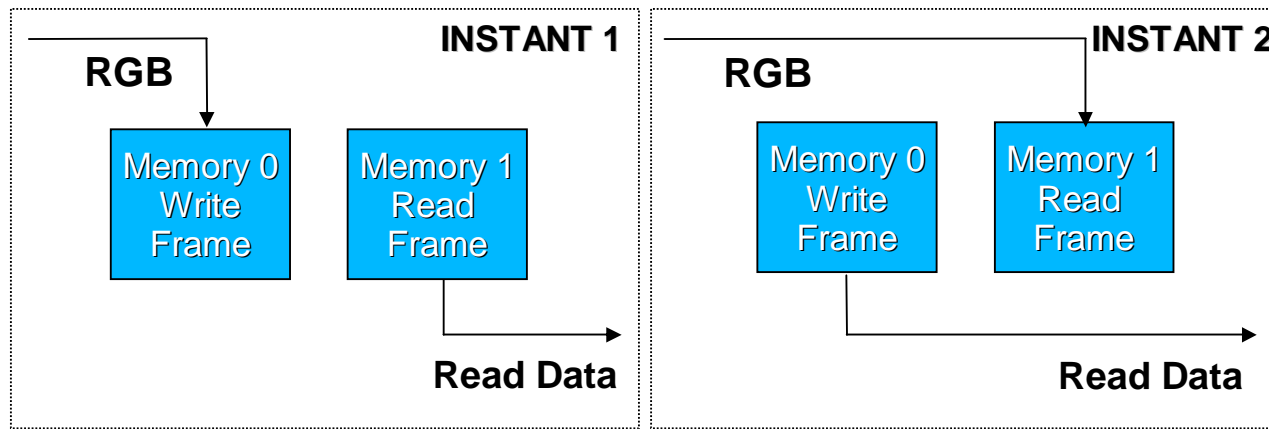
FPGA Platform development for Image and Video Processing

Results



FPGA Platform development for Image and Video Processing

Results



FPGA Platform development for Image and Video Processing

```
entity TRANSLATE is
  Port ( CLK           : in      STD_LOGIC;
         RESET        : in      STD_LOGIC;
         PAL_MAX_ADDRESS : in      STD_LOGIC_VECTOR (18 downto 0);

         USER_ACCESS_OK : in      STD_LOGIC;

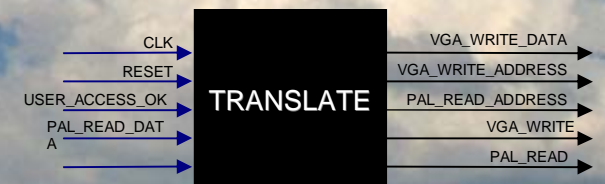
         PAL_READ_DATA   : in      STD_LOGIC_VECTOR (31 downto 0);
         VGA_WRITE_DATA  : out     STD_LOGIC_VECTOR (31 downto 0);

         PAL_READ_ADDRESS : inout   STD_LOGIC_VECTOR (18 downto 0);
         VGA_WRITE_ADDRESS : inout   STD_LOGIC_VECTOR (18 downto 0);

         PAL_READ        : inout   STD_LOGIC;
         VGA_WRITE       : out     STD_LOGIC
        );
end TRANSLATE;
```

```
architecture TRANSLATE_ARCH of TRANSLATE is
begin
  -- User Code

end architecture;
```



Andrés Miguel Airabella

FPGA Platform development for Image and Video Processing

Results

By-Pass, Basic EQ, RGB2GS conversion and GS2BW has been performed.



Andrés Miguel Airabella



Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

- Objectives
- Results
- Drawbacks
- Staff
- What's next?

Andrés Miguel Airabella



Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

Drawbacks

- Virtex II is obsolete.
- Due to the FPGA discontinuity, old software versions must be used. Xilinx ISE 9.1 or 9.2.
- Clock distribution in the board is too bad!
- The board programming methods are a Parallel Port JTAG cable (too old!) and a CompactFlash.
- The video inputs and outputs are Analog.

Andrés Miguel Airabella



Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

- Objectives
- Results
- Drawbacks
- Staff
- What's next?

Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing Staff

Andres Miguel Airabella

Carlos Federico Sosa Paez

Ricardo Petrino

Andrés Miguel Airabella



Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

- Objectives
- Results
- Drawbacks
- Staff
- What's next?

Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis



FPGA Platform development for Image and Video Processing

What's next?

- Test more complex algorithms, that involves processing more than one pixel at each time.
- Test image processing in control applications, but using the platform.
- Develop a Firewire Interface for new digital video cameras.

Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis



National University of San Luis

San Luis, Argentina

Research and Electronic Services Lab

(Laboratorio de Electrónica, Investigación y Servicios)

L.E.I.S.

Topics:

- FPGA Platform development for Image and Video Processing
- Virtual Instrumentation (RVI)

Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis

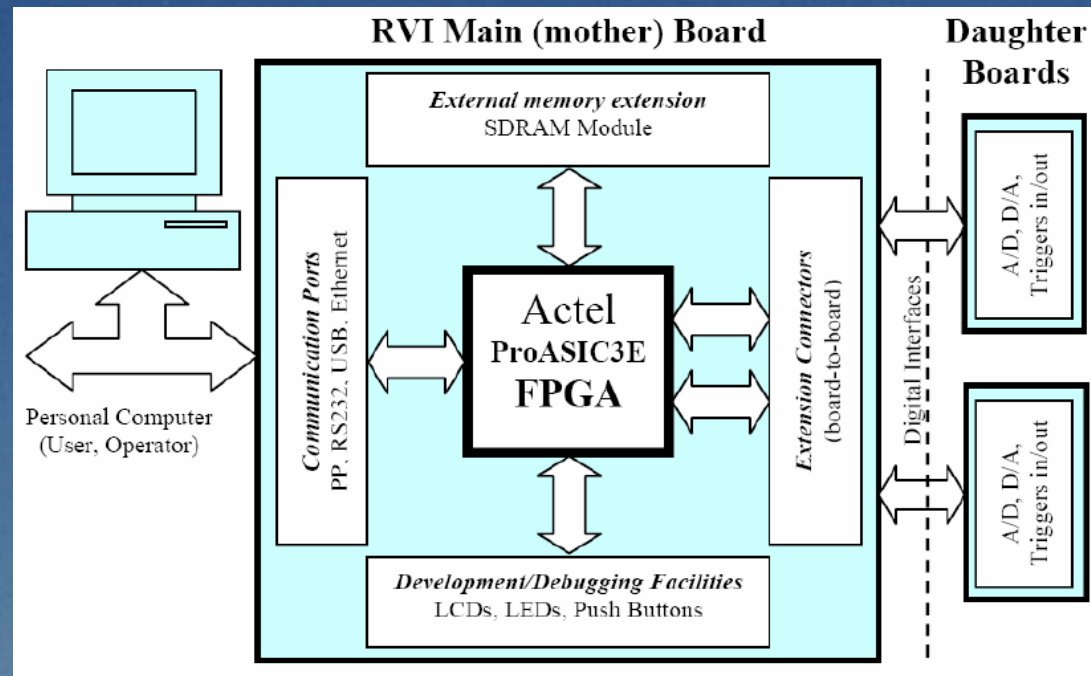


Virtual Instrumentation (RVI)

Objectives

Design a digital oscilloscope using the FPGA RVI Prototype Board and the LP Data Conversion Daughter Board, provided by ICTP.

- Develop the Hardware
- Develop the Software



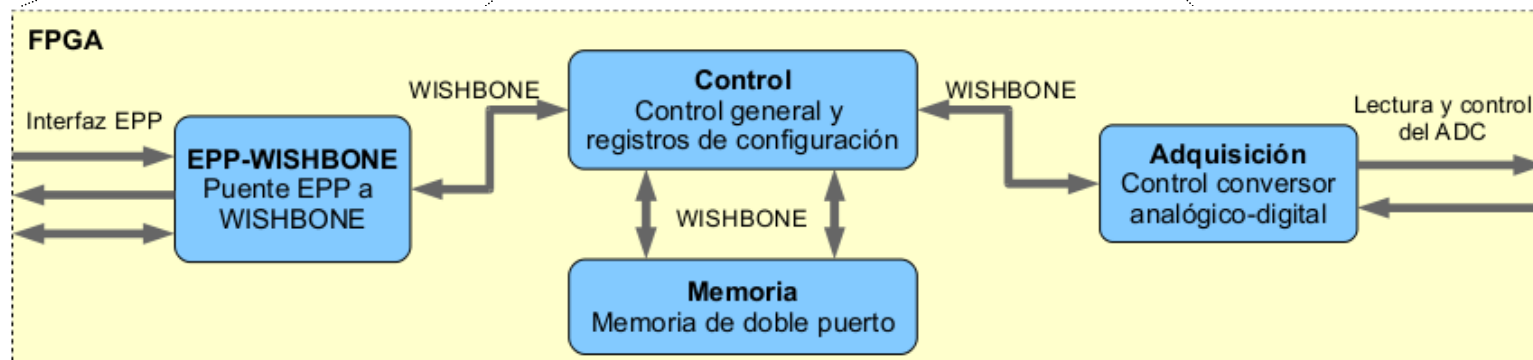
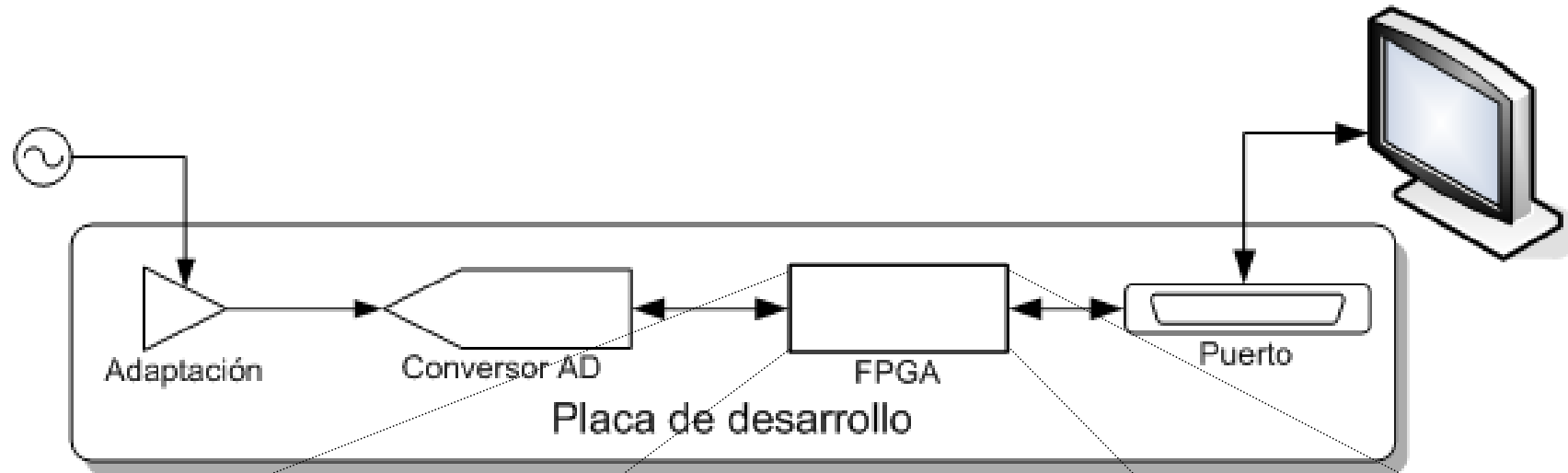
Andrés Miguel Airabella



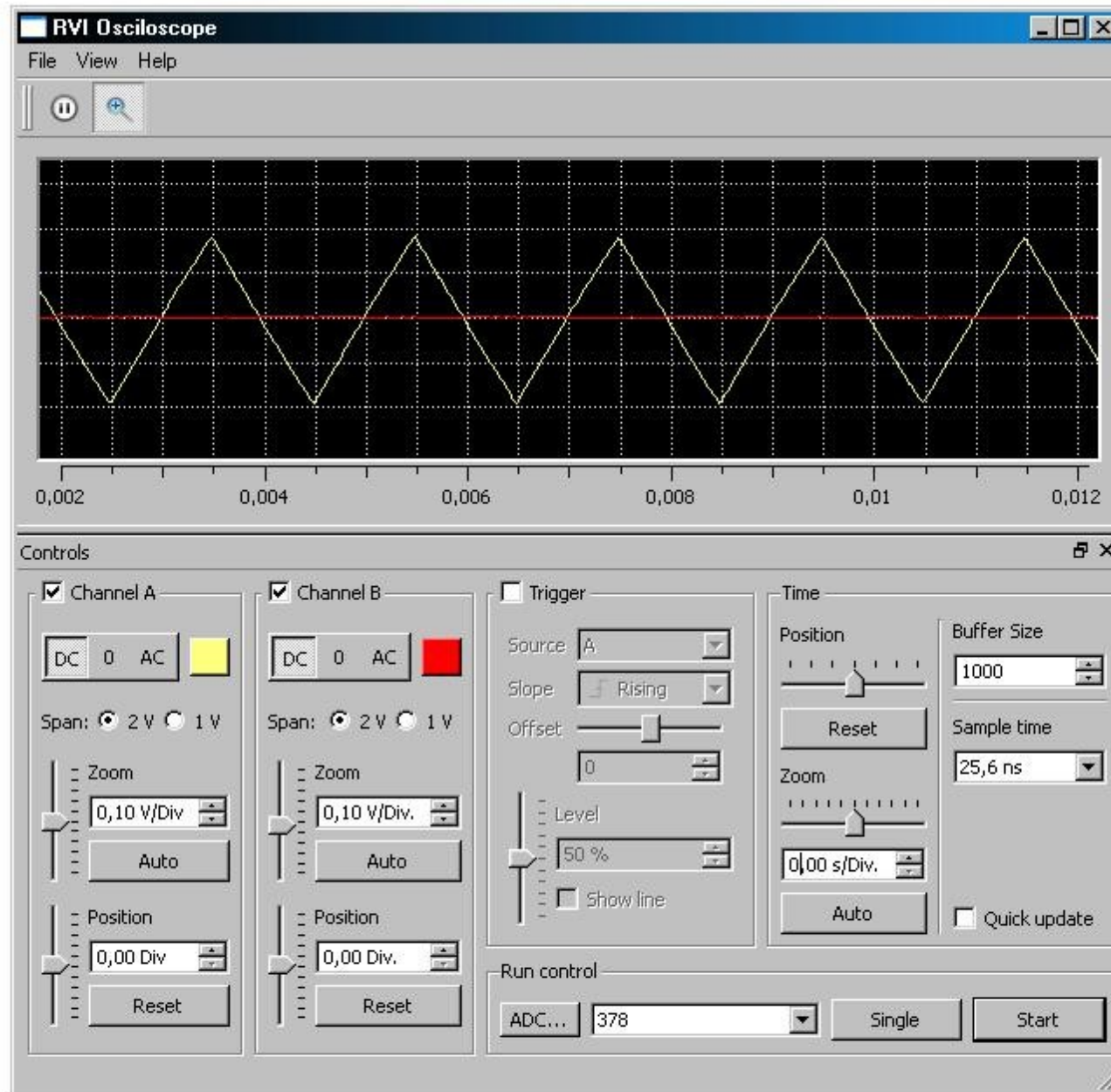
Universidad Nacional de San Luis



Virtual Instrumentation (RVI)



Hardware



Software

Virtual Instrumentation (RVI)

Staff

Facundo Aguilera

Carlos Federico Sosa Paez

Diego Costa

Andrés Miguel Airabella

FCFMN

Universidad Nacional de San Luis





Email: andres.airabella@ysbes.com.ar

Website: <http://www.unsl.edu.ar/~amairabe>



General Projects and Challenges

Andreia Barbiero (andreia.barbiero@lactec.org.br)

- GRADUATED IN COMPUTER SCIENCE AT FEDERAL UNIVERSITY OF PARANÁ - BRAZIL
- MSC IN COMPUTER ARCHITETURE AT UFPR – BRAZIL
- WORKING FOR 3 YEARS AT **LACTEC RESEARCH INSTITUTE**
- DEVELOPING EMBEDDED SYSTEM PROJECTS USING MICROCONTROLLERS, C LANGUAGE, LINUX, JAVA...



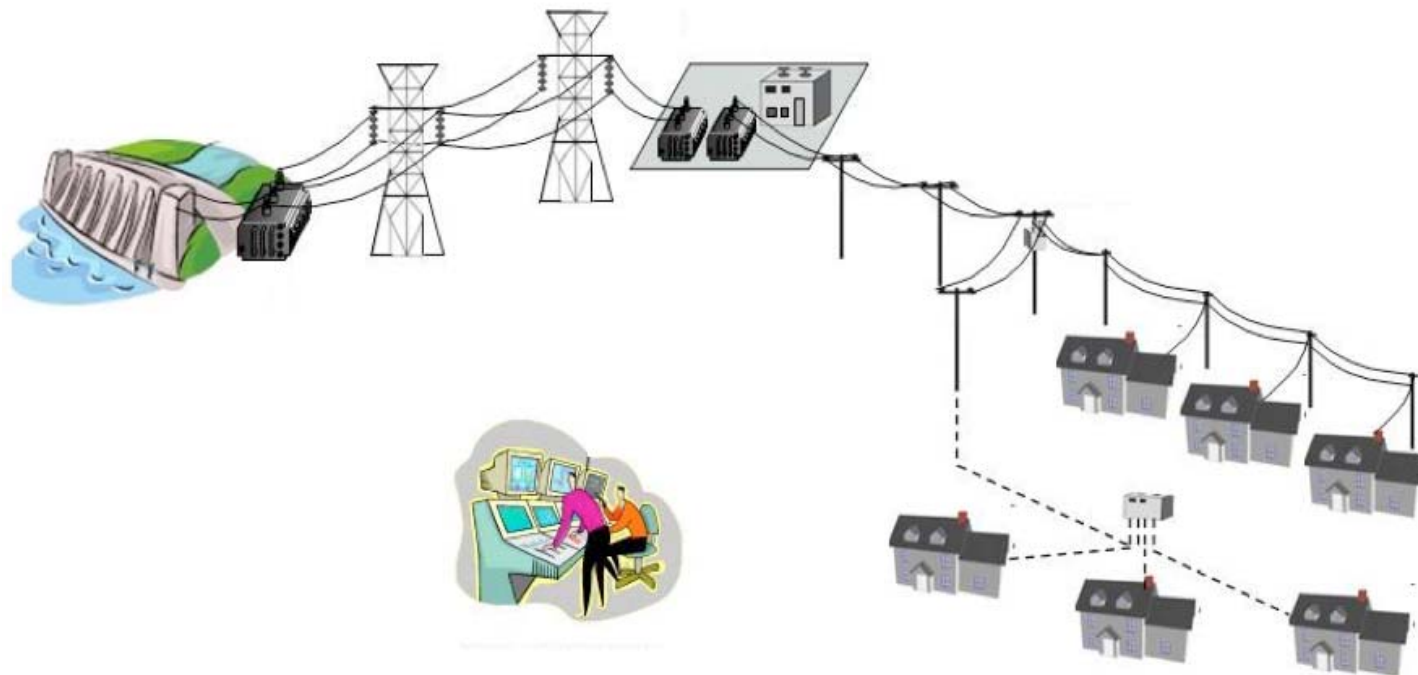
LACTEC RESEARCH INSTITUTE FOR DEVELOPMENT

- 50 YEARS OLD INSTITUTE
- TECHNOLOGICAL RESEARCH CENTER AND A NON-PROFIT AND SELF-SUSTAINABLE INSTITUTE
- WAS CREATED BY A GOVERNMENT COMPANY (COPEL) AND FEDERAL UNIVERSITY OF PARANÁ
- AROUND 400 RESEARCHERS WITH 60 DOCTORS AND MASTERS
- LOCATED AT CURITIBA – PARANÁ - BRAZIL

- MAINLY R&D PROJECTS INVOLVING POWER ENERGY UTILITIES COMPANIES

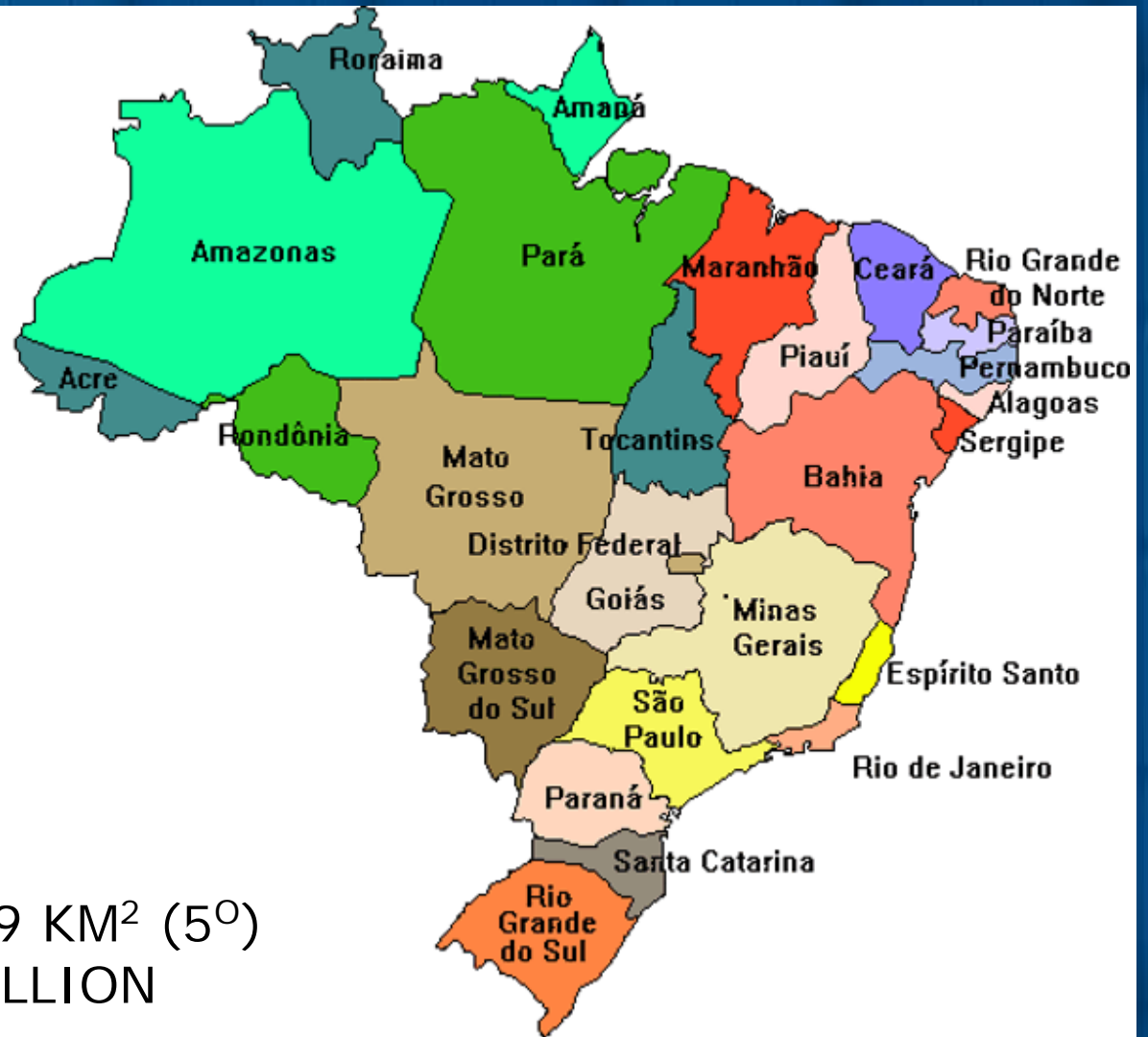
- POWER UTILITIES COMPANIES HAVE TO SPEND 0.5% OF THEIR PROFITS IN R&D PROJECTS. IT IS A LAW!

ENERGY GENERATION, TRANSMISSION AND DISTRIBUTION



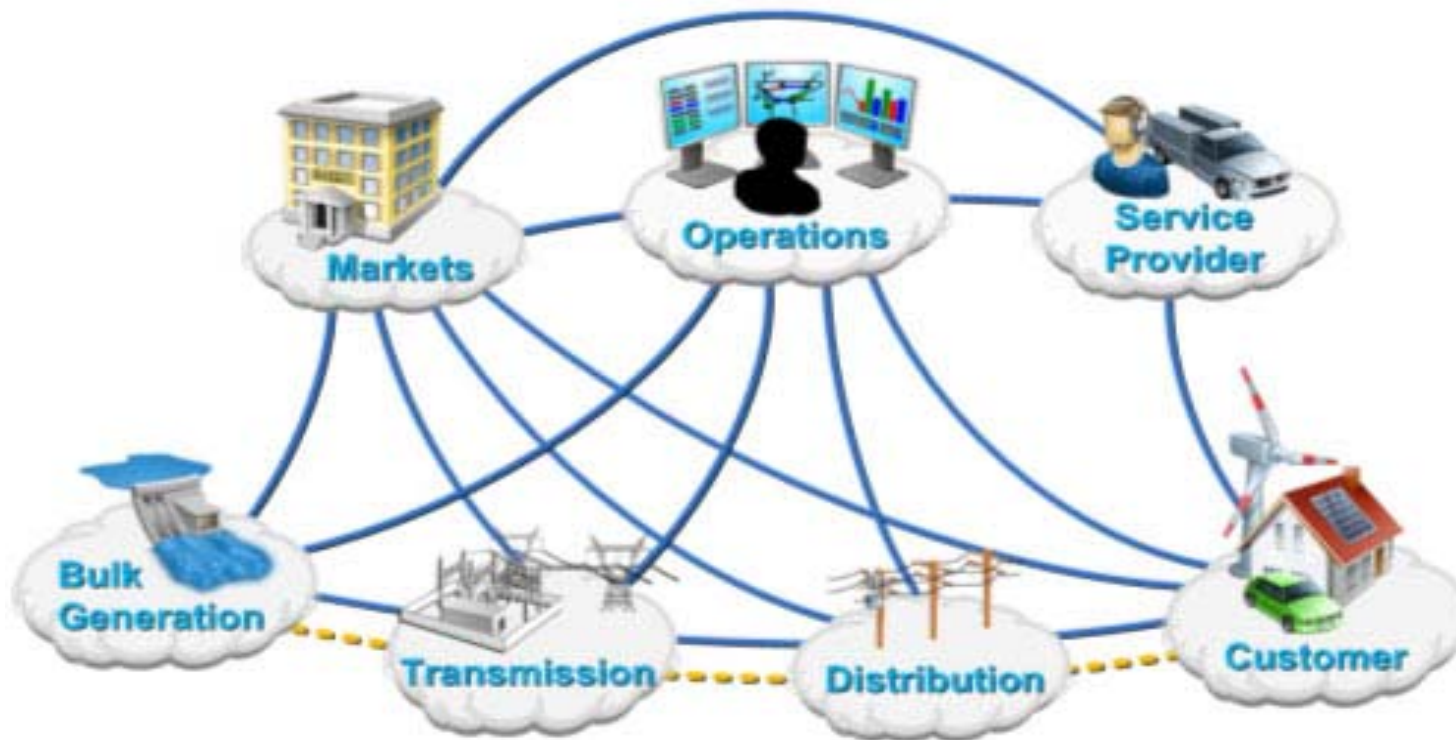
BRAZIL A HUGE COUNTRY...



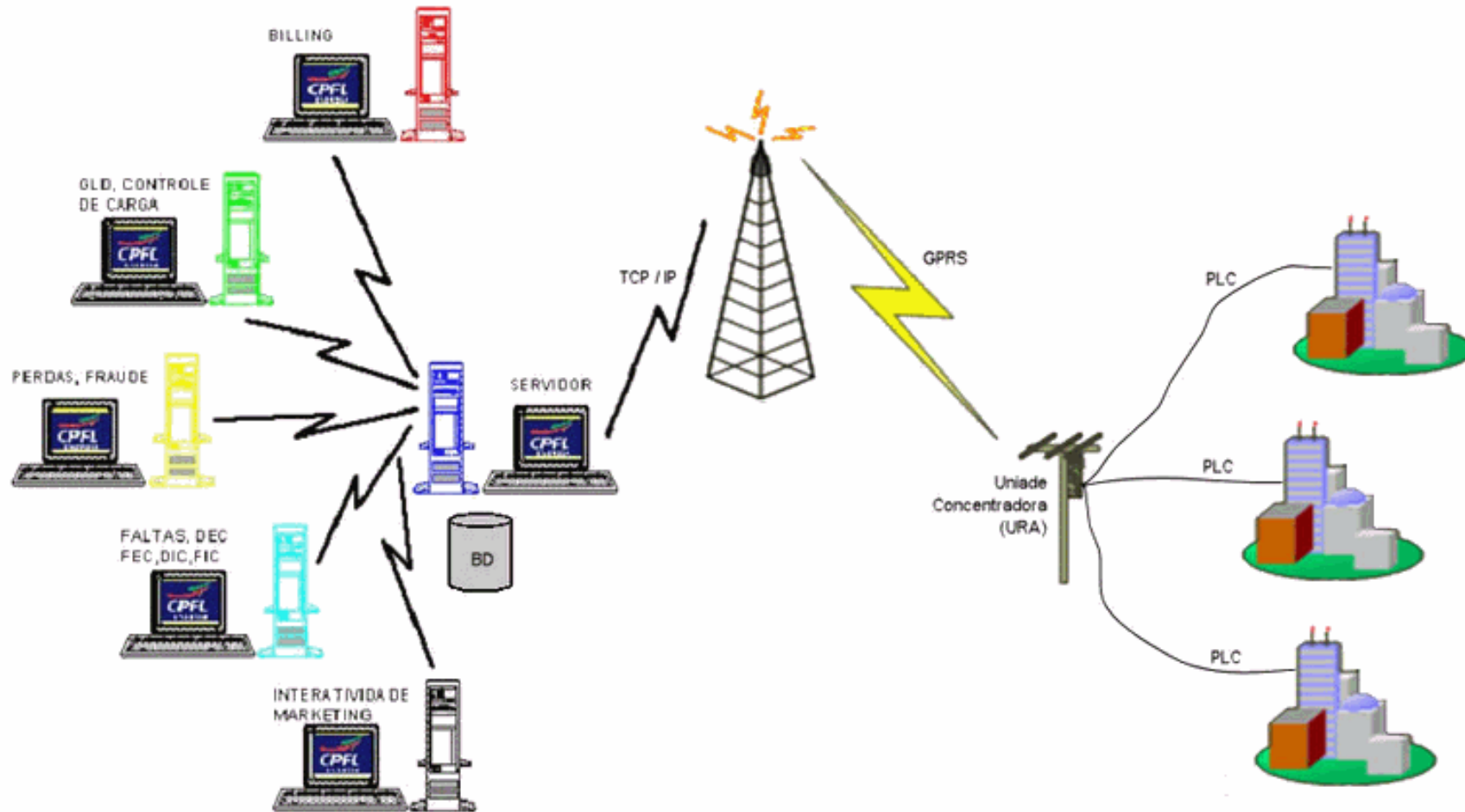


AREA : 8.514.876,599 KM² (5^o)
 POPULATION: 191 MILLION

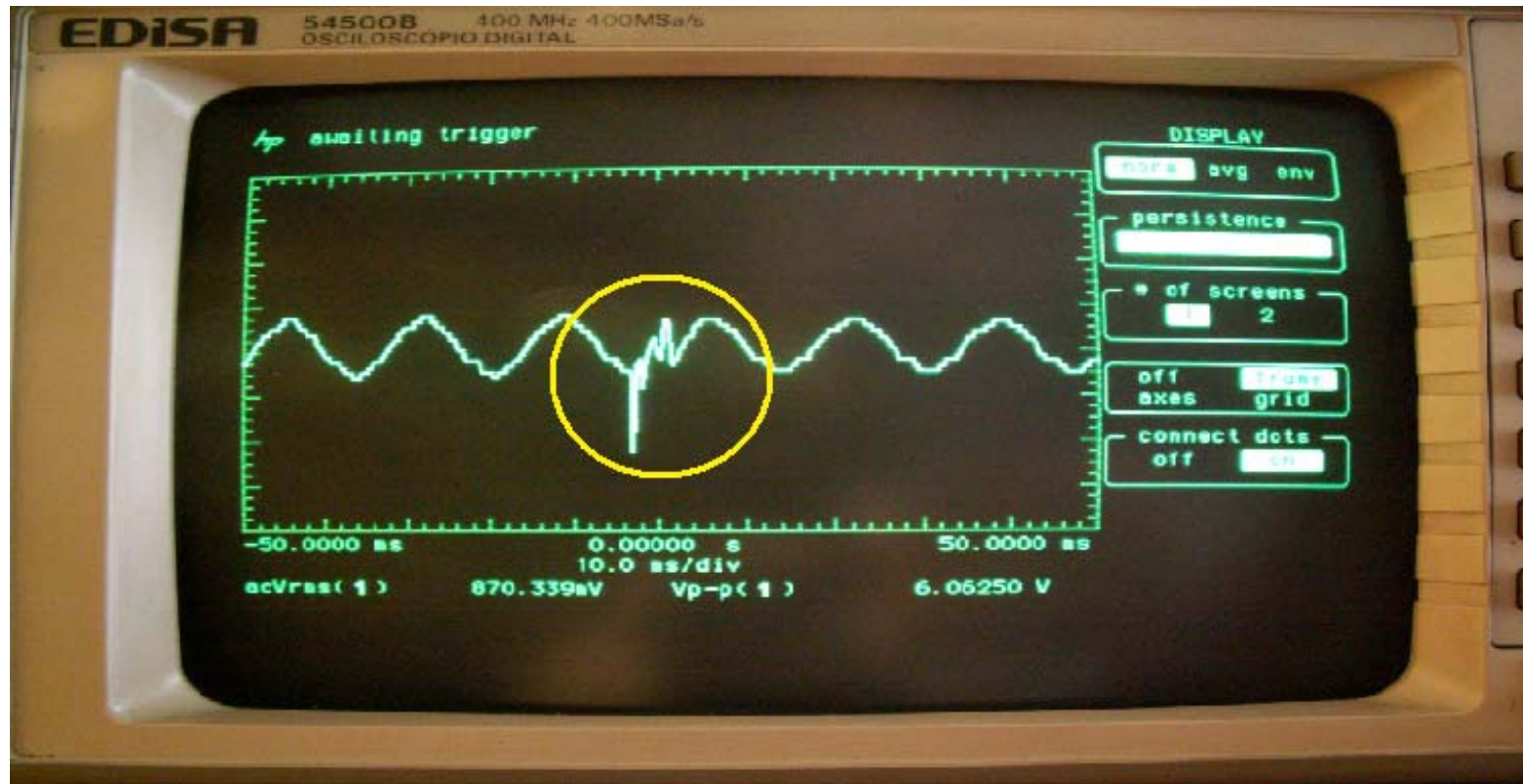
- ELECTRICAL ENERGY NETWORK IMPROVEMENT AND INTEGRATION (SMARTGRID)



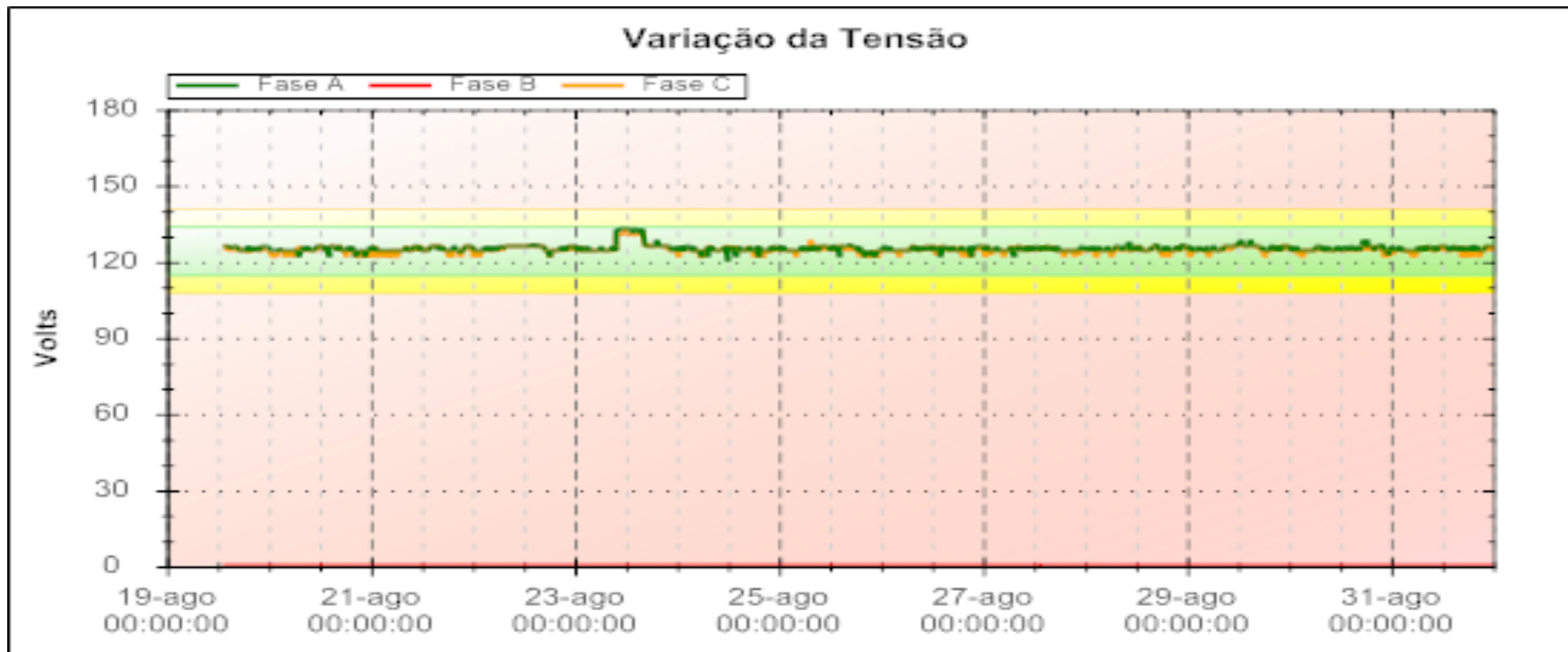
- PREVENT BLACK-OUT
- AVOID ENERGY ROBBERY (25-30% ENERGY LOSS)
- SELF-RECOVERY
- ONLINE CONSUMPTION MEASURE



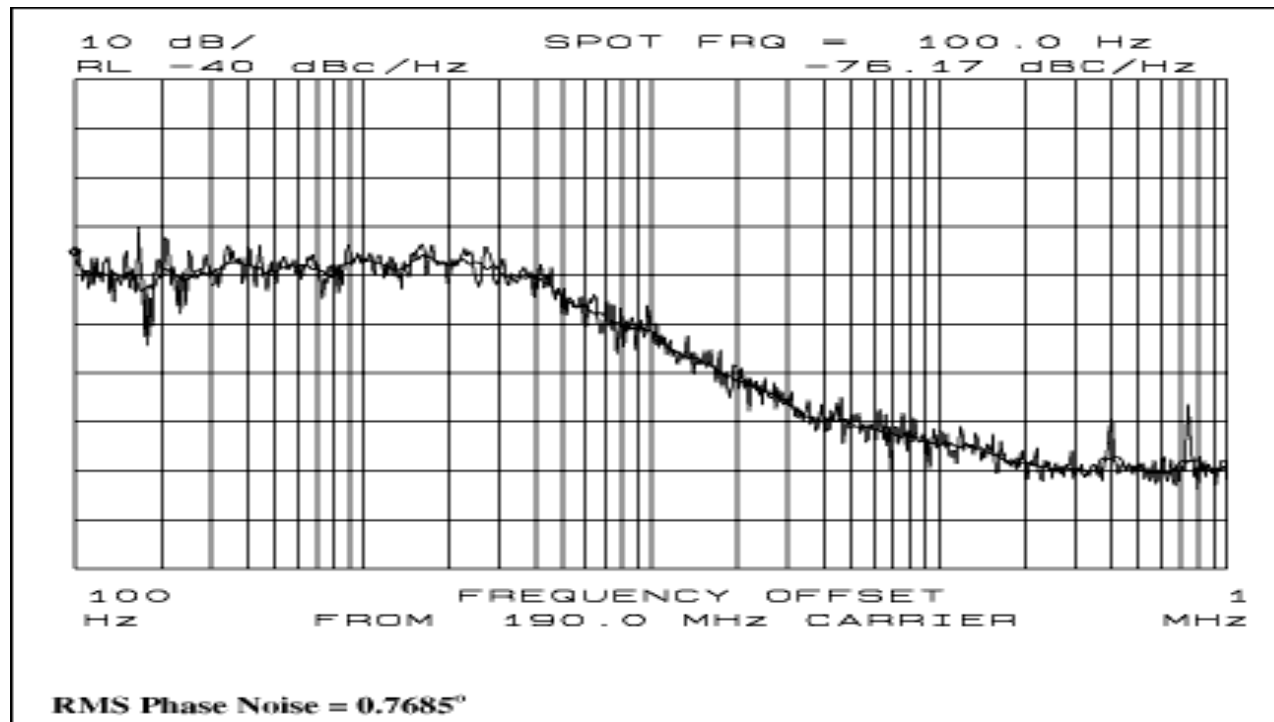
CURRENT/VOLTAGE SENSORS TO DETECT SURGES (LIGHTNING)



POWER QUALITY SYSTEMS



SIGNAL FILTERS TO CORRECT PHASE NOISE





www.lactec.org.br

FPGA Implementations of Artificial Neural Networks

Prof. Amr Khairat Radi
Physics Department
Faculty of Sciences
Ain Shams University



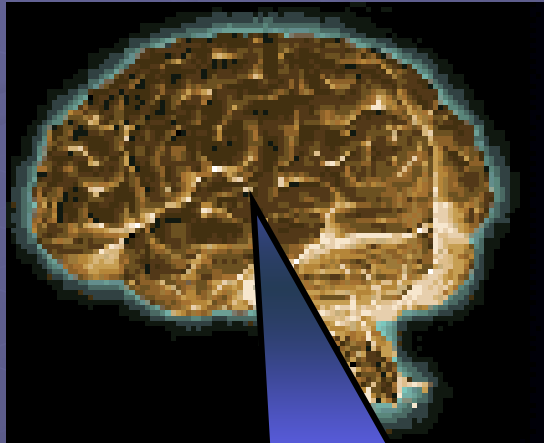
Artificial Neural Networks

- What is Neural Networks
- Artificial Neural Networks
- Applications
- Modelling a Neuron
- Hardware Neuron

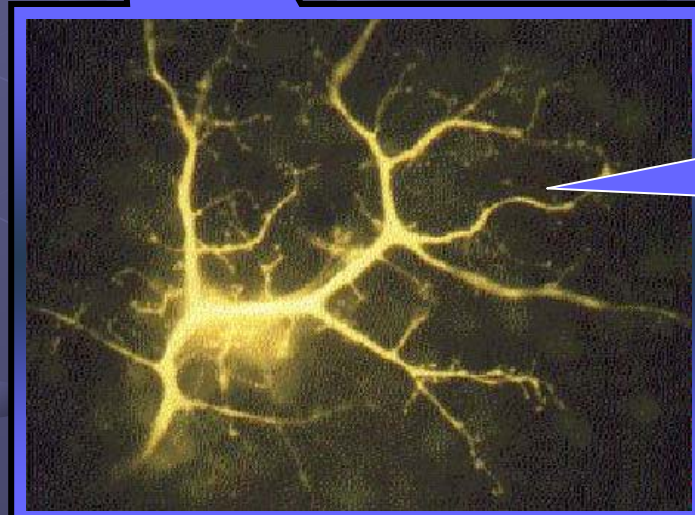
The question
'What is a neural network?'
is ill-posed.
-- Pinkus (1999)

A method of computing, based on the
interaction of multiple connected
processing elements

Brain Computer: What is it?



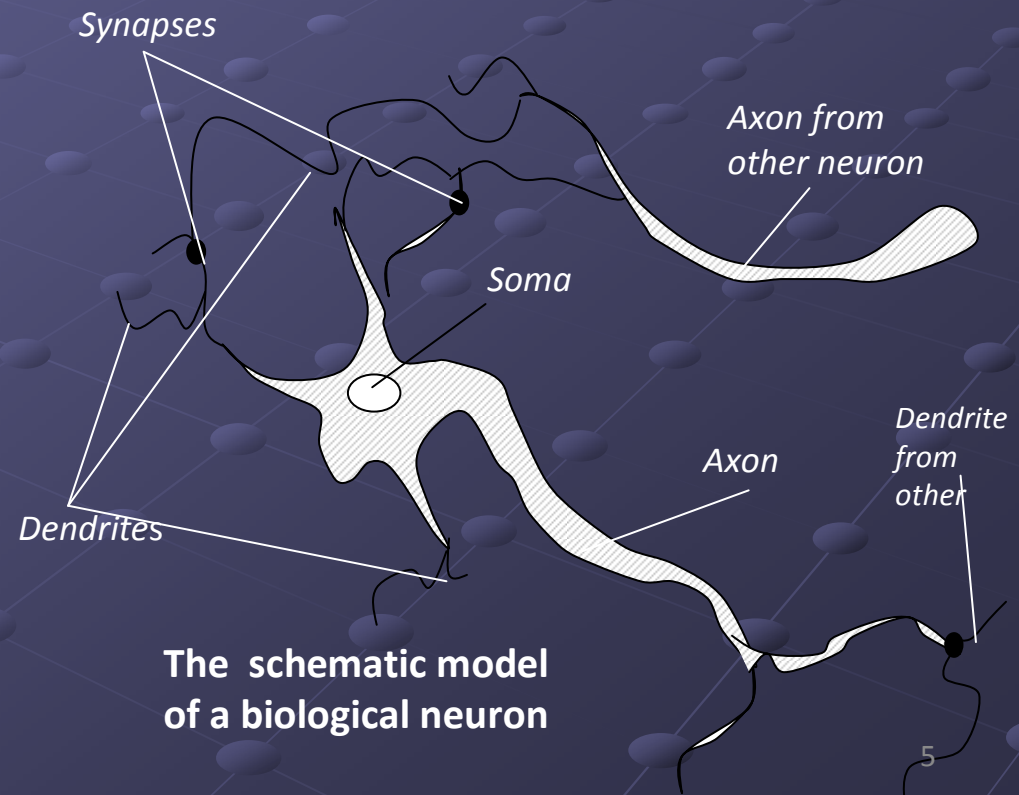
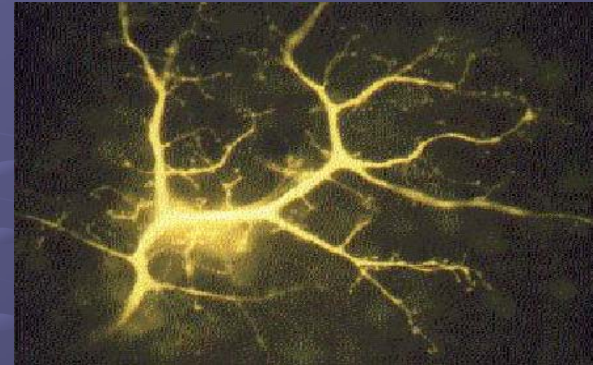
Human brain contains a massively interconnected net of 10^{13} - 10^{14} (1000 billion) neurons (cortical cells)



Biological Neuron
- The simple
"arithmetic
computing" element

Biological Neurons

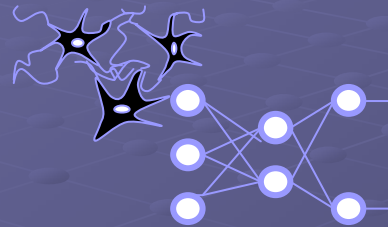
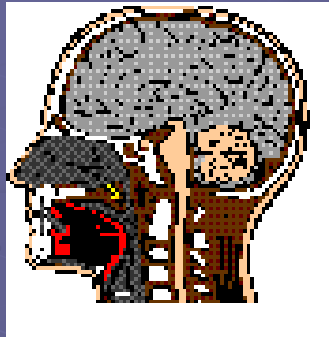
1. **Soma or body cell** - is a large, round central body in which almost all the logical functions of the neuron are realized.
2. **The axon (output)**, is a nerve fibre attached to the soma which can serve as a final output channel of the neuron. An axon is usually highly branched.
3. **The dendrites (inputs)**- represent a highly branching tree of fibres. These long irregularly shaped nerve fibres (processes) are attached to the soma.
4. **Synapses** are specialized contacts on a neuron which are the termination points for the axons from other neurons.



The schematic model of a biological neuron

ANN as a Brain-Like Computer

NN as an model of
brain-like Computer



Brain

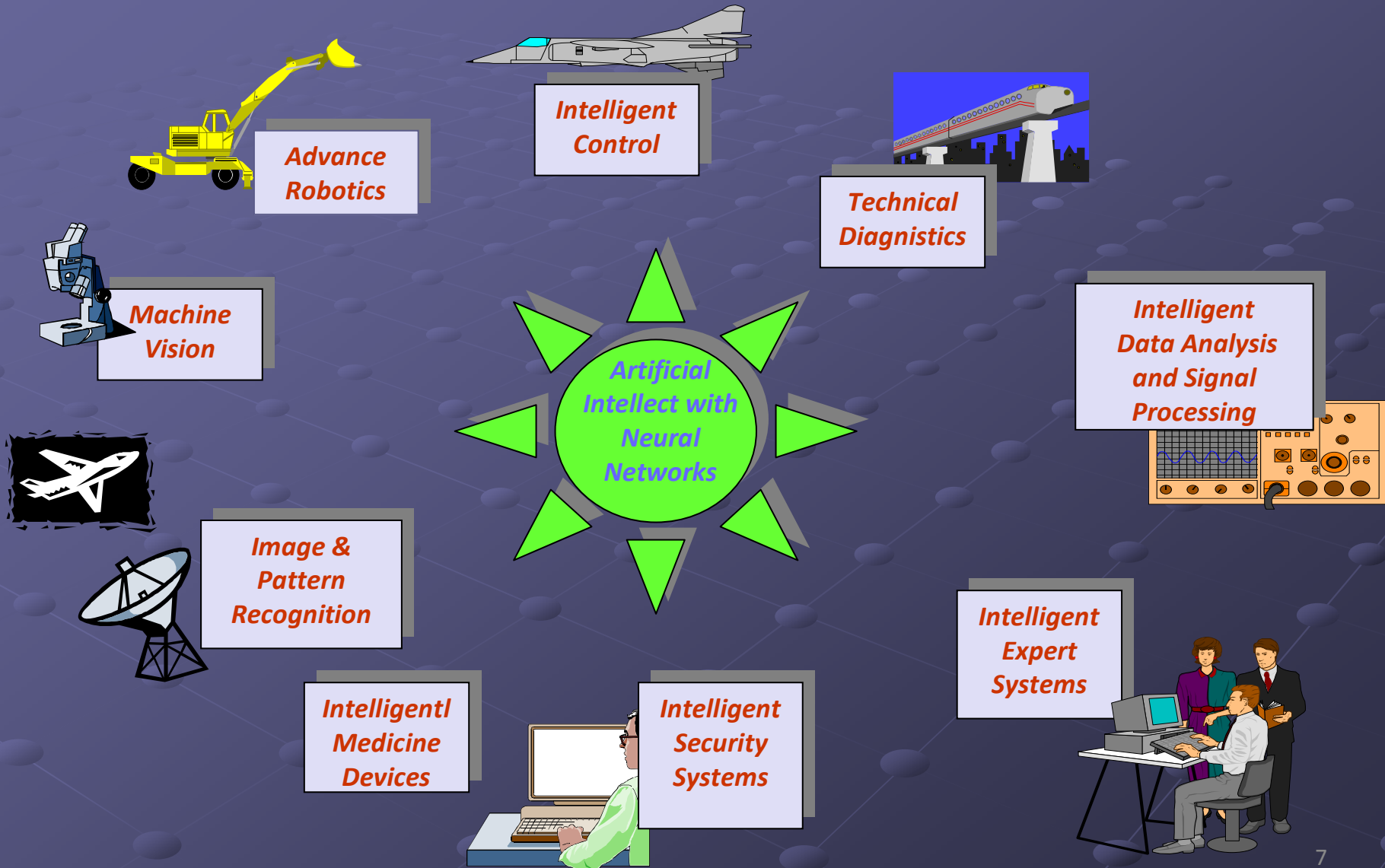
- *The human brain is still not well understood and indeed its behavior is very complex!*
- *There are about 1000 billion neurons in the human cortex and 60 trillion synapses of connections*
- *The brain is a highly complex, nonlinear and parallel computer (information-processing system)*

❖ An artificial neural network (ANN) is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use. It means that:

- *Knowledge is acquired by the network through a learning (training) process;*
- *The strength of the interconnections between neurons is implemented by means of the synaptic weights used to store the knowledge.*

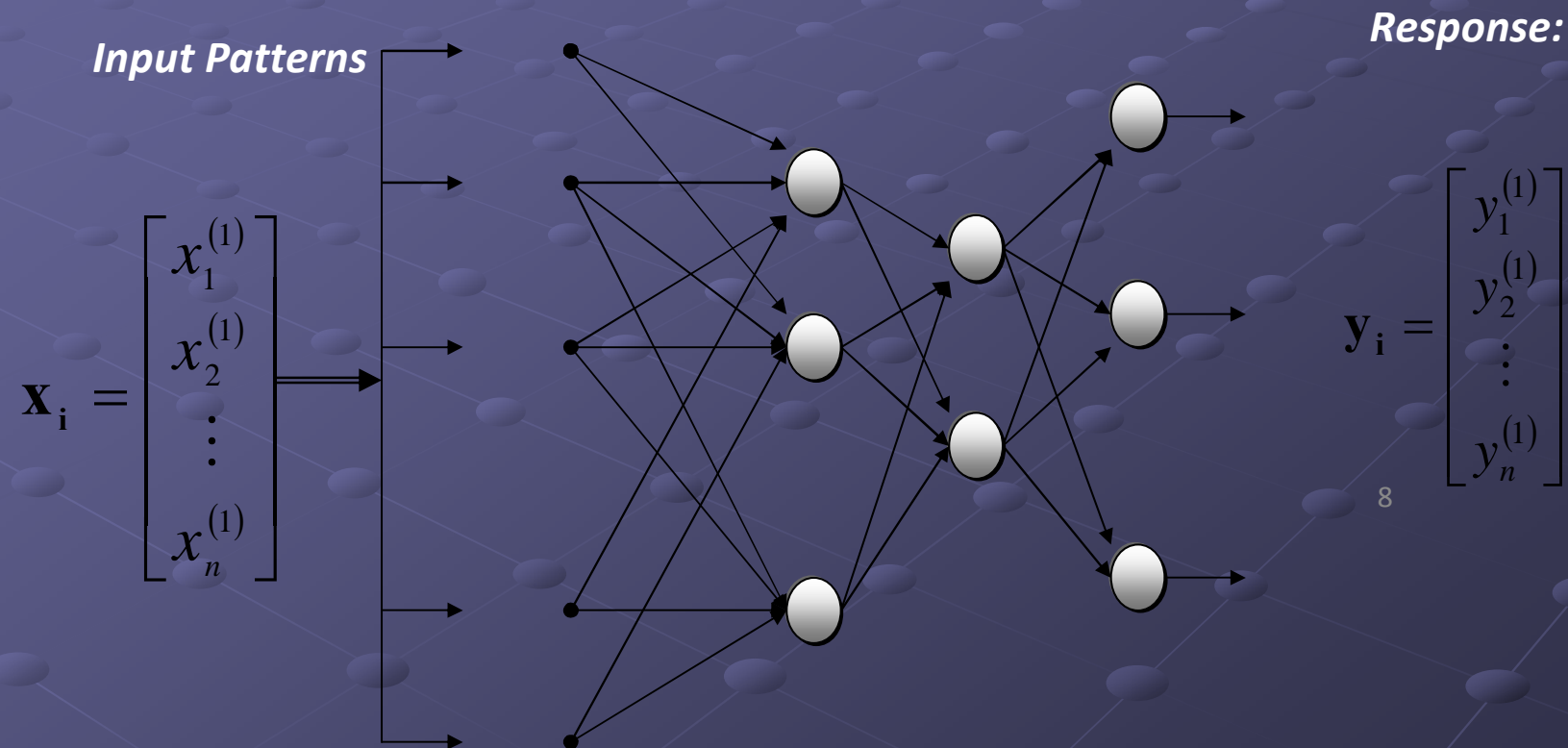
The learning process is a procedure of the adapting the weights with a learning algorithm in order to capture the knowledge. On more mathematically, the aim of the learning process is to map a given relation between inputs and output (outputs) of the network.

Applications of Artificial Neural Networks



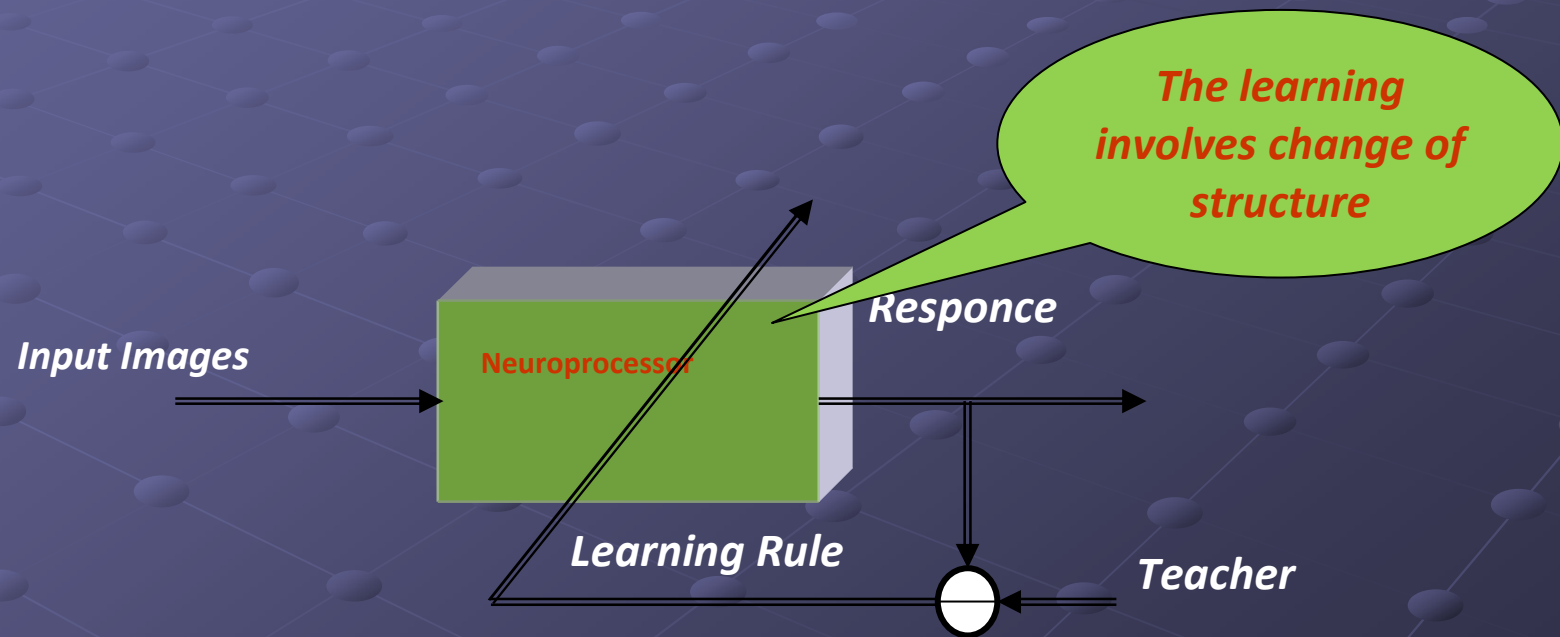
Mathematical Interpretation of Classification in Neural Networks

Mathematical model of quantization:
“Learning by Examples”

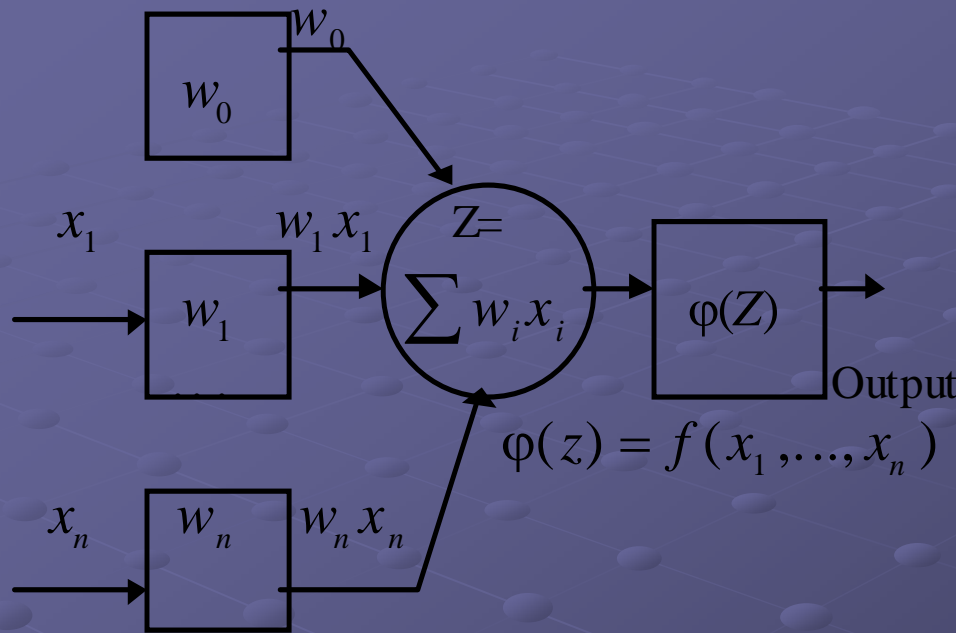


Learning via Self-Organization Principle

Self-organization – basic
principle of learning:
Structure reconstruction



Modelling a Neuron



➤ A neuron has a set of n *synapses* associated to the *inputs*. Each of them is characterized by a weight .

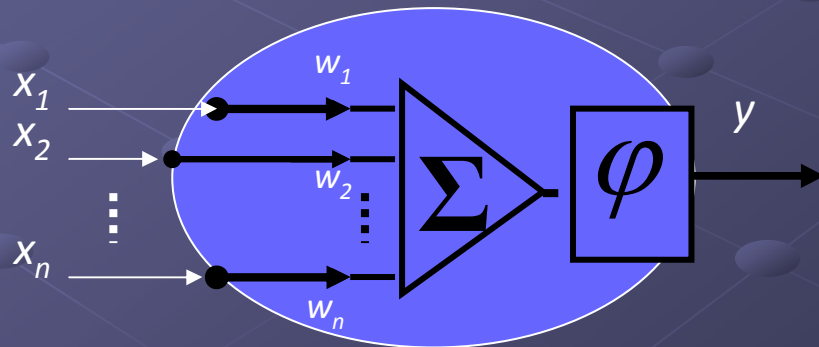
➤ A signal x_i , $i = 1, \dots, n$ at the i^{th} input is multiplied (weighted) by the weight w_i , $i = 1, \dots, n$

$$\varphi(z) = f(x_1, \dots, x_n)$$

➤ The weighted input signals are summed. Thus, a linear combination of the input signals $w_1 x_1 + \dots + w_n x_n$ is

obtained. A "free weight" (or bias) w_0 which does not correspond to any input, is added to this linear combination and this forms a *weighted sum* $z = w_0 + w_1 x_1 + \dots + w_n x_n$

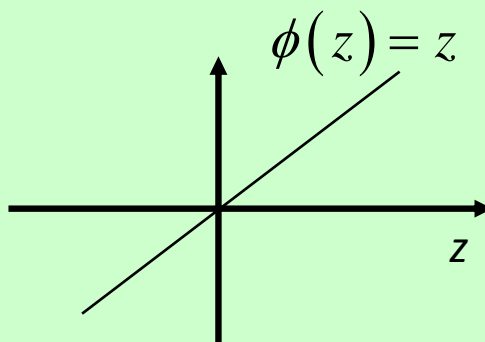
➤ A **nonlinear activation function φ** is applied to the weighted sum. A value of the activation function $y = \varphi(z)$ is the neuron's output.



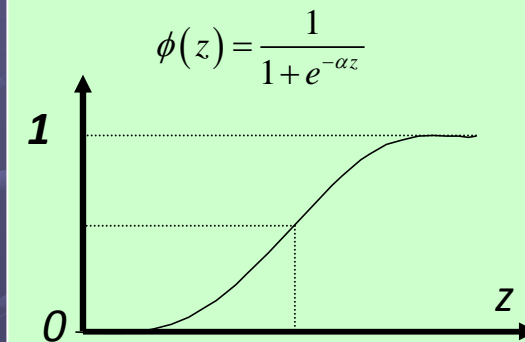
Artificial Neuron: Classical Activation Functions



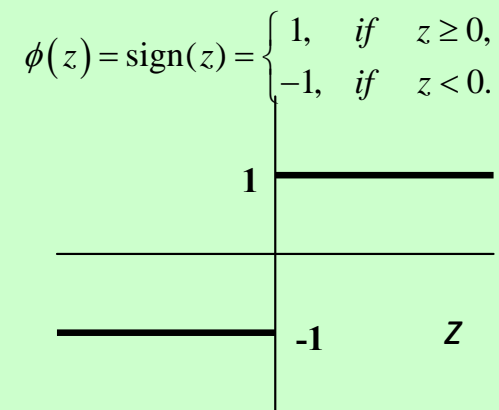
Linear activation



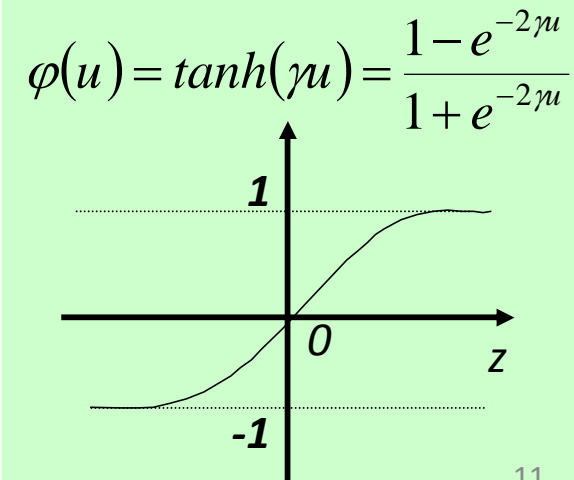
Logistic activation



Threshold activation



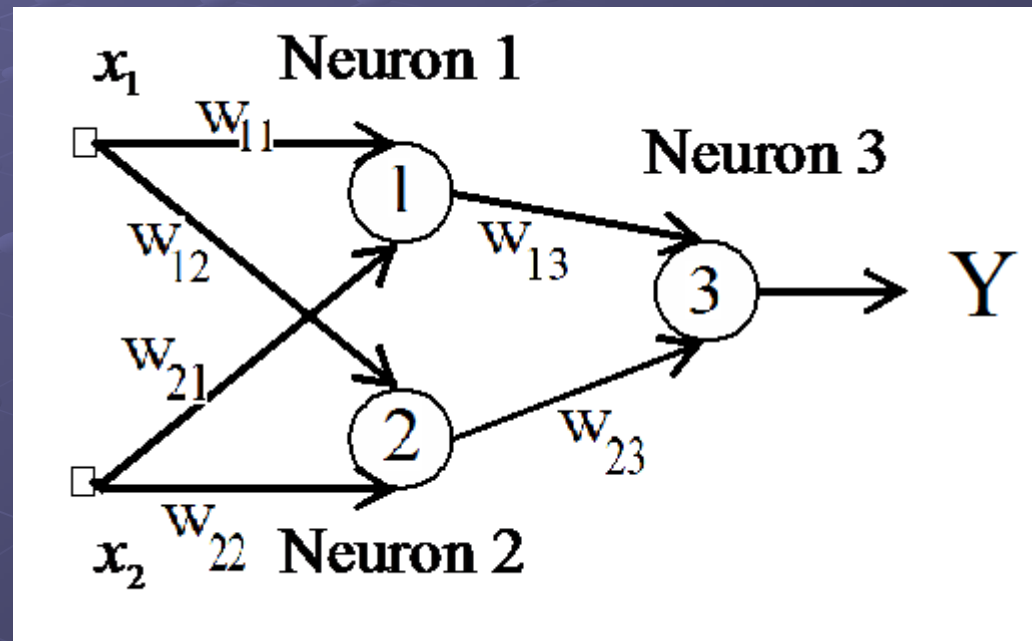
Hyperbolic tangent activation



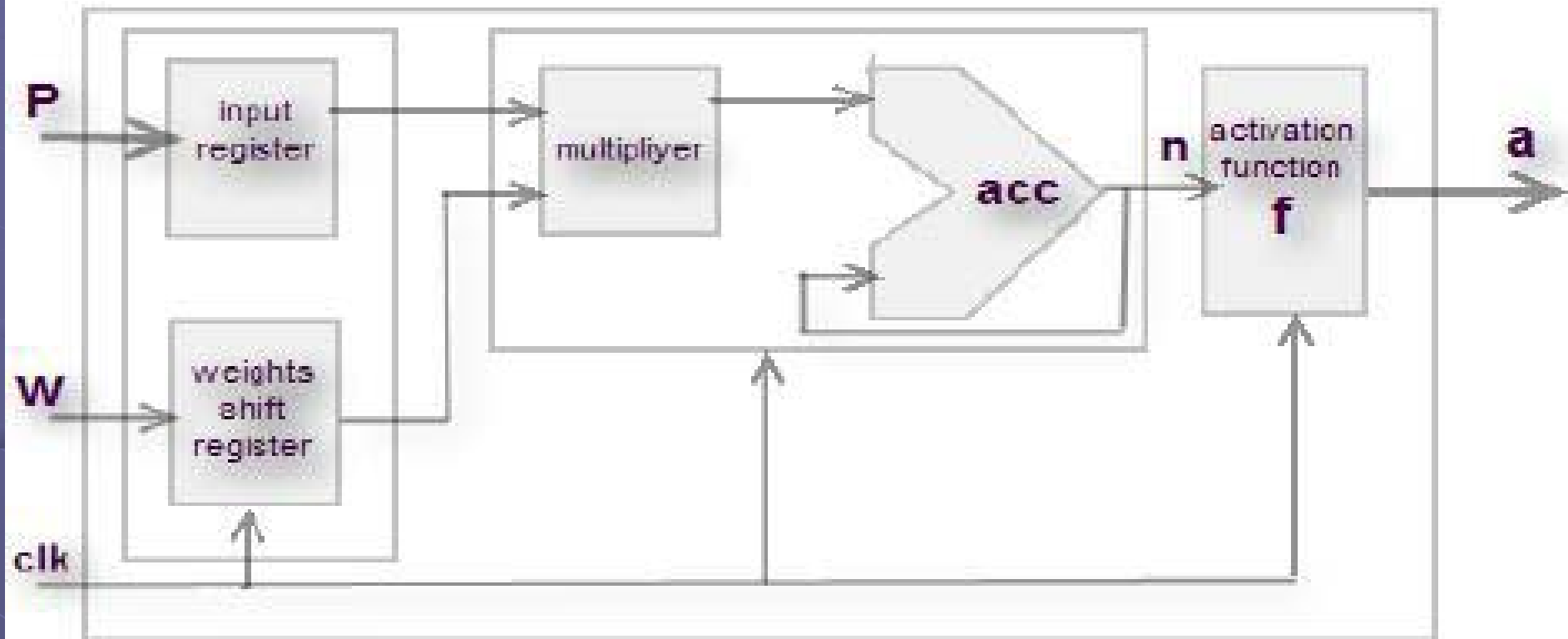
A simplest network

➤ What we need to implement ANN:

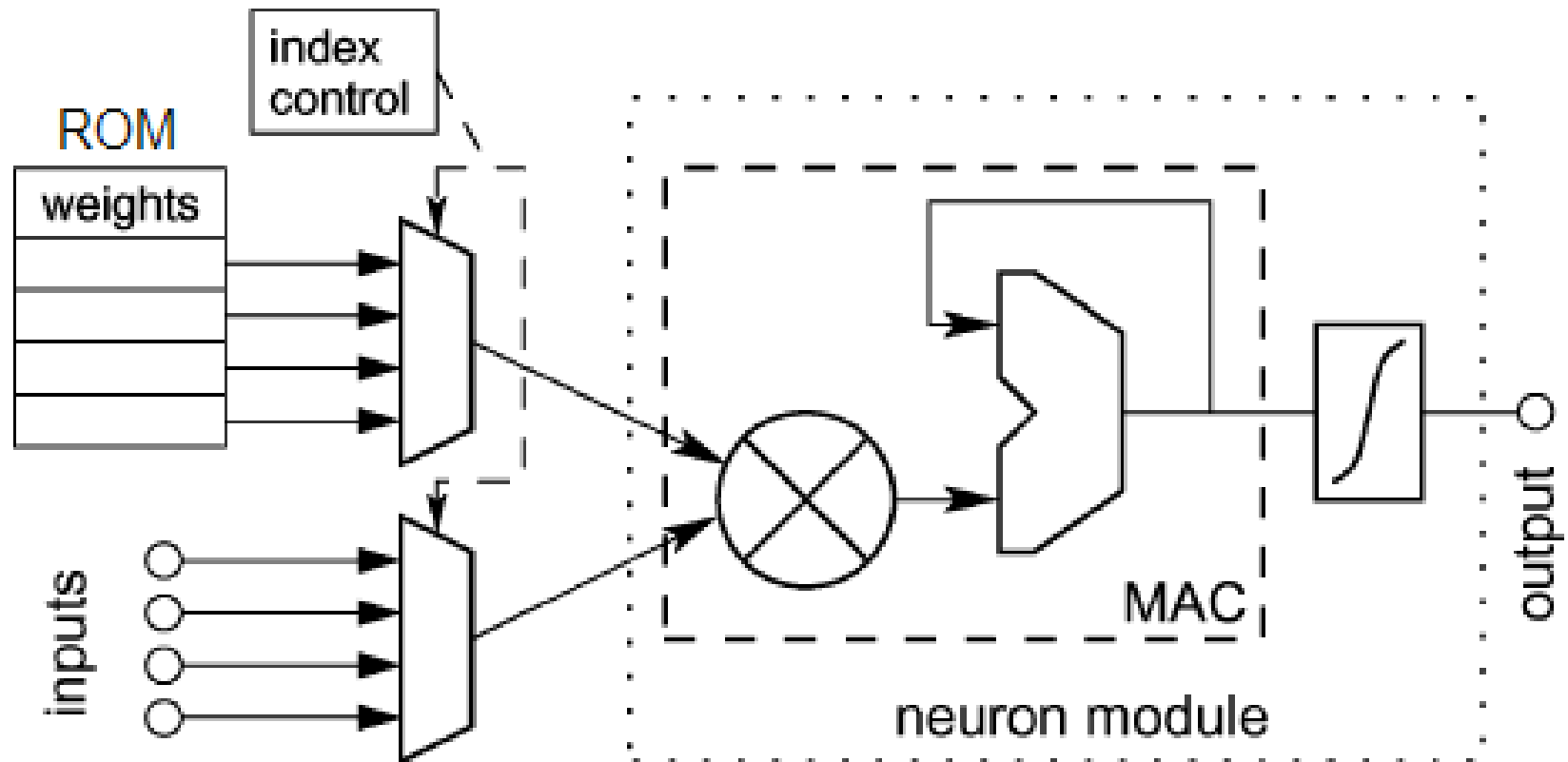
- Hardware neuron (sigmoid function)
- Random generator for the weights
- Comparing
- Changing the weights after an iteration



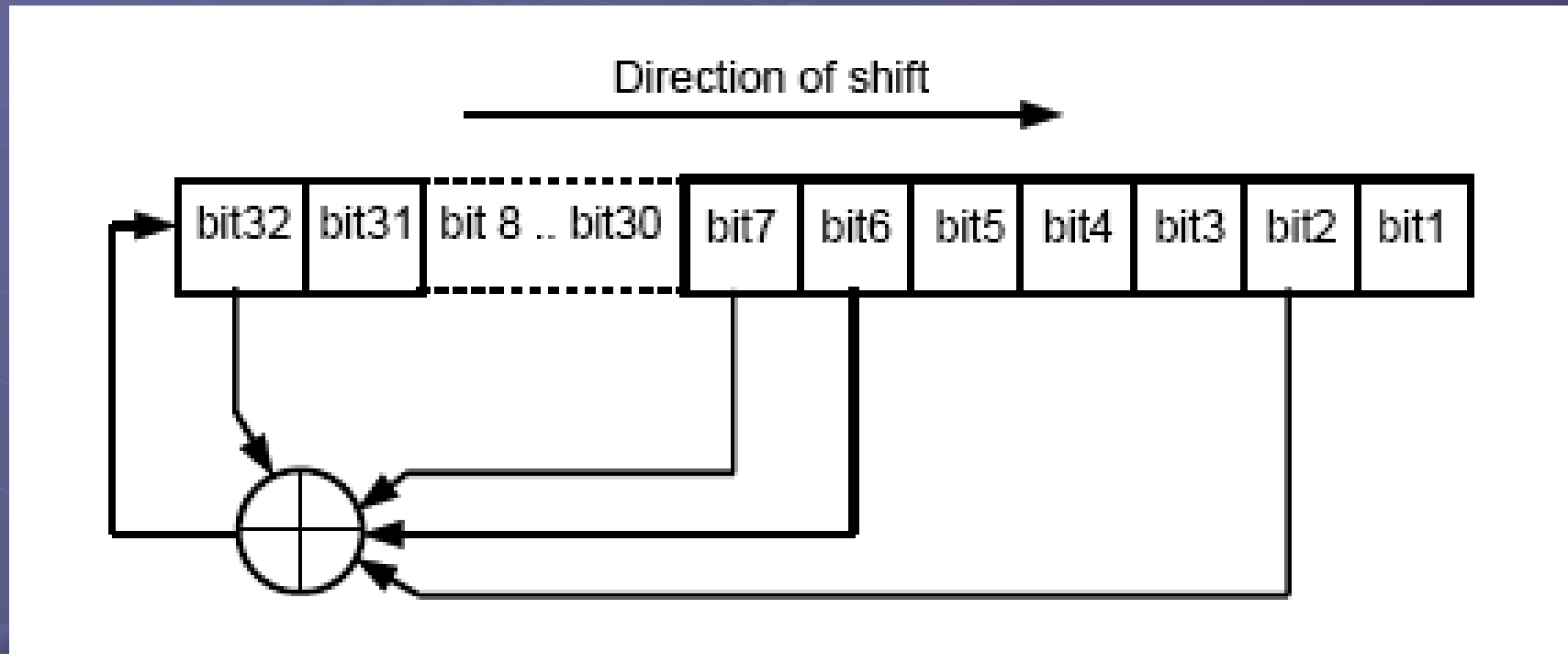
Hardware Neuron



Hardware Neuron

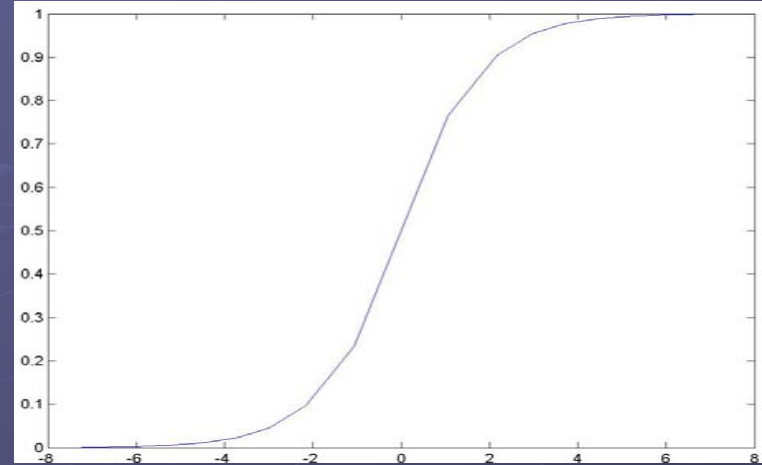


Logical Feedback Shift Register Random Number Generator

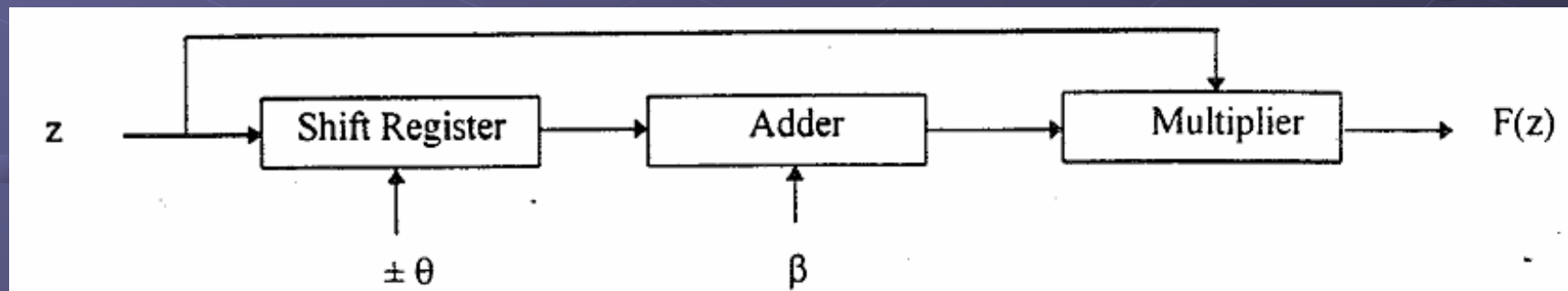


Sigmoid function

$$F(z) = \begin{cases} z(\beta - \theta z) & \text{for } 0 \leq z \leq L \\ z(\beta + \theta z) & \text{for } -L \leq z < 0 \end{cases}$$

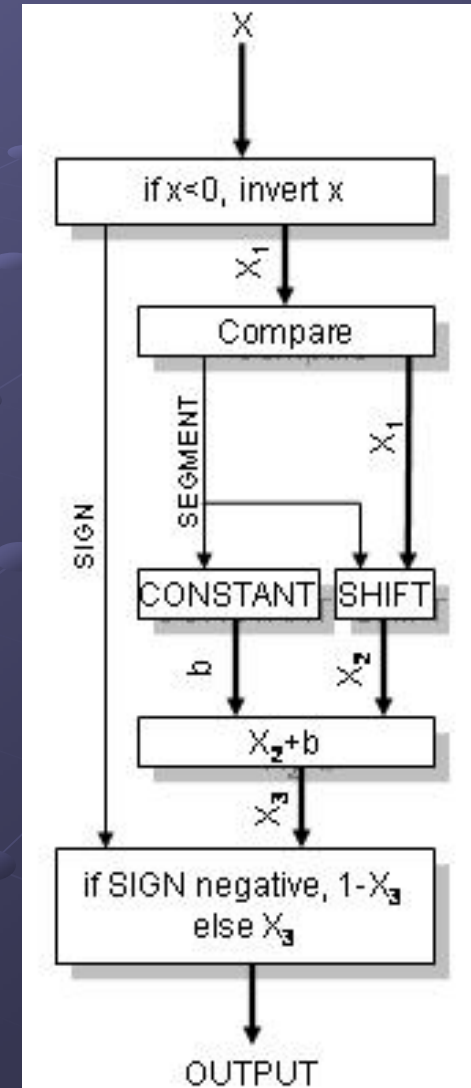


where β and θ represent the slope and the gain of the nonlinear function $F(z)$ between the saturation regions $-L$ and L .
Fig. 2 shows a block diagram of the activation function implemented using this process.



Shift and Add

- $Y(x) = 2^{-n}x + b$
- Advantages
 - Small Design
 - Short Latency of 5
- Disadvantages
 - Piecewise Outputs
 - Limited Accuracy



References

[1] L. Smith, ed. (1996, 2001), "An Introduction to Neural Networks", URL: <http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html>

[2] Sarle, W.S., ed. (1997), Neural Network FAQ, URL: <ftp://ftp.sas.com/pub/neural/FAQ.html>

[3] StatSoft, "Neural Networks", URL: <http://www.statsoftinc.com/textbook/stneunet.html>

[4] S. Cho, T. Chow, and C. Leung, "A Neural-Based Crowd Estimation by Hybrid Global Learning Algorithm", IEEE Transactions on Systems, Man and Cybernetics, Part B, No. 4. 1999.

[5] W. S. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5 pp. 115-133, 1943.

[6] J. L. McClelland, D. E. Rumelhart, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, The MIT Press, 1986.



Thank You

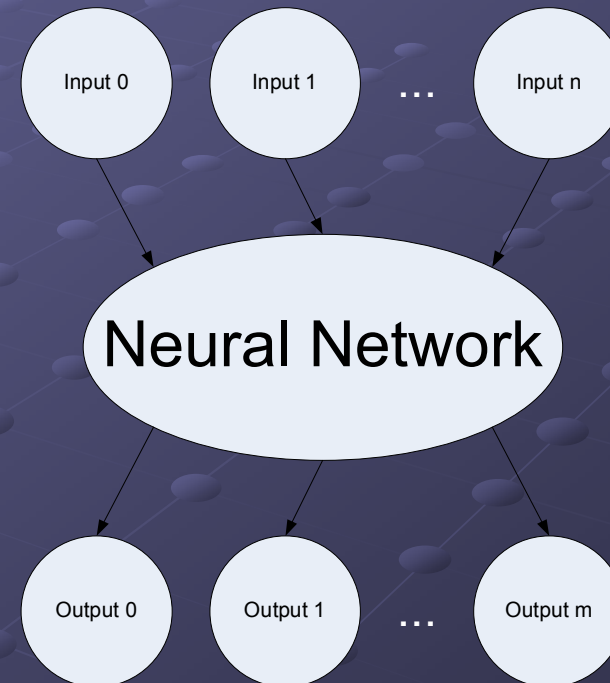
What can a Neural Net do?

- Compute a known function
- Approximate an unknown function
- Pattern Recognition
- Signal Processing

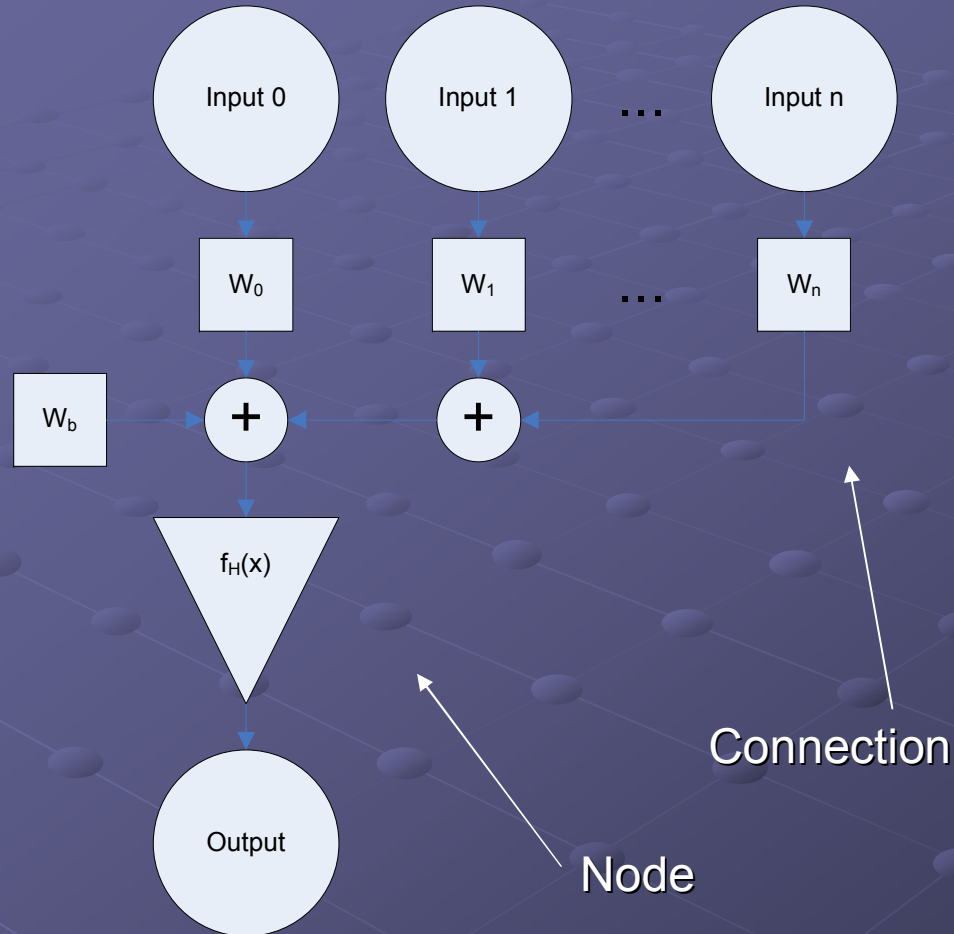
- Learn to do any of the above

Basic Concepts

- A Neural Network generally maps a set of inputs to a set of outputs
- Number of inputs/outputs is variable
- The Network itself is composed of an arbitrary number of nodes with an arbitrary topology



Basic Concepts



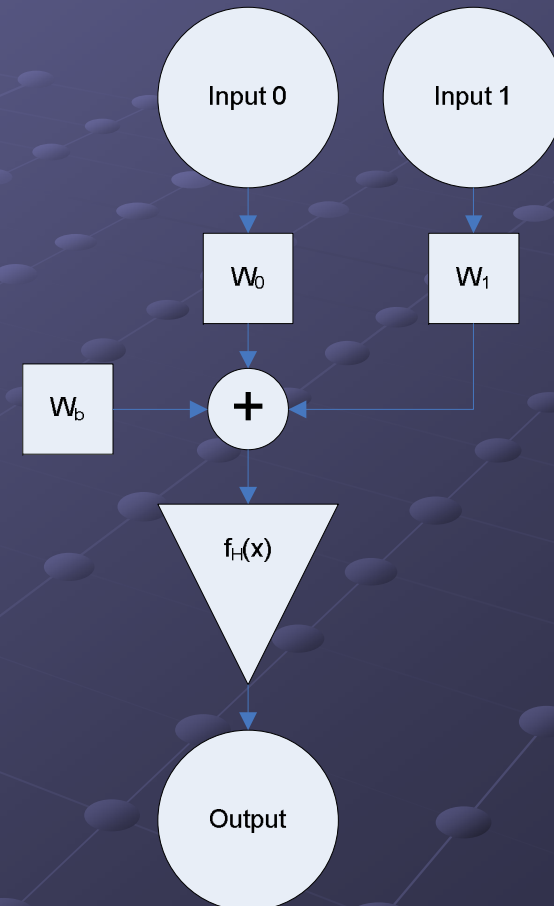
Definition of a node:

- A node is an element which performs the function

$$y = f_H(\sum(w_i x_i) + W_b)$$

Simple Perceptron

- Binary logic application
- $f_H(x) = u(x)$ [linear threshold]
- $W_i = \text{random}(-1, 1)$
- $Y = u(W_0X_0 + W_1X_1 + W_b)$
- Now how do we train it?



Basic Training

- Perception learning rule

$$\Delta W_i = \eta * (D - Y) * X_i$$

- η = Learning Rate

- D = Desired Output

- Adjust weights based on a how well the current weights match an objective

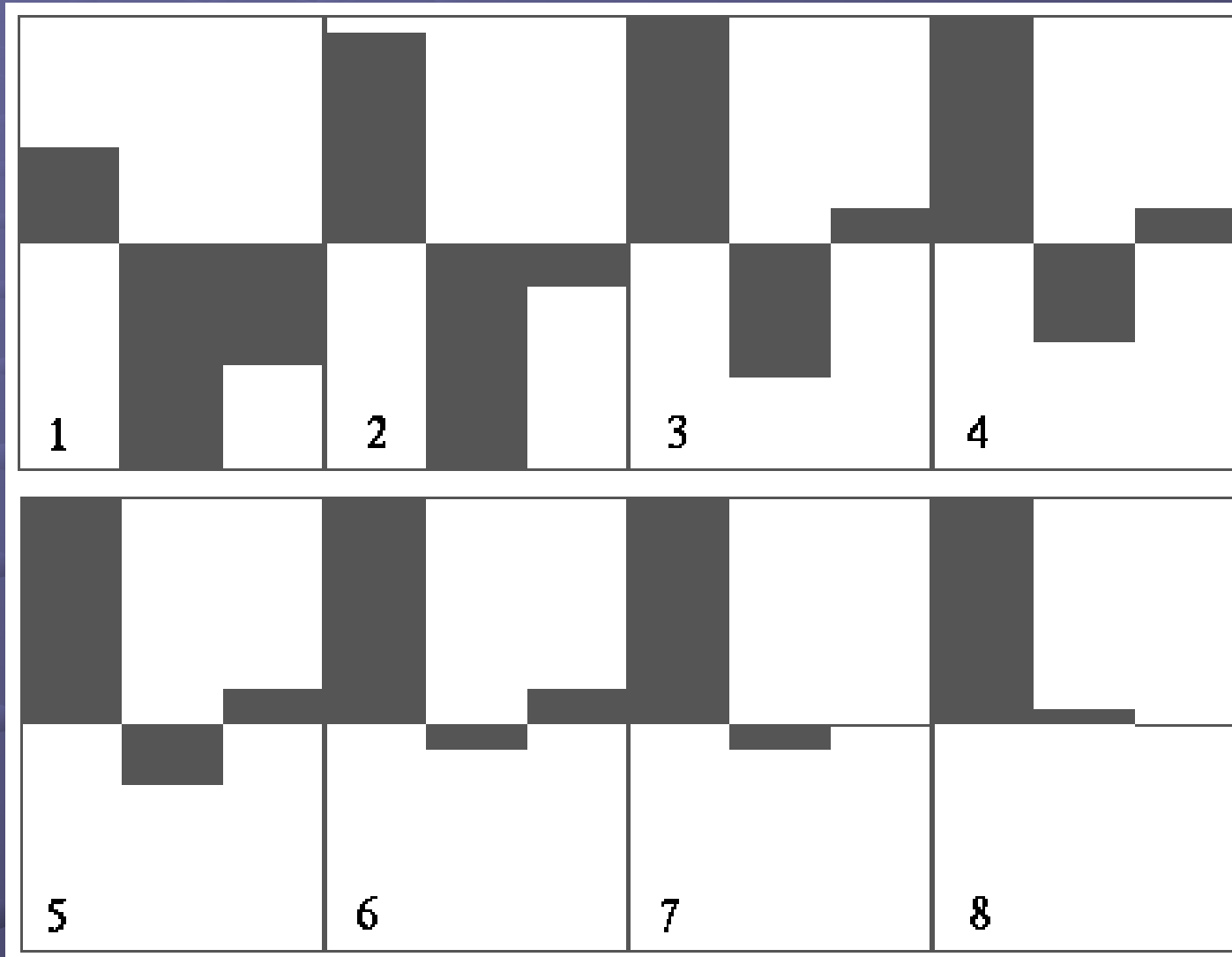
Logic Training

- Expose the network to the logical OR operation
- Update the weights after each epoch
- As the output approaches the desired output for all cases, ΔW_i will approach 0

X_0	X_1	D
0	0	0
0	1	1
1	0	1
1	1	1

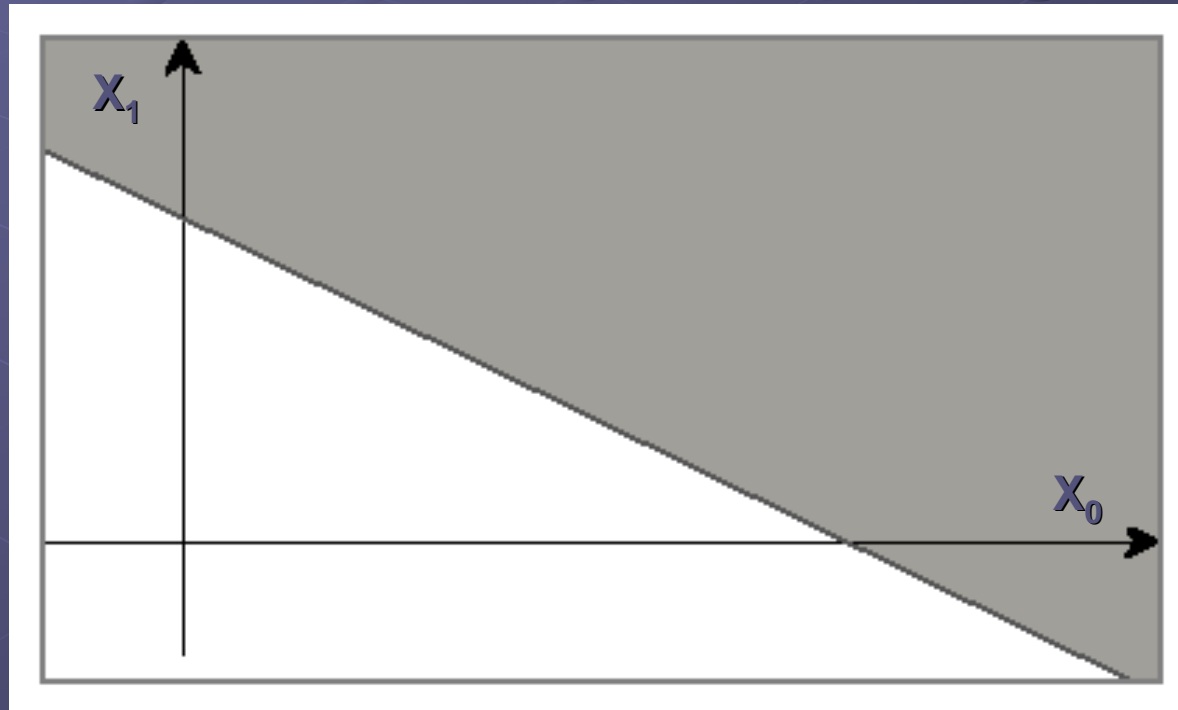
Results

W_0 W_1 W_b



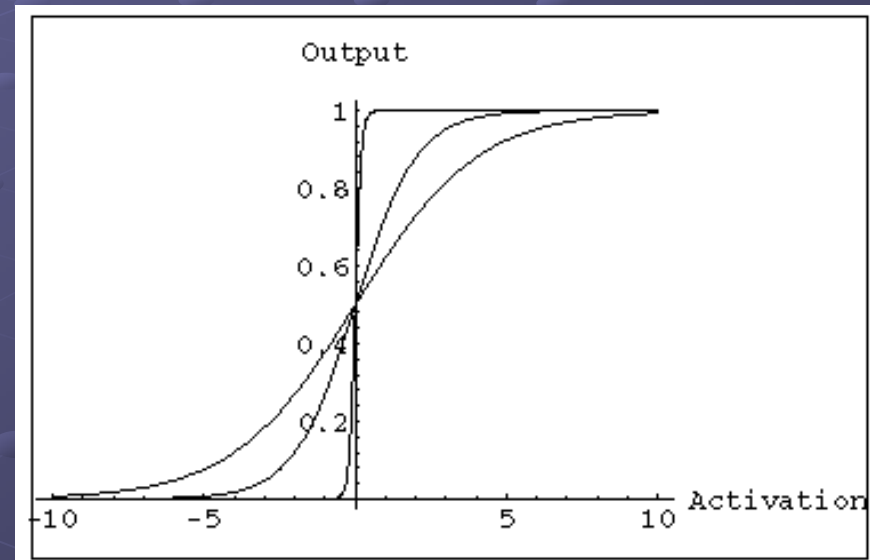
Details

- Network converges on a hyper-plane decision surface
- $X_1 = (W_0/W_1)X_0 + (W_b/W_1)$



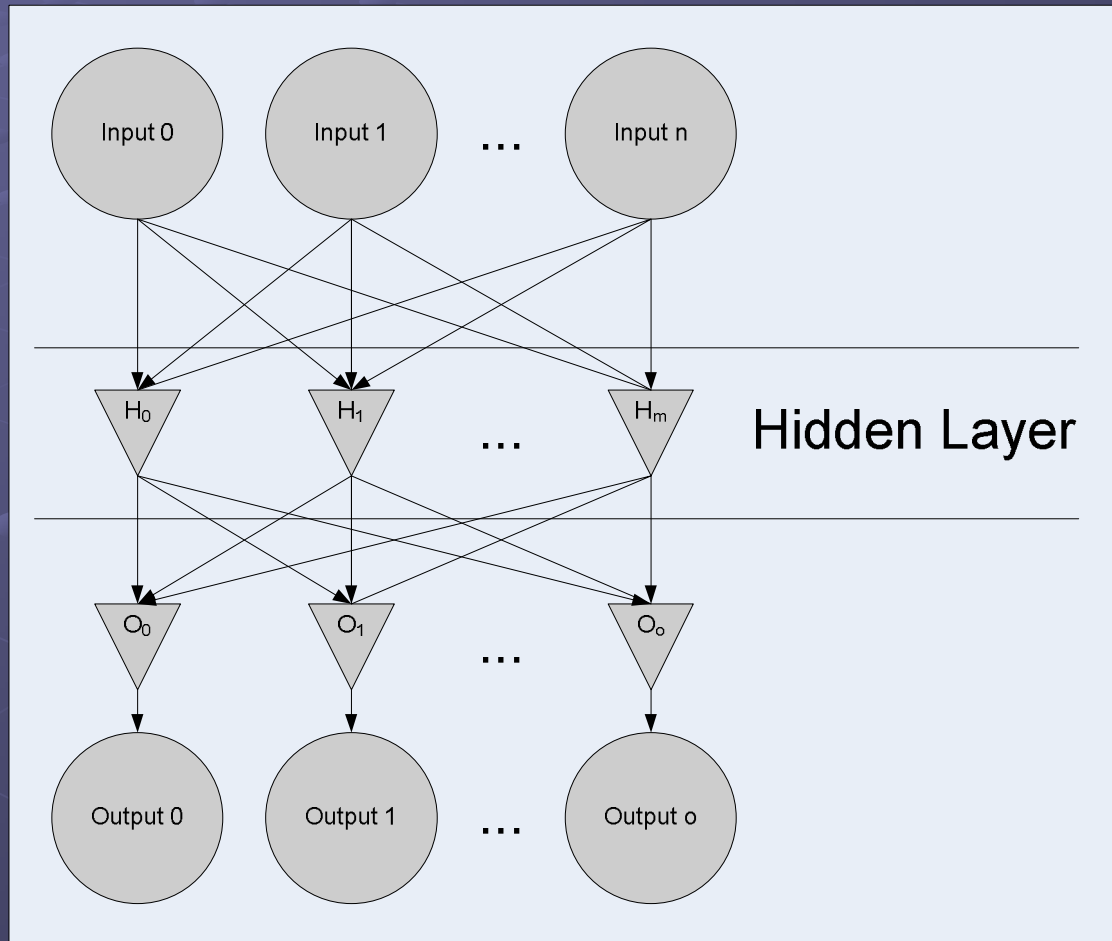
Typical Activation Functions

- $F(x) = 1 / (1 + e^{-k \sum (w_i x_i)})$
- Shown for
 $k = 0.5, 1$ and 10
- Using a nonlinear function which approximates a linear threshold allows a network to approximate nonlinear functions



Back-Propagated Delta Rule Networks (BP)

- Inputs are put through a 'Hidden Layer' before the output layer
- All nodes connected between layers



BP Network Details

● Forward Pass:

- Error is calculated from outputs
- Used to update output weights

● Backward Pass:

- Error at hidden nodes is calculated by back propagating the error at the outputs through the new weights
- Hidden weights updated

In Matrix Form

For:

n inputs, m hidden nodes
and q outputs

o_{lk} is the output of the l^{th} neuron
For the k^{th} of p patterns

$$\mathbf{A} = \begin{pmatrix} a_{10} & a_{11} & \cdots & a_{1n} \\ a_{20} & a_{21} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m0} & a_{m1} & \cdots & a_{mn} \end{pmatrix},$$
$$\mathbf{B} = \begin{pmatrix} b_{10} & b_{11} & \cdots & b_{1m} \\ b_{20} & b_{21} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{q0} & b_{q1} & \cdots & b_{qm} \end{pmatrix}.$$

$$o_{lk} = f_H \left(\sum_{j=0}^m b_{lj} f_H \left(\sum_{i=0}^n a_{ji} x_{ik} \right) \right), \quad 1 \leq k \leq p.$$

\mathbf{v}_k is the output of the hidden layer
 \mathbf{o}_k is the true output vector

$$\mathbf{v}_k = \begin{pmatrix} 1 \\ F_H(\mathbf{A}\mathbf{x}_k) \end{pmatrix}$$
$$\mathbf{o}_k = F(\mathbf{A}, \mathbf{B}, \mathbf{x}_k) = F_H(\mathbf{B}\mathbf{v}_k)$$

Matrix Tricks

- $E(\mathbf{A}, \mathbf{B}) = \sum_{k=1}^p (\mathbf{t}_k - \mathbf{o}_k)^T (\mathbf{t}_k - \mathbf{o}_k)$
- \mathbf{t}_k denotes true output vectors
- The optimal weight matrix of B can be computed directly if $f_H^{-1}(\mathbf{t})$ is known
- $\mathbf{B}' = f_H^{-1}(\mathbf{t})\mathbf{v}^T(\mathbf{v}\mathbf{v}^T)^*$
- So... $E(\mathbf{A}, \mathbf{B}) = E(\mathbf{A}, \mathbf{B}(\mathbf{A})) = E'(\mathbf{A})$
 - Which makes our weight space much smaller

Hybrid LS RS/SA/GA Training

- Delta rule training may converge to a local minimum
- Hybrid Global Learning (HGL) will converge on a global minimum
 - Randomize A $[-0.5, 0.5]$
 - Minimize the Error function $E'(A)$

Random Sampling

- Randomize A $[-0.5, 0.5]$
- Loop for a long time
 - Compute $B_{\text{new}} = B(A)$
 - if $E(A, B) < \text{THRESH}$
 - Training Complete
 - else
 - $A_{\text{new}} = A + N(0, \sigma)$
 - $\sigma = \sigma(1 - \text{dec})$
 - if $E(A, B) > E(A_{\text{new}}, B_{\text{new}})$
 - $E(A, B) = E(A_{\text{new}}, B_{\text{new}})$
 - $A = A_{\text{new}}; B = B_{\text{new}}; \sigma = \sigma_{\text{bound}}$

Other methods

- Simulated Annealing

- More accurate results
- Much slower

- Genetic Algorithms

- More accurate results
- Slower

- For details on methods and results see:

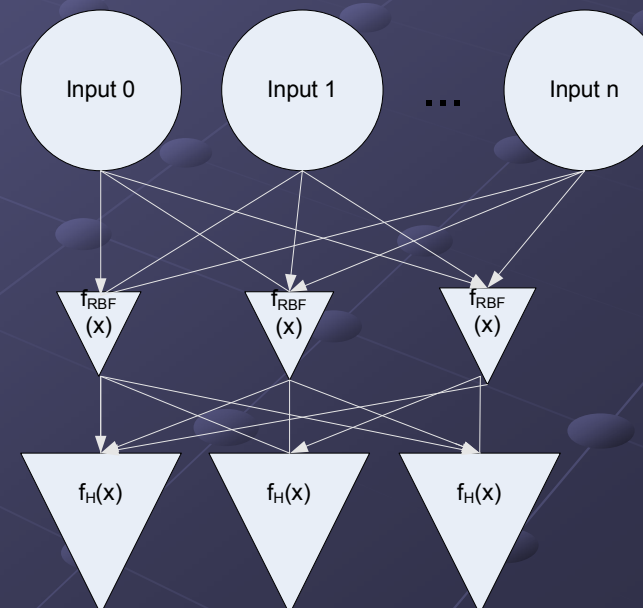
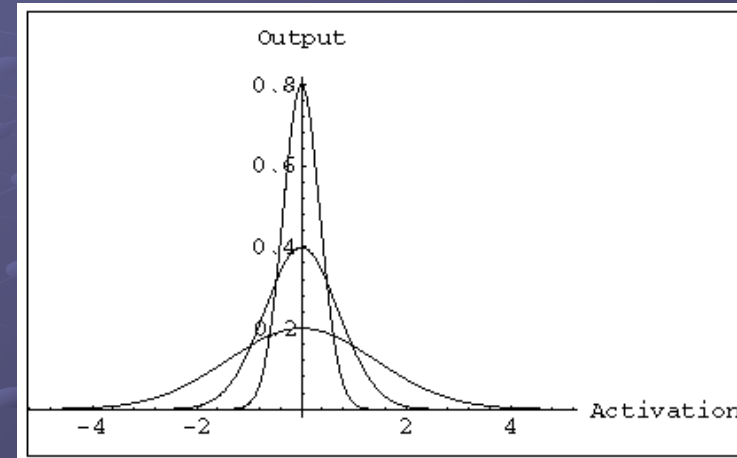
- S. Cho, Chow, C. Leung, “A neural-based crowd estimation by hybrid global learning algorithm”, Systems, Man and Cybernetics, Part B, IEEE Transactions on, Page(s): 535-541

Alternative Activation functions

- Radial Basis Functions

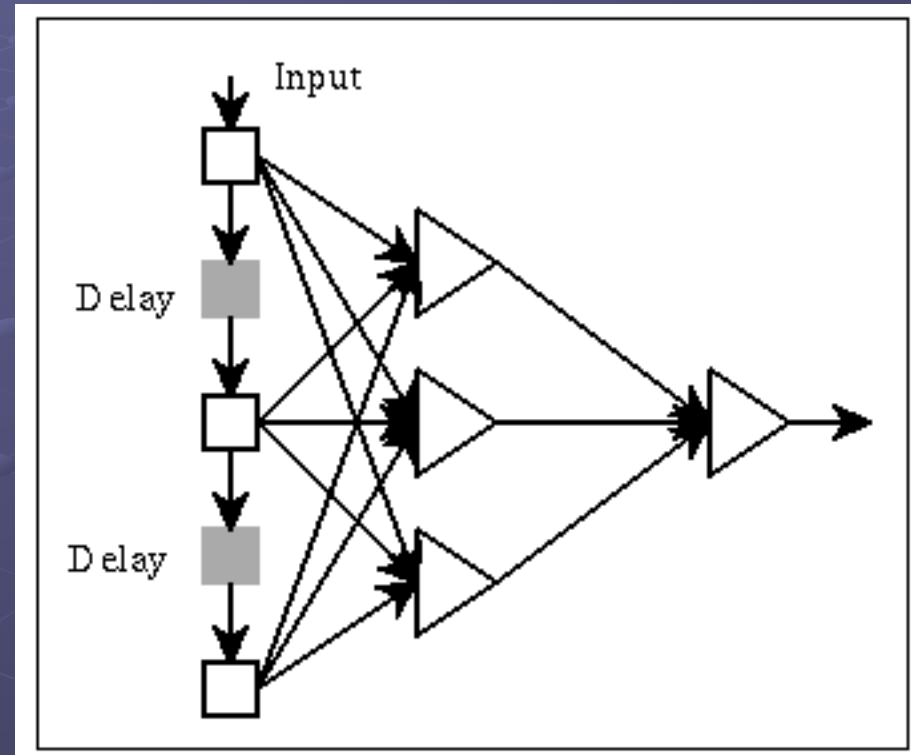
- Square
- Triangle
- Gaussian!

- (μ, σ) can be varied at each hidden node to guide training



Alternate Topologies

- Inputs analyze signal at multiple points in time
- RBF functions may be used to select a 'window' in the input data



Typical Topologies

- Set of inputs
- Set of hidden nodes
- Set of outputs
- Too many nodes makes network hard to train

Supervised Vs. Unsupervised

- Previously discussed networks are 'supervised'
 - Need to be trained ahead of time with lots of data
- Unsupervised networks adapt to the input
 - Applications in Clustering and reducing dimensionality
 - Learning may be very slow

Current Applications

- Investment Analysis
 - Predicting movement of stocks
 - Replacing earlier linear models
- Signature Analysis
 - Bank Checks, VISA, etc.
- Process Control
 - Chemistry related
- Monitoring
 - Sensor networks may gather more data than can be processed by operators
 - Inputs: Cues from camera data, vibration levels, sound, radar, lydar, etc.
 - Output: Number of people at a terminal, engine warning light, control for light switch

Current Work

- Work of S. Cho, Chow and C. Leung
- Training Neural Network to approximate crowd size based on image cues
- Network approximates a non linear mapping between input and output space
- Their input cues: Black Pixels, White Pixels, Edge Pixels
- Our proposed inputs: Edge Histogram, Blob Size histogram (based on background model)



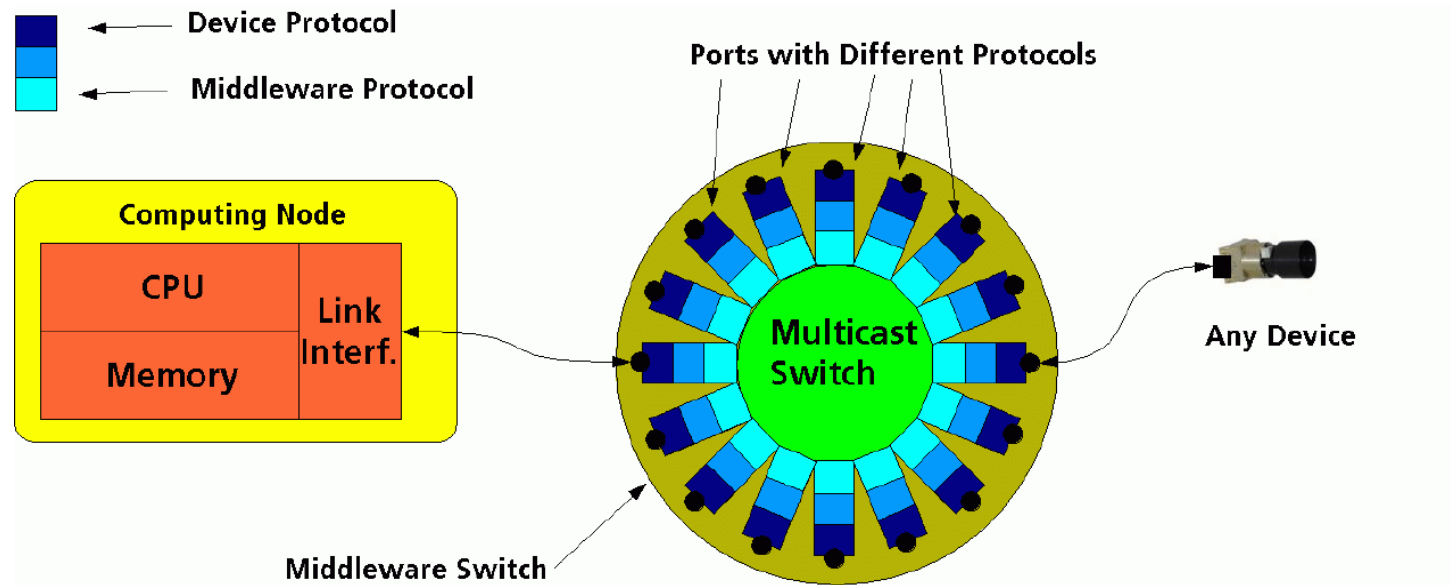
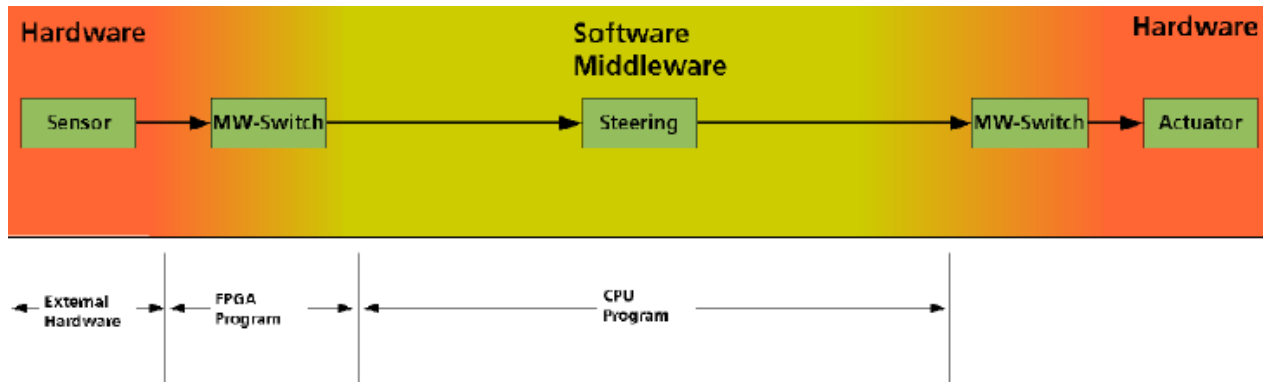
Middleware Switch

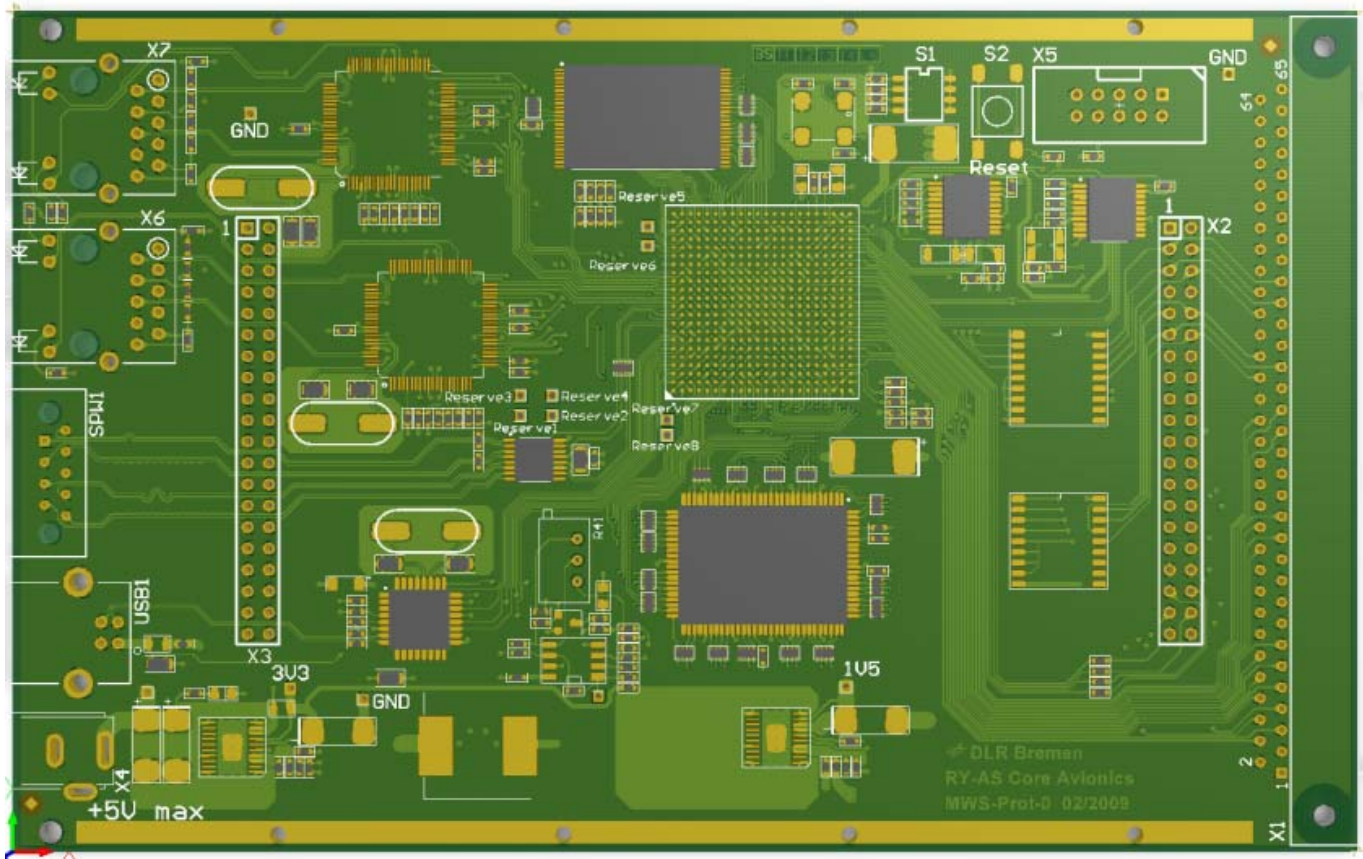
Muhammad Faisal
German Aerospace Center



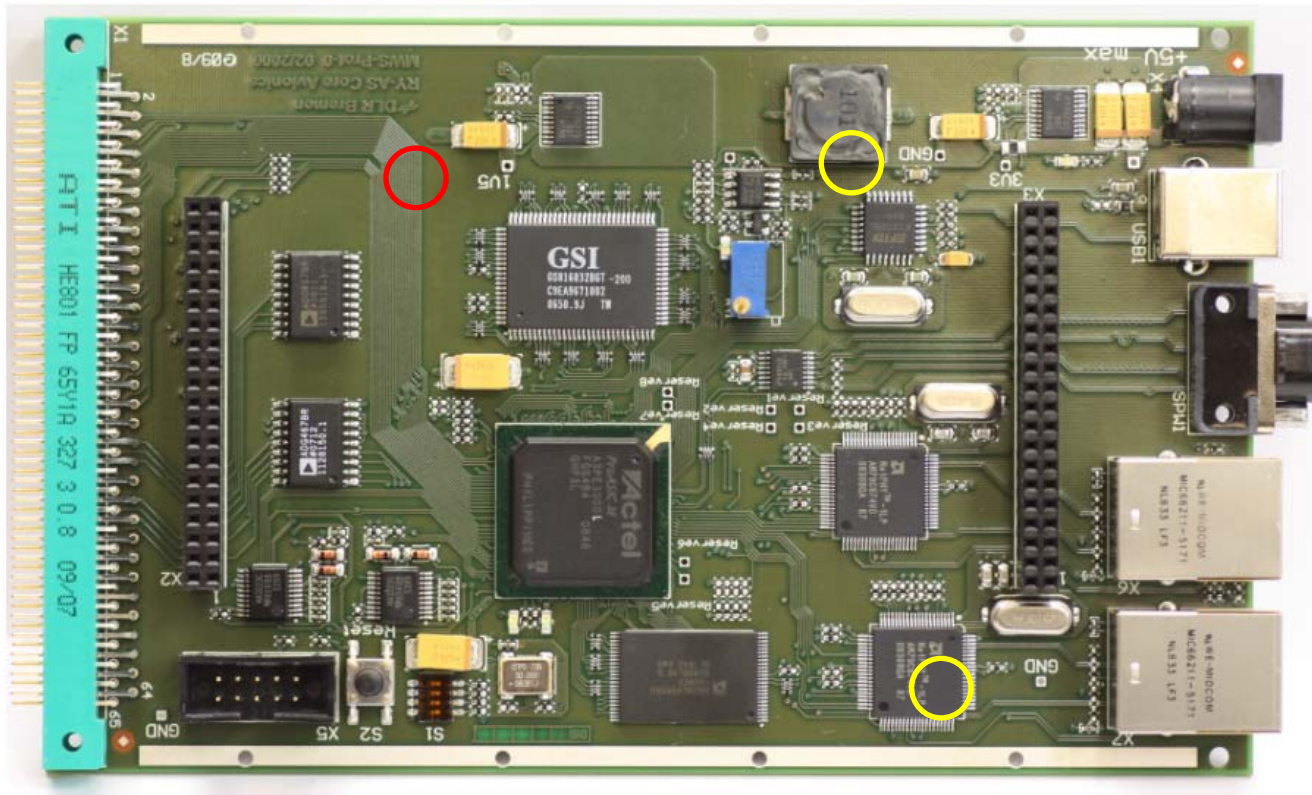
Middleware Switch

- **History**
- Need for the Core avionic System in Standard Satellite Bus (SSB) .
- Start of *Compatksatellit* project.

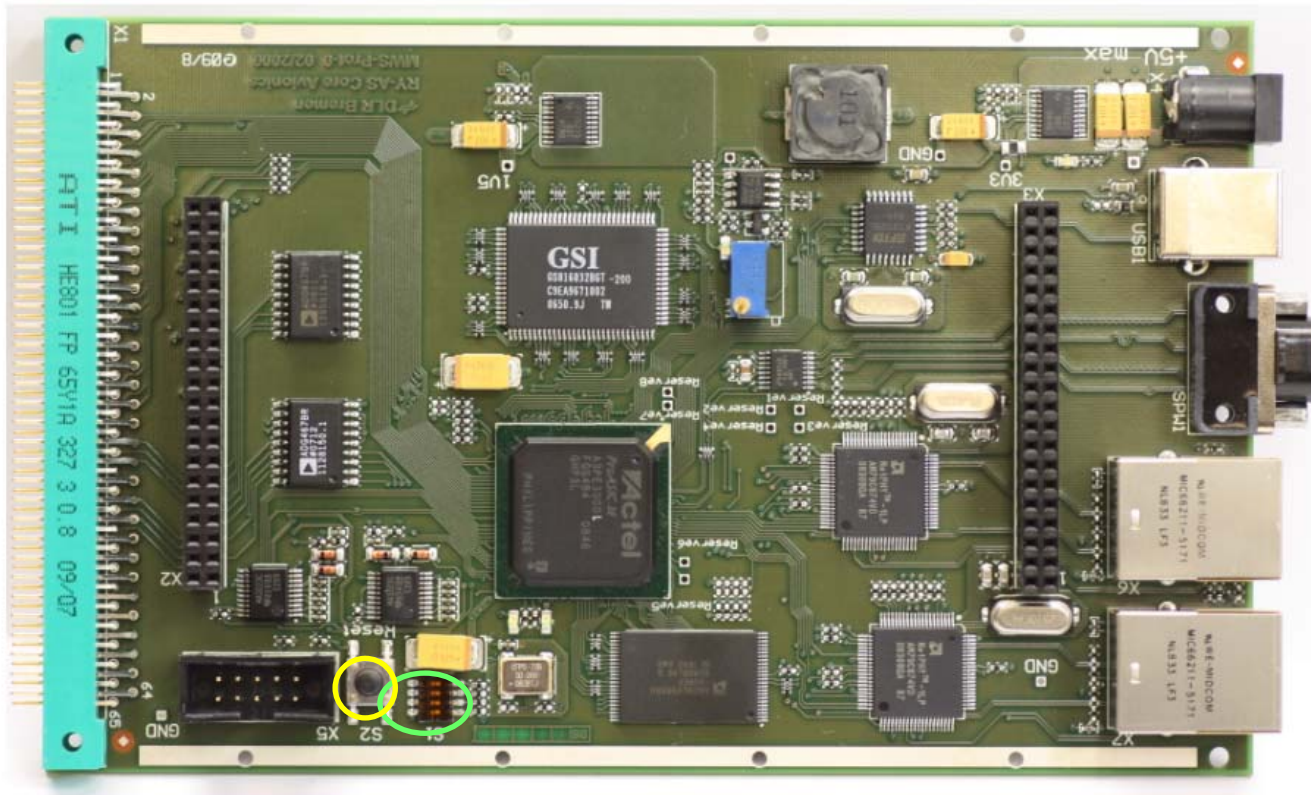




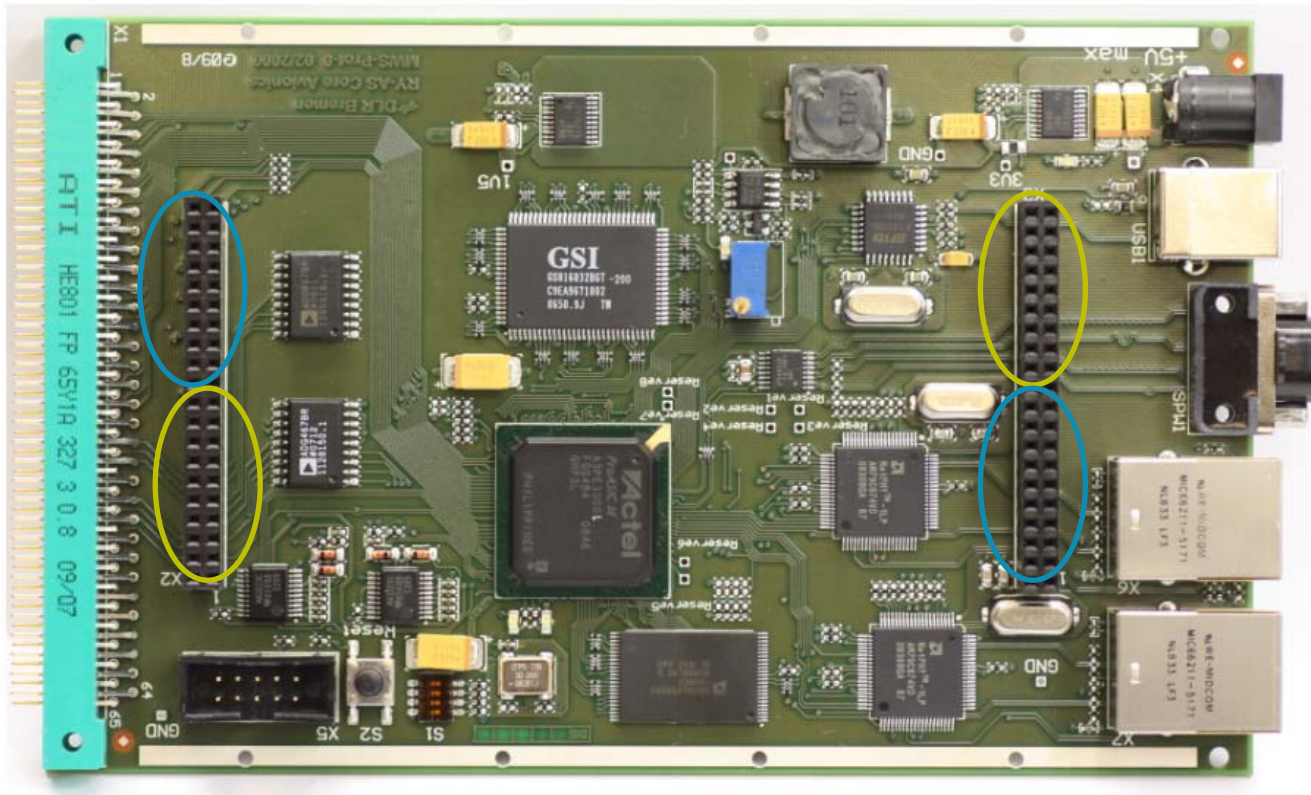
Testing of the Middleware Switch board



Testing of the Middleware Switch board



Testing of the Middleware Switch board



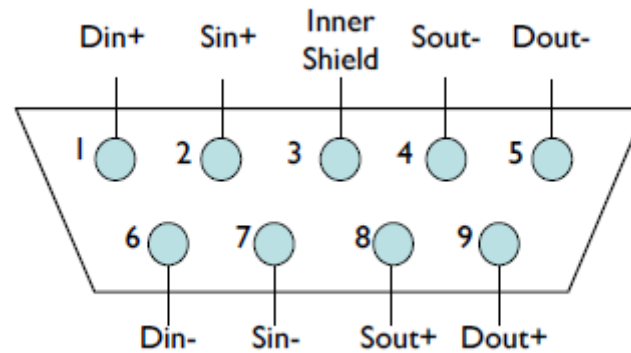
SpaceWire Port



SpaceWire Protocol

SpaceWire is a standard for high-speed links and networks for use onboard spacecraft, easing the interconnection of: enormous mass-memory processing units, and downlink telemetry sub-systems.

SpaceWire Protocol

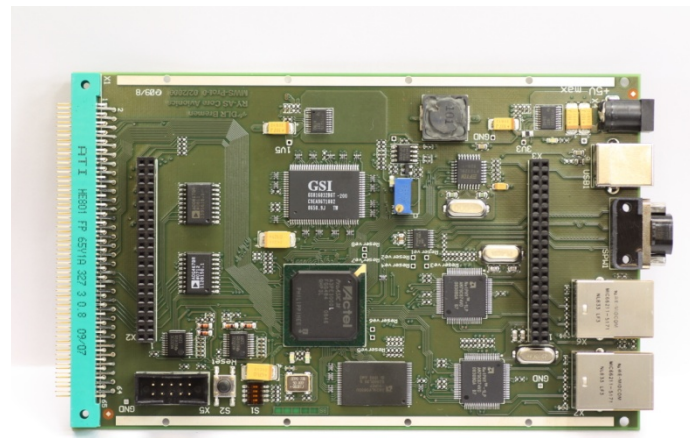


Spacewire

- . SpaceWire is defined in the European Cooperation for Space Standardization [ECSS-E50-12A standard](#)

SpaceWire Protocol

- SpaceWire is being widely used on many space missions by:
 - [ESA](#)
 - [NASA](#)
 - [JAXA](#)
- SpaceWire equipment is connected together using SpaceWire links which are:
 - serial,
 - high-speed (2 Mbits/sec to 200 Mbits/sec),
 - bi-directional,
 - full-duplex.







Residue-to-Decimal Converters for Moduli Sets With Common Factors

Kazeem Gbolagade^{1,2} and Sorin Cotofana²

1. Computer Sc. Dept., UDS, Navrongo, Ghana

2. Computer Engineering Laboratory,

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology (TU Delft), The Netherlands.





Presentation Overview

- Introduction
- Proposed Algorithm
- Performance Evaluation
- Conclusions



Residue Number Systems

- The basis for an RNS is a set of relatively prime integers $\{m_1, m_2, m_3, \dots, m_n\}$ where $\gcd(m_i, m_j) = 1$ for $i \neq j$ and $M = \prod_{i=1}^n m_i$
- $X \longrightarrow [0, M-1] \Longrightarrow X = (x_1, x_2, x_3, \dots, x_n)$ where $x_i = |X|_{m_i}$
- Suppose $K = (k_1, k_2, k_3, \dots, k_n)$ and $L = (l_1, l_2, l_3, \dots, l_n)$ and \ominus rep. operation of $+, -, \& *$
- If $W = (w_1, w_2, w_3, \dots, w_n)$
and $W = K \ominus L \Longrightarrow w_i = |k_i \ominus l_i|_{m_i}$



RNS Advantages and Challenges

- RNS advantages:
 - Carry free addition/subtraction,
 - Parallelism,
 - Error detection and correction.
- RNS challenges:
 - Sign detection,
 - Magnitude comparison,
 - Overflow detection,
 - Division and other complex operations,
 - Moduli Selection,
 - Residue to Binary/Decimal Conversion and vice-versa.



RNS to Decimal Conversion

Conventional Methods

Chinese Remainder Theorem
(CRT)

Large Modulo M

Mixed Radix Conversion
(MRC)

Naturally Serial

Research Question: Given the moduli set $\{2^{n+2}, 2^{n+1}, 2^n\}$, can we select a moduli set such that CRT can be performed with less expensive modulo operations? Given that larger dynamic range is of practical interest, can we extend the moduli set $\{2^{n+2}, 2^{n+1}, 2^n\}$ to $\{2^{n+3}, 2^{n+2}, 2^{n+1}, 2^n\}$?



Proposed Conversion Method

Assume the moduli set $\{2n+2, 2n+1, 2n\}$.

Strategy:

- Eliminate the computation of multiplicative inverses,
- Simplify the resulting CRT,
- Eliminate the explicit calculation of the modulo operation,
- Possible extension to $\{2n+3, 2n+2, 2n+1, 2n\}$.



Elimination of the Multiplicative Inverses

Given the moduli set $\{2n+2, 2n+1, 2n\}$ with $m_1=2n+2$, $m_2=2n+1$, $m_3=2n$, we obtain the following :

$$\left| (m_1 m_2)^{-1} \right|_{\frac{m_3}{2}} = \frac{n+1}{2}, \quad (1)$$

$$\left| \left(m_2 \frac{m_3}{2} \right)^{-1} \right|_{m_1} = n+2, \quad (2)$$

$$\left| \left(m_1 \frac{m_3}{2} \right)^{-1} \right|_{m_2} = 2n-1. \quad (3)$$



CRT Simplification (1)

For moduli set sharing a common factor, it has been proved in (Wang, 1999) that (x_1, x_2, x_3) represents a valid number if and only if $(x_1 + x_3)$ is even.

The decimal equivalent of the RNS number (x_1, x_2, x_3) for the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2n+2, 2n+1, 2n\}$ is computed for $(x_1 + x_3)$ even, as follows:

$$X = \left| \frac{m_2 m_3}{2} x_1 - m_1 m_3 x_2 + \frac{m_1 m_2}{2} x_3 \right|_{M_L} \quad (4)$$



CRT Simplification (2)

Using the lemma $|am_1|_{m_1m_2} = m_1|a|_{m_2}$ presented in (Wang, '99), we further simplify the equations and we obtain for X positive and negative, respectively, the following:

$$X = (x_2 - x_1)m_1 + x_1 + m_1m_2 \left| \frac{(x_1 + x_3)}{2} - x_2 \right|_{\frac{m_3}{2}} \quad (5)$$

$$X = (x_2 - x_1)m_1 + x_1 + m_1m_2 \left(\left| \frac{(x_1 + x_3)}{2} - x_2 \right|_{\frac{m_3}{2}} + \frac{m_3}{2} \right) \quad (6)$$



Extension: $\{2n+3, 2n+2, 2n+1, 2n\}$

The decimal equivalent of the RNS number (x_1, x_2, x_3, x_4) for the moduli set $\{m_1, m_2, m_3, m_4\}$ in the form $\{2n+3, 2n+2, 2n+1, 2n\}$ is computed as follows:

$$X = (x_1 + x_2 + x_3) + m_1 m_2 m_3 \left| k_1 x_1 + k_2 x_2 + k_3 x_3 + \left| M_4^{-1} \right|_{m_4} x_4 \right|_{m_4}$$

k_1 , k_2 , and k_3 can be pre-computed.

This proposal is better than the traditional CRT.



Low Cost Modulo Calculation

We summarize the 4 possibilities as follows:

- If the tentative sum is $< -\frac{m_3}{2}$, add m_3 ;
- If the tentative sum is $\geq -\frac{m_3}{2}$, add $\frac{m_3}{2}$;
- If tentative sum is zero and $x_1 > x_2$, add $\frac{m_3}{2}$;
- Otherwise, do nothing.

In this way all the modulo operations have been replaced by a single corrective addition or subtraction greatly reducing the complexity of the converter.



Performance Evaluation

Area-delay comparison:

Metrics	Wang, et al , '99	Our proposal
Area	4 adders, 2 multipliers	4 adders, 2 multipliers
Delay	3 additions 2 multiplications	3/4 additions 1 multiplications
Mod Operations	$m_1 m_3$	m_3

- Apparently similar area and smaller delay,
- Simpler modulo operation.



Performance Evaluation (1)

Synthesized results: Area-delay comparison

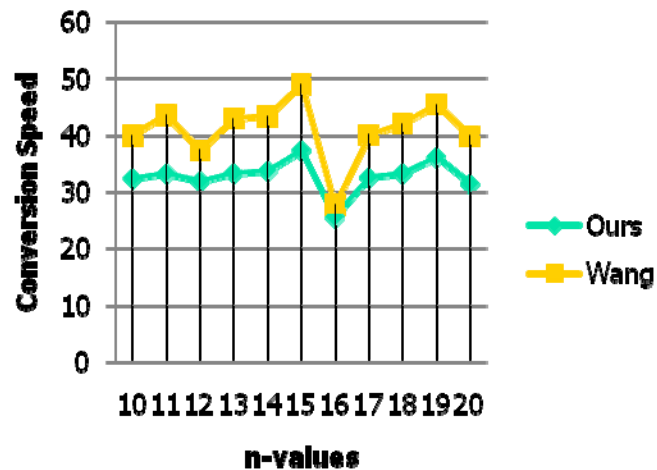
n	Our Area	Area (Wang)	Our Delay	Delay (Wang)
10	56	80	32.524	40.042
11	78	107	33.372	43.683
12	56	78	31.978	37.454
15	88	116	37.424	49.108
20	64	93	31.545	39.9

- On average, our proposal is 20% faster than Wang's converter,
- On average, our proposal requires 29% lesser area.

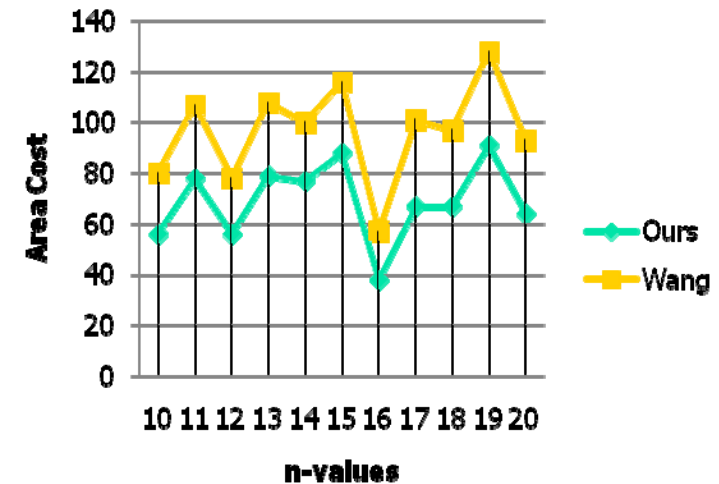


Performance Evaluation (2)

Delay Comparison



Area Comparison

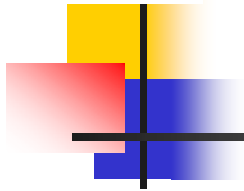


- The figures show that the proposed converter outperforms Wang's proposal in terms of both area and delay.



Conclusions

- We presented a new algorithm for RNS to Decimal conversion using the moduli set $\{2n+2, 2n+1, 2n\}$.
- We eliminated the computation of multiplicative inverses.
- We further simplified the resulting CRT in order to obtain a low complexity implementation which does not require the explicit utilization of modulo operation in the conversion process.
- The proposed scheme is shown to be better in terms of both area cost and conversion delay when compared to the best known equivalent state of the art converters.



THANK
YOU!



MINIMIZATION OF REVERSIBLE LOGIC CIRCUITS

Md. Saiful Islam

Lecturer, Institute of Information Technology

University of Dhaka

Dhaka-1000, Bangladesh

CONTENTS

- Reversible Logic
 - Motivation
 - Reversible Computing
 - Applications
- Reversible Gates
 - Basic Reversible Gates
 - Feynman, Toffoli, Fredkin, Peres and F2G gate
- Properties of Reversible Network
- Reversible Full-Adder: An Example
- Toffoli Gate: An Example
- Complexity Analysis
- Reversible Logic Minimization
- Generalized Algorithm: An Example

MOTIVATION TOWARDS REVERSIBLE LOGIC

November 4, 2009

3

MOTIVATION

Getting more and more information...

What about Nizar...?



You may be overheated and lose your hair ...uh, its too hot to keep



Don't keep it, lets them go out...um, now its cool

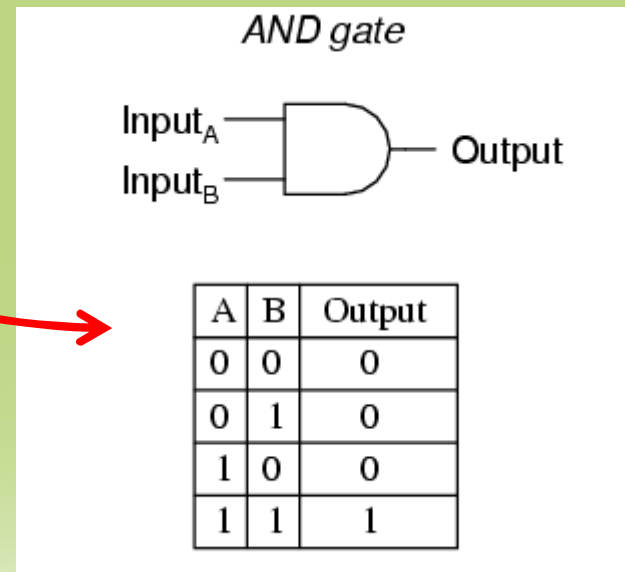
November 4, 2009

MOTIVATION(CONTD...)

Landauer 's Principle:

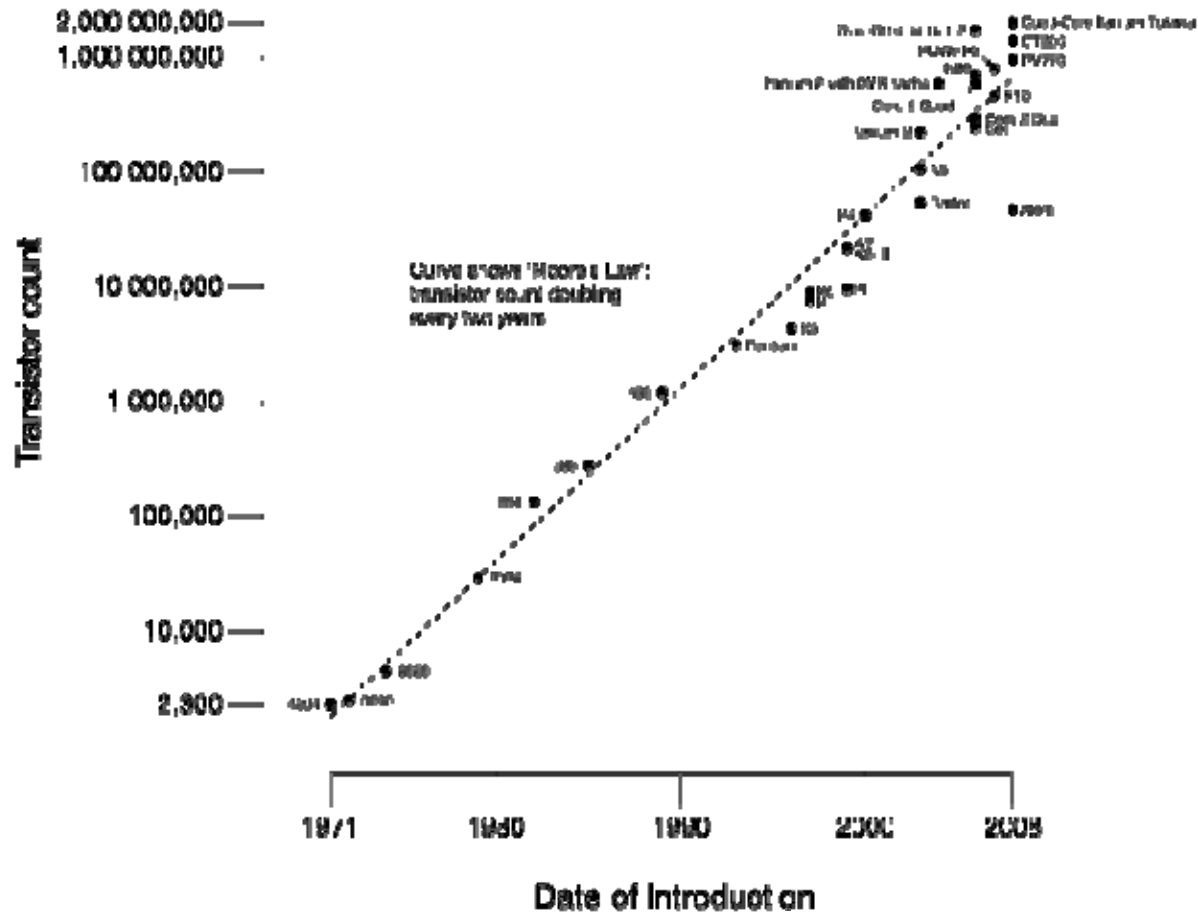
$kT \ln 2$ Joule for every bit of information lost

Where k is the Boltzmann Constant (1.38×10^{-23} J/K)
 T is the absolute temperature



MOTIVATION(CONTD...)

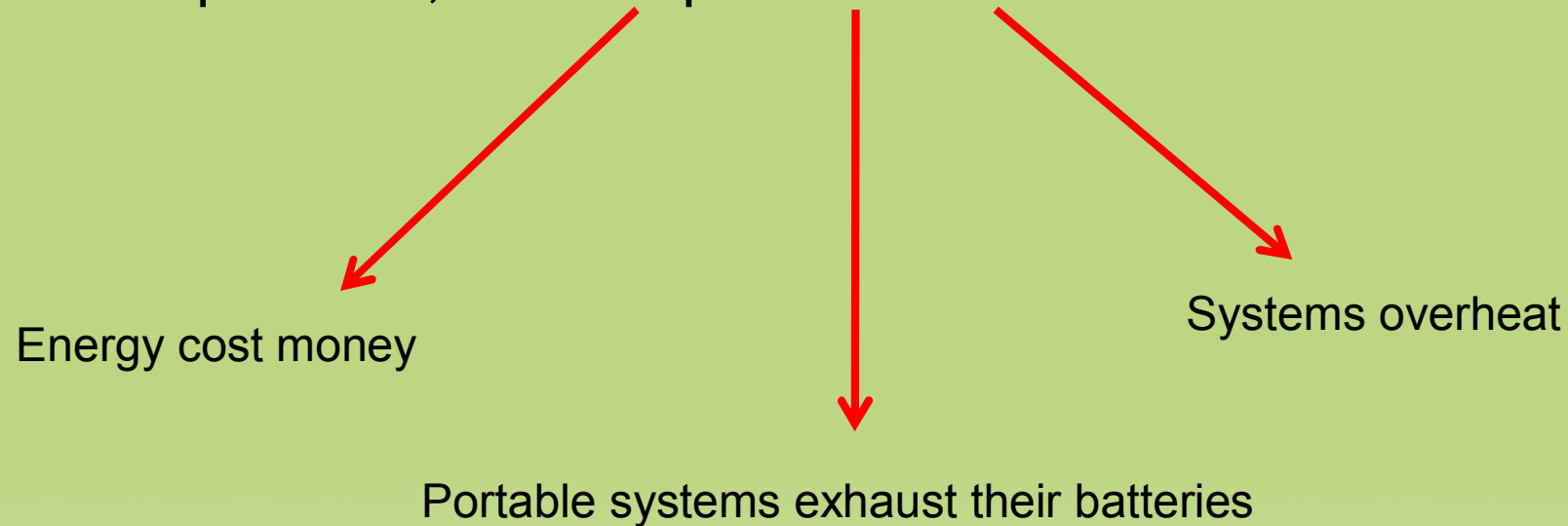
CPU Transistor Counts 1971-2008 & Moore's Law



November 4, 2009

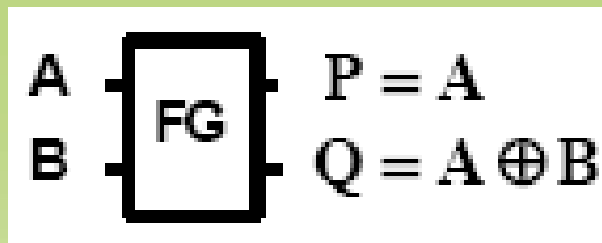
MOTIVATION(CONTD...)

As we pack more and more logic elements into smaller and smaller volumes and clock them at higher and higher frequencies, we dissipate more and more heat.



REVERSIBLE COMPUTING

- Also known as non-destructive computing
- Implements bijective function
- One-to-one mapping
- Every input vector will be mapped to a unique output vector



A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

APPLICATIONS

- Cryptography
- Digital Signal Processing
- DNA Computing
- Optical Information Processing
- Nanotechnology
- Low Power CMOS Design, etc.

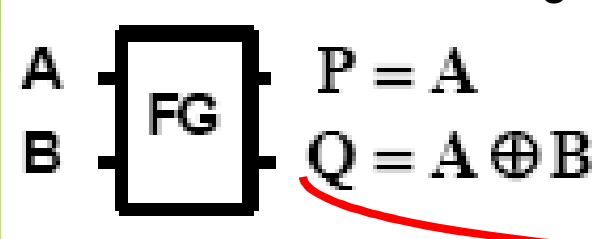
REVERSIBLE GATES

November 4, 2009

10

BASIC REVERSIBLE GATES

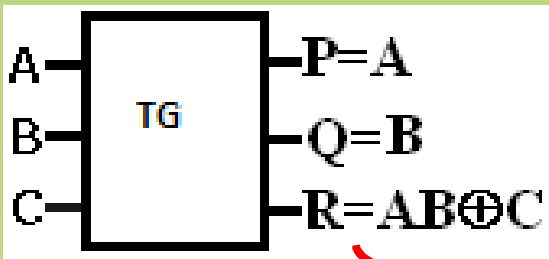
One-through gate...



A	B	P	Q
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Feynman Gate (1985)

Two-through gate...



A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Toffoli Gate (1980)

BASIC REVERSIBLE GATES (CONTD...)

ICTP is boring, no dance, no party...

We wish to dance and have party...cheers

The lady is dancing controlled by the gentleman...

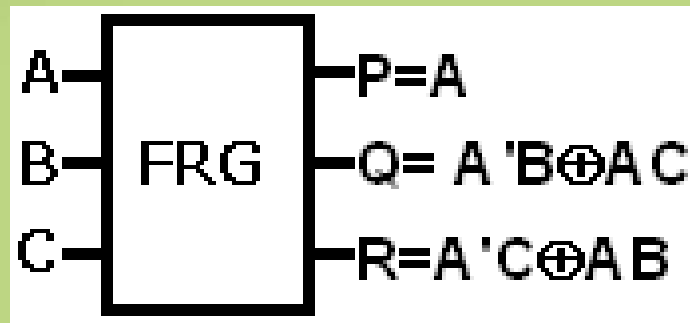


November 4, 2009

12

BASIC REVERSIBLE GATES (CONTD...)

One-through gate...



A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

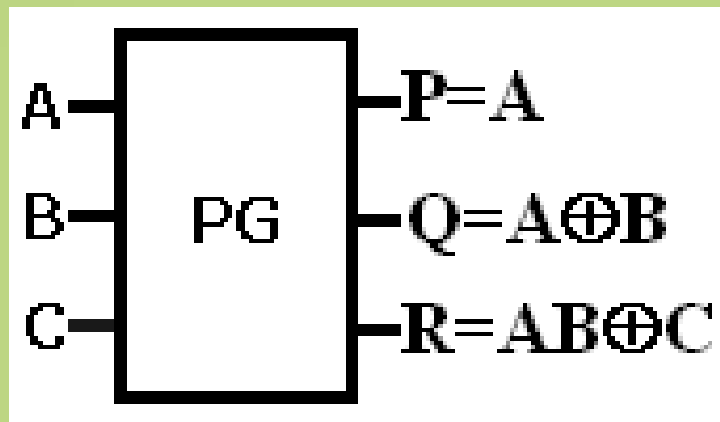
The gentle man (active)...

Fredkin gate (1982)

The lady is changing her hands...(exchange)

BASIC REVERSIBLE GATES (CONTD...)

One-through gate...

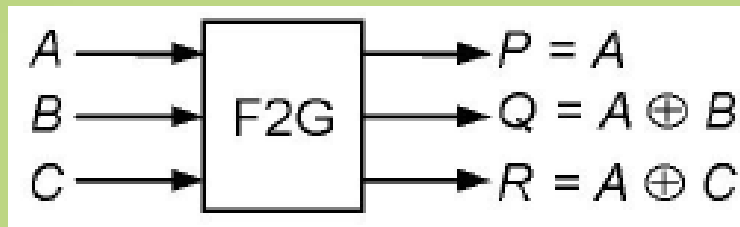


A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Peres gate (1985)

BASIC REVERSIBLE GATES (CONTD...)

One-through gate...



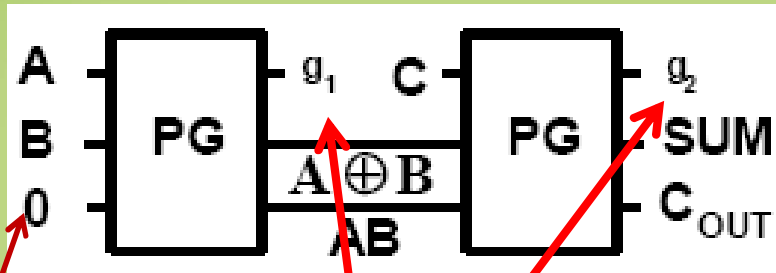
Feynman Double gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

PROPERTIES

- No. of input lines must be equal to no. of output lines.
- No fan-out is allowed
- No feedback
- Networks include only **Reversible** gates

REVERSIBLE FULL ADDER: AN EXAMPLE

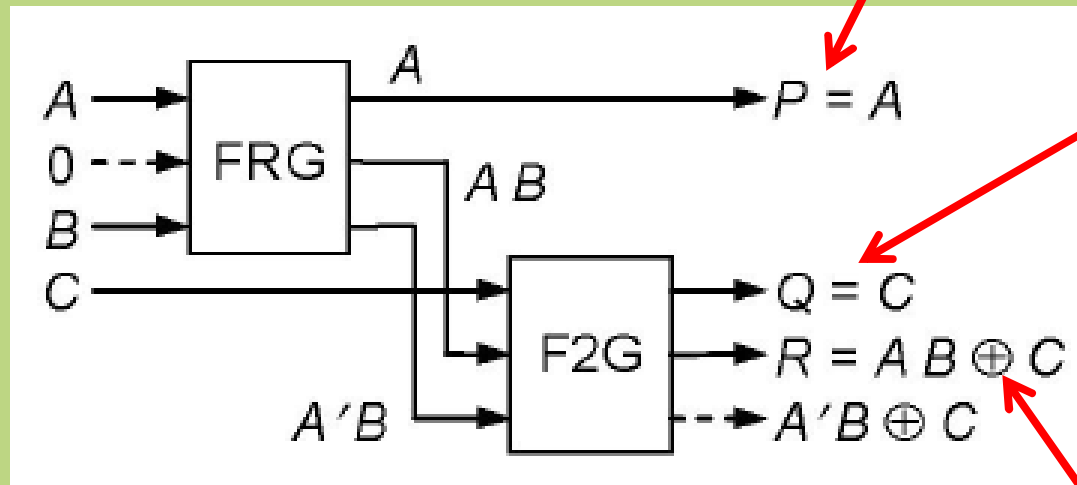


Reversible Full-Adder utilizes only two Peres gate

Constants

Garbages

TOFFOLI GATE: AN EXAMPLE (CONTD...)



Toffoli gate from other Reversible gates...

COMPLEXITY ANALYSIS



Beautiful



Sweet



Charming



Many more...

He is too bad...



I don't like her with this guy anymore...

November 4, 2009

20

COMPLEXITY ANALYSIS (CONTD...)

- ❑ **Garbage Outputs (GO):** outputs that are not used as primary outputs are termed as garbage outputs
- ❑ **Constant Inputs (CI):** constants are the input lines that are either set to zero(0) or one (1) in the circuit's input side
- ❑ **Gate Count (GC):** number of gates used to realize the system
- ❑ **Hardware Complexity (HC):** refers to the number of basic gates (NOT, AND and EXOR gate) used to synthesize the given function

COMPLEXITY ANALYSIS (CONTD...)

- **Area Consumption (AC):** the area consumed by a design can be calculated as the sum of the width of individual gates used to realize the system.

$$AC = \sum_{i=1}^n w_i$$

- **Path Delay (PD):** total number of gates on the way of final output

REVERSIBLE LOGIC MINIMIZATION

November 4, 2009

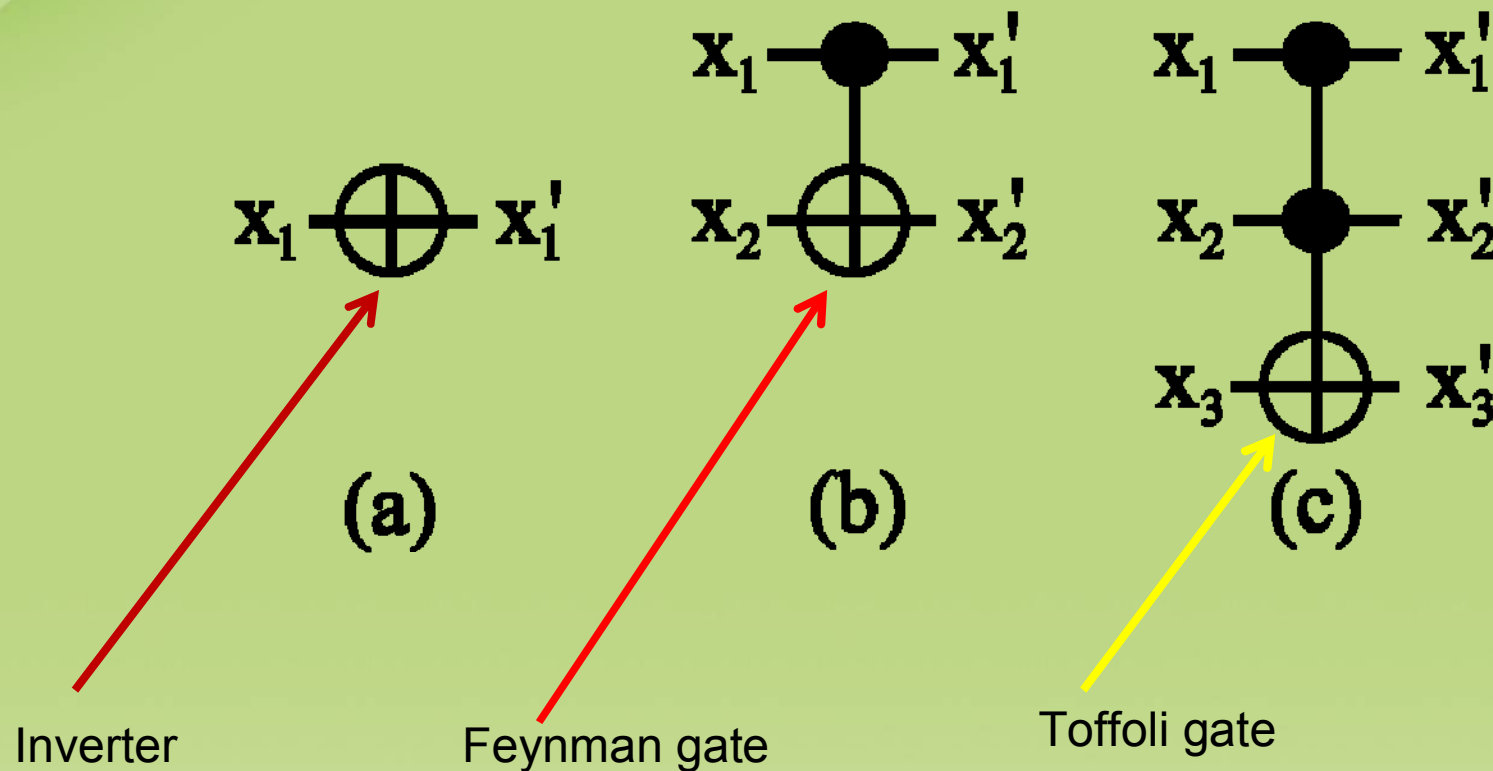
23

REVERSIBLE LOGIC MINIMIZATION

To synthesize a given function

- ✓ Minimize the Gate Count
- ✓ Use minimum no. of Constant Inputs
- ✓ Reduce the no of Garbage Outputs produced
- ✓ Reduce Hardwire Complexity as much as possible
- ✓ Minimize Area Consumption
- ✓ Reduce the overall Path Delay

GENERALIZED ALGORITHM: TRANSFORMATION-BASED



We use such gates for any number of inputs

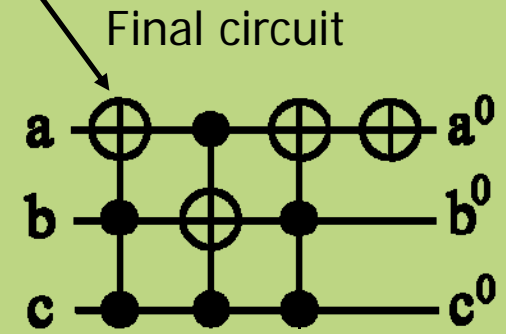
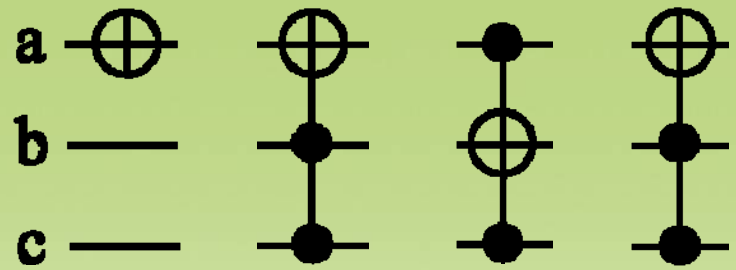
GENERALIZED ALGORITHM: TRANSFORMATION-BASED

- Reversible function is given as a truth table
- For **each row** in the truth table
 - Introduce Toffoli gates such that the **output equals the input**
 - Make sure that **previous outputs are not affected**

TRANSFORMATION-BASED (CONTD...)

Observe that after applying S3 wires are as in input

in	out	S1	S2	S3	S3
000	001	000	000	000	000
001	000	001	001	001	001
010	011	010	010	010	010
011	010	011	011	011	011
100	101	100	100	100	100
101	111	110	111	101	101
110	100	101	101	111	110
111	110	111	110	110	111



TRANSFORMATION-BASED (CONTD...)

BIDIRECTIONAL

in	out
000	111
001	000
010	001
011	010
100	011
101	100
110	101
111	110

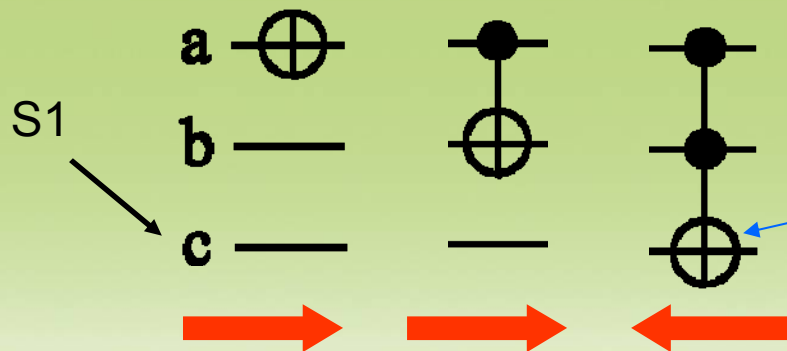
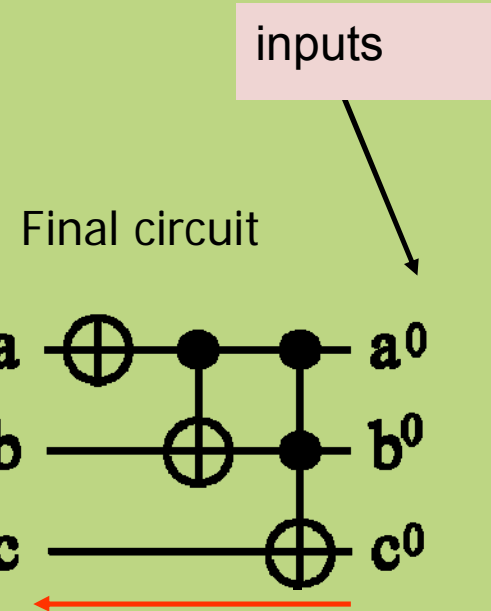
need three gates

need only one gate

TRANSFORMATION-BASED (CONTD...)

BIDIRECTIONAL

in	out	S1	S2	S3
000	111	000	000	000
001	000	111	001	001
010	001	010	010	010
011	010	001	111	011
100	011	100	100	100
101	100	011	101	101
110	101	110	110	110
111	110	101	011	111



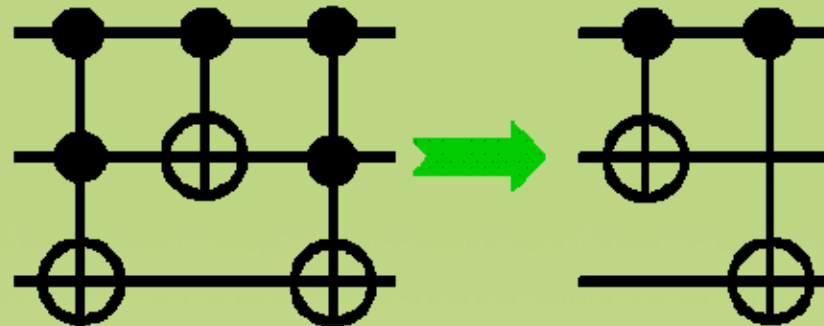
After applying S3 my wires are the same as inputs

TRANSFORMATION-BASED (CONTD...) IMPROVEMENTS

- Output permutations
 - via swap gates
- Control input reduction
 - there may be **more than one possible assignment** of control variables
 - select the one that makes the function **“simple”**
- Template matching

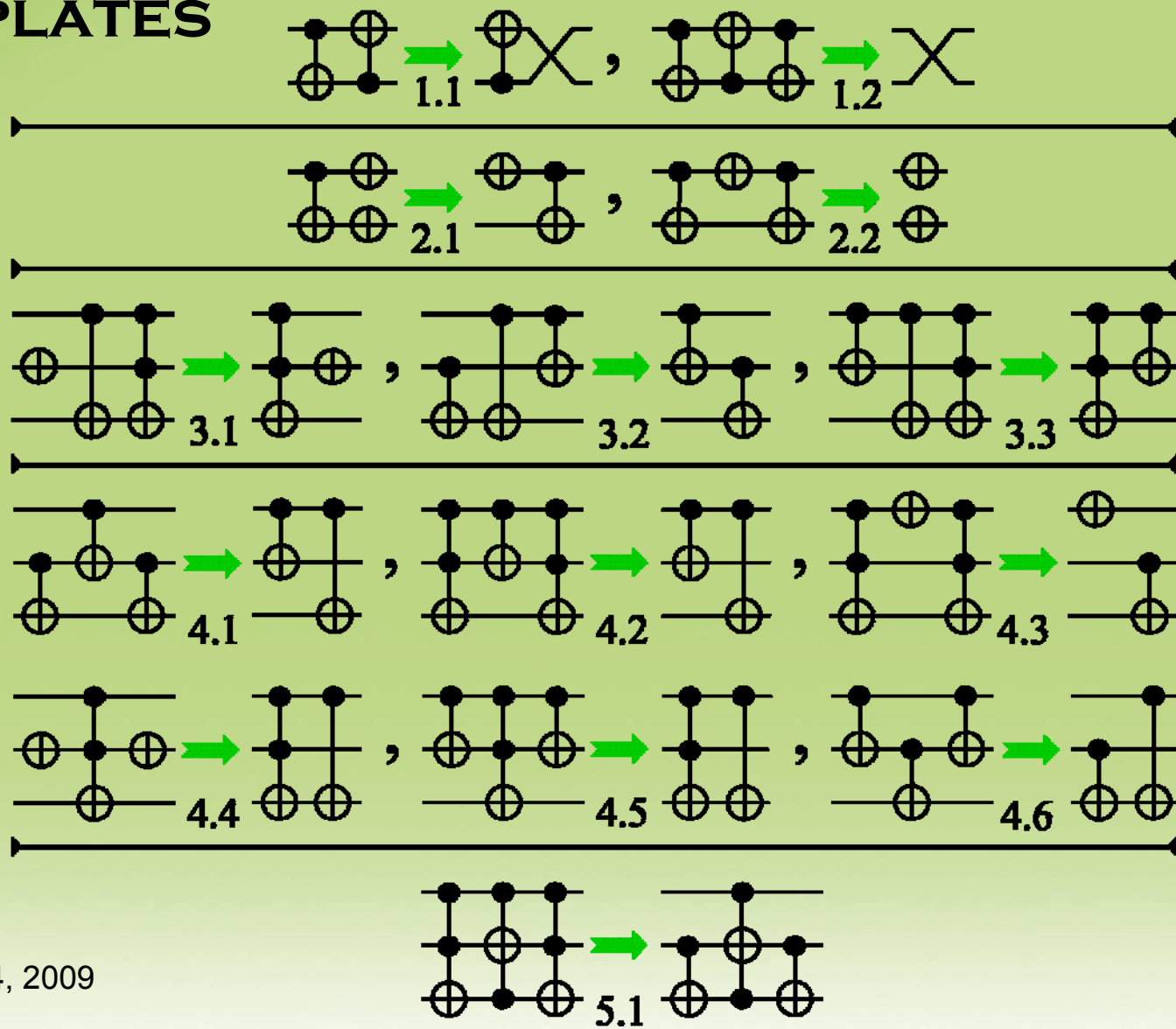
TRANSFORMATION-BASED (CONTD...) TEMPLATES

- Idea: replace a sequence of gates with an equivalent shorter sequence
- example:



TRANSFORMATION-BASED (CONTD...)

TEMPLATES



November 4, 2009

REFERENCES

- **Nizar Abdallah**
http://www.oecd.org/document/31/0,3343,en_21571361_38973579_40151263_1_1_1_1,00.html
- R. Landauer, “Irreversibility and Heat Generation in the Computational Process”, IBM Journal of Research Development, 5,1961, 183-191.
- Bennett, C., “Logical Reversibility of Computation,” IBM Journal of Research and Development, 17, 1973, 525-532.
- R. Feynman, “Quantum Mechanical Computers,” Optics News, Vol. 11, pp. 11–20, 1985.
- T. Toffoli, “Reversible Computing,” Tech memo MIT/LCS/TM-151, MIT Lab for Comp. Sci, 1980.
- E. Fredkin and T. Toffoli, “Conservative logic,” Int’l J. Theoretical Physics, Vol. 21, pp.219–253, 1982.
- Peres, A., 1985. Reversible Logic and quantum computers. Physical Review: A, 32(6): 3266-3276.
- Dmitri Maslov, "Reversible Logic Synthesis", PhD Dissertation, Computer Science Department, University of New Brunswick, Canada, Oct 2003.

November 4, 2009

33

REFERENCES (CONTD...)

- **Md. Saiful Islam**, et. al., Fault Tolerant Reversible Logic Synthesis: Carry Look-Ahead and Carry-Skip Adders, ACTEA 2009, pp. 396-401.
- Ralph C. Merkle, “Two Types of Mechanical Reversible Logic” , <http://www.zyvex.com/nanotech/mechano.html>
- **Md. Saiful Islam** and Md. Rafiqul Islam, “Minimization of Reversible Adder Circuits”, Asian Journal of Information Technology, Vol. 4, No. 12, pp. 1146-1151, 2005.
- B. Parhami, “Fault-Tolerant Reversible Circuits”, Proc. 40th Asilomar Conf. Signals, Systems and Computers, October 2006, Pacific Grove, CA, pp: 1726-1729.
- M. Miller, D. Masolv and G. W. Dueck, “A Transformation Based Algorithm for Reversible Logic Synthesis”, Design Automation Conference, 2003.

**DON'T ASK MUCH...THIS WILL MAKE
ME HAPPY ...!!!!!!!!!!!!!!**

November 4, 2009

35

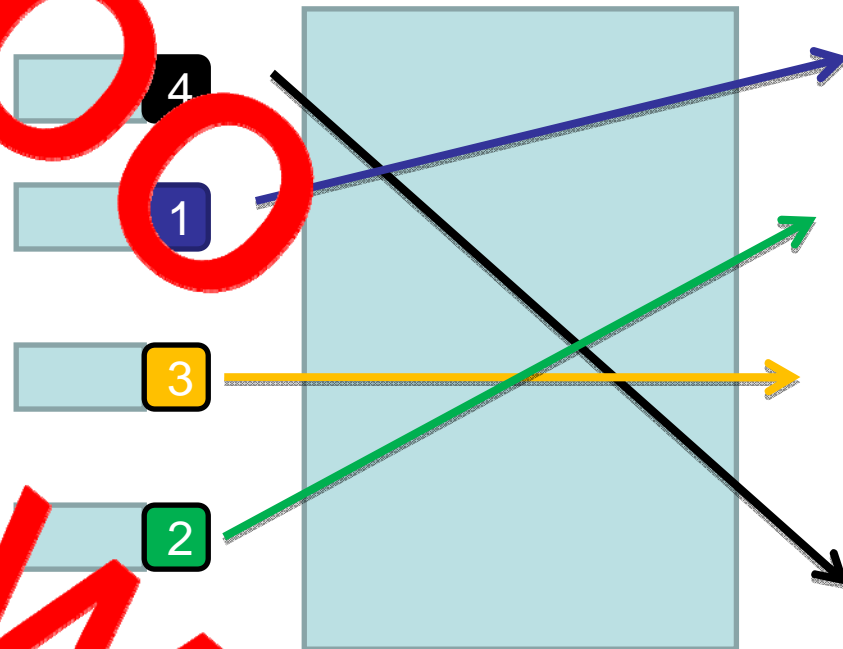
Implementing a Distributed Shared Memory for Switch Fabric on FPGA

Hadi Khani

Router lab, ECE Department, Faculty of Engineering,
University of Tehran²
Tehran, Iran

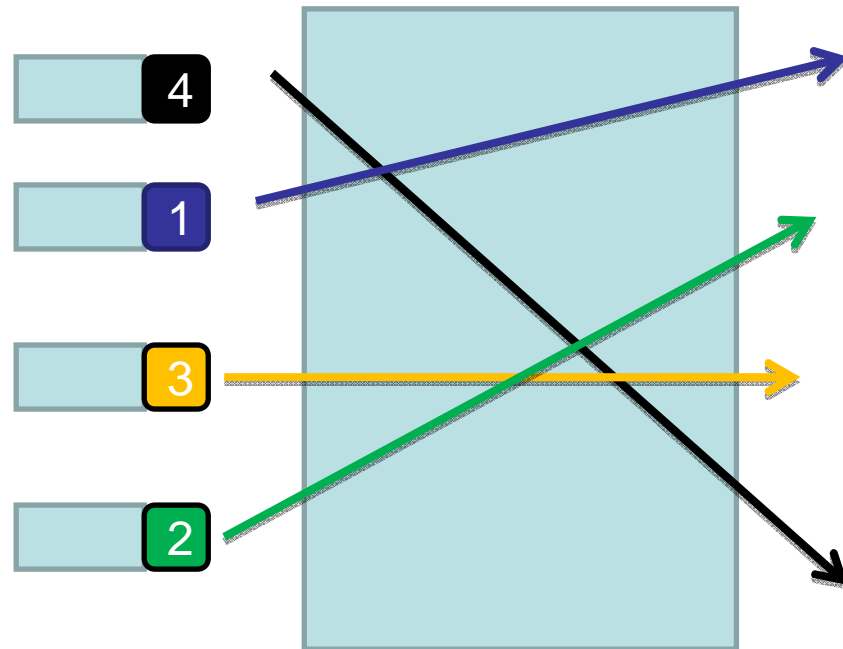
What is Switch Fabric?

- Switch Fabric is a Hardware Module in Network Routers
- Switch Fabric directs a cell to the proper output port according to destination field in the header



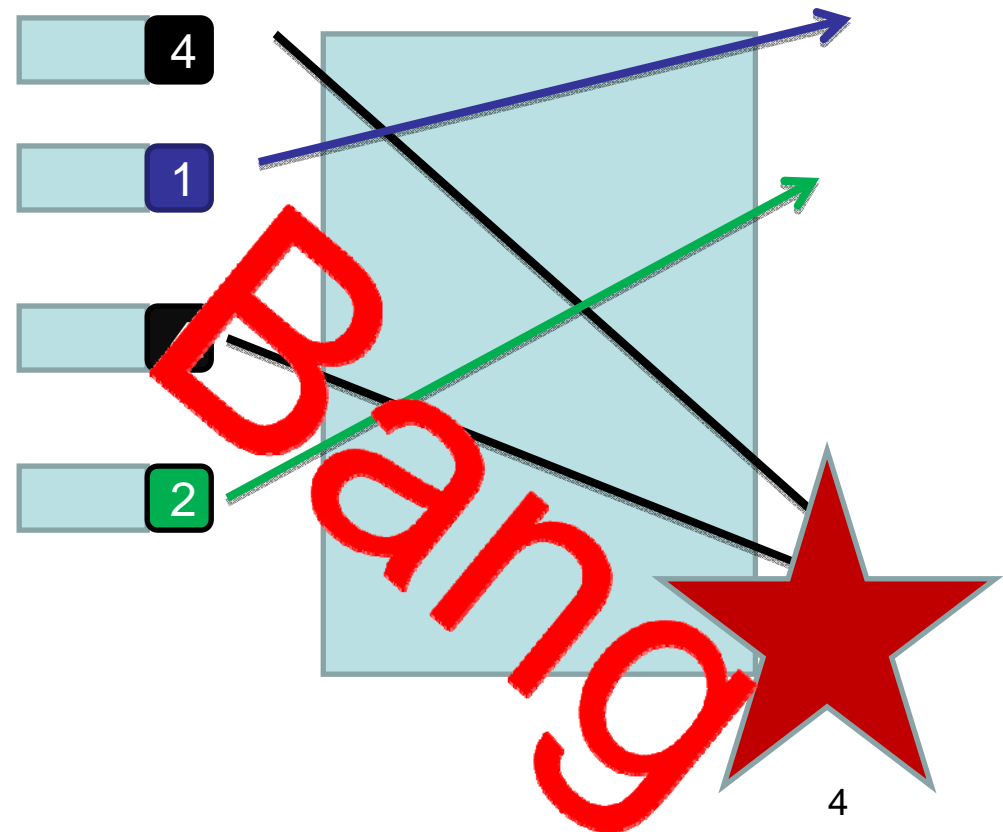
Speed Up

- Problem: If we direct cell in sequence it will be too slow!
- Solution: We can direct all arriving cells
 - At the same time
 - In parallel
 - Concurrently
 - In a Dataflow Manner



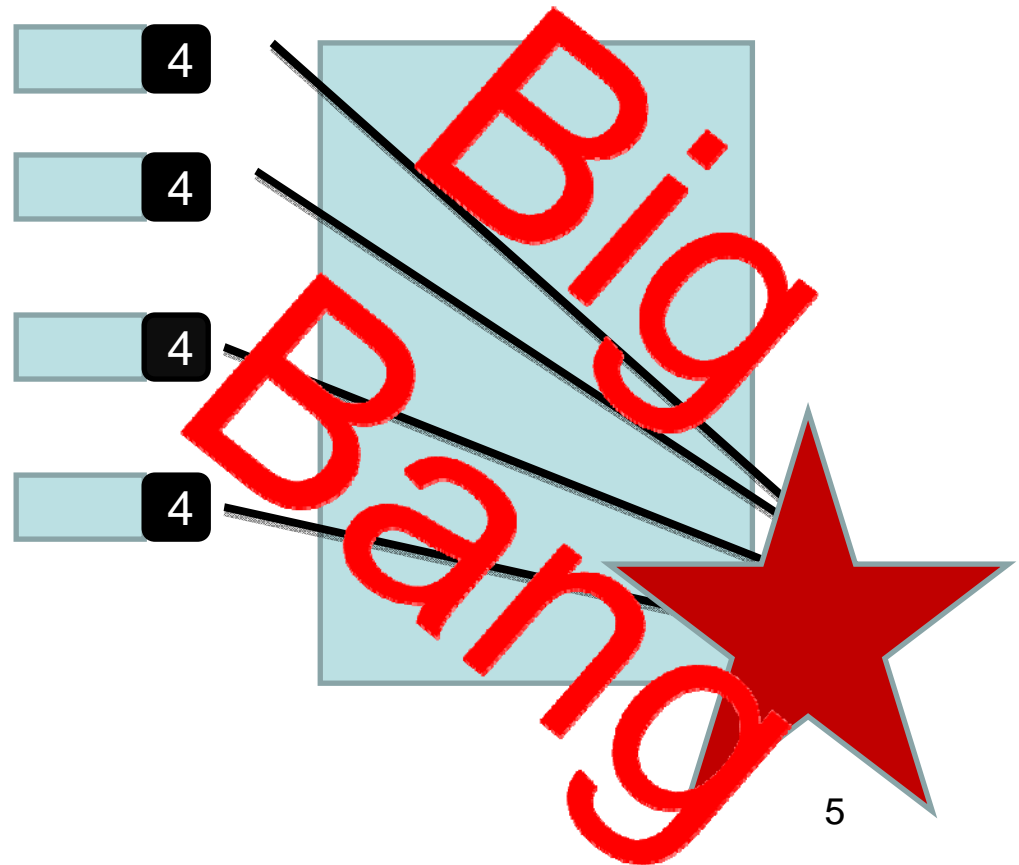
Collision

- Question : what happens if two cells go to one output?



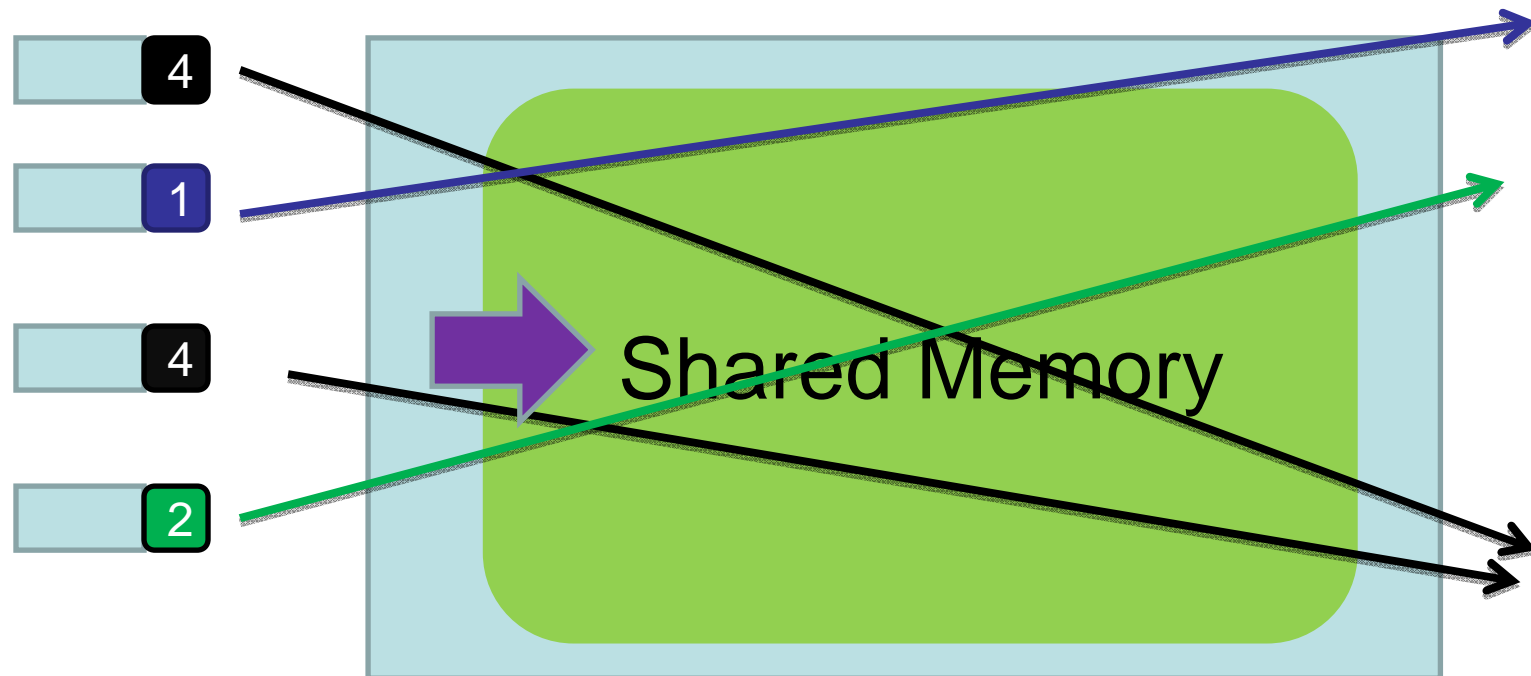
Collision

- And Worse, what happens if all cells go to one output?



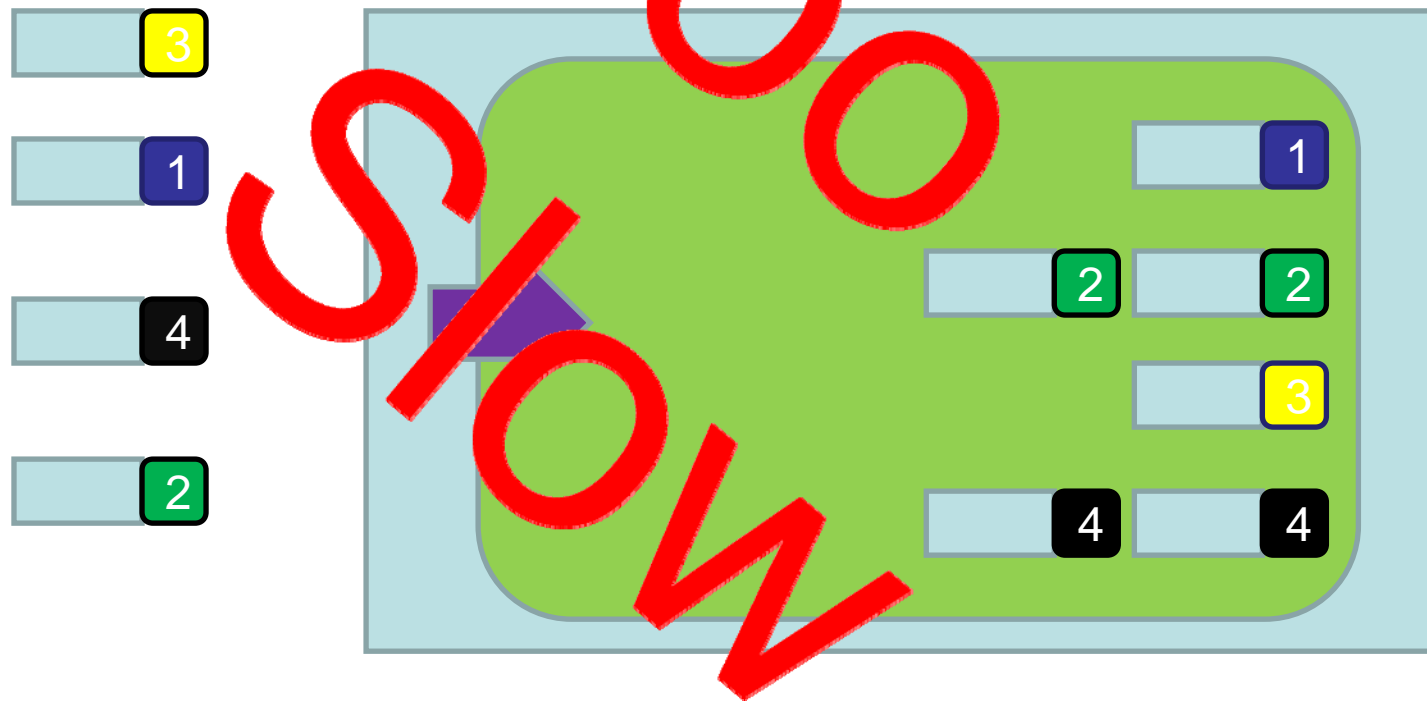
Solution (Queuing)

- Some cells must be stored in a shared memory before going outside



After a while We have queues at every output ports

So every cells must be written into a **shared memory** through one single port **in sequence**



Low Capacity Switch Fabric

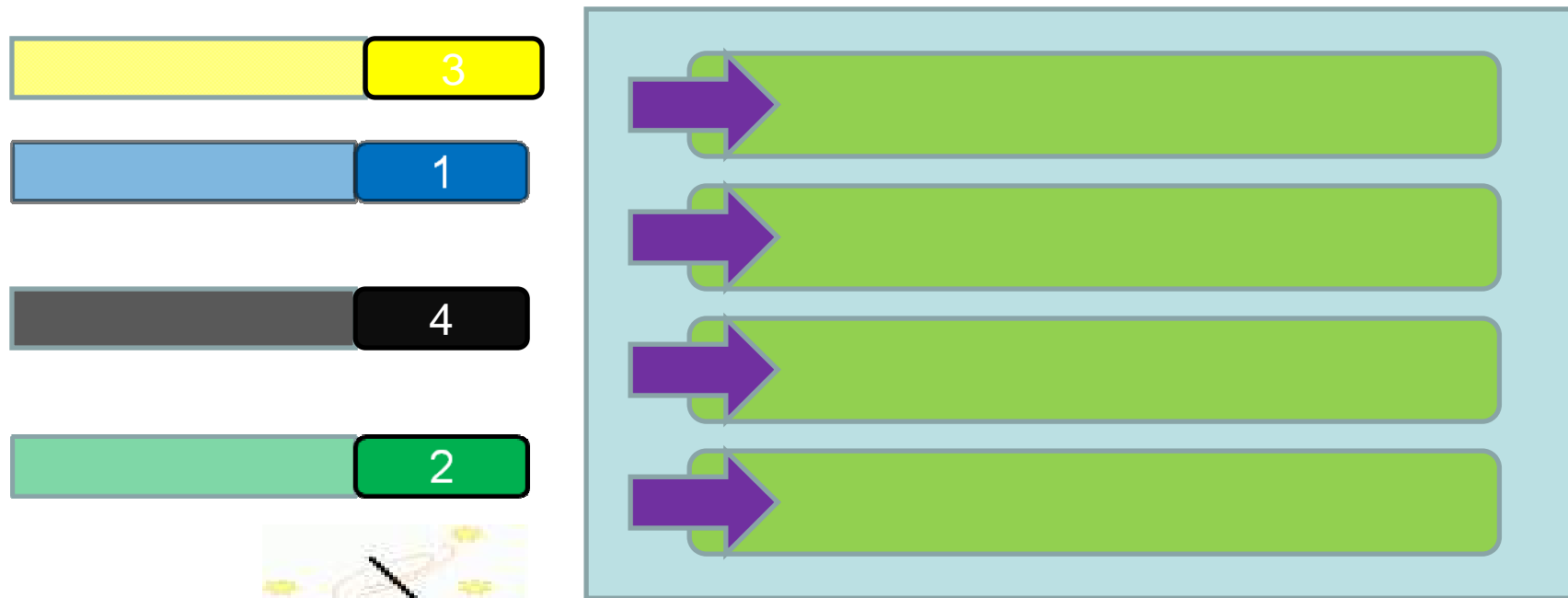
$$\text{Switch Capacity} = \text{Memory Frequency} \times \text{Port Width}$$

- Example
 - Memory port width = 32 bit
 - Memory Access Frequency = 100 MHz
 - Switch Capacity = 3.2 Gbps

Distributed Shared Memory

(instead of a monolithic one)

In order to avoid any collision during writing process we must distribute cell to all memory banks

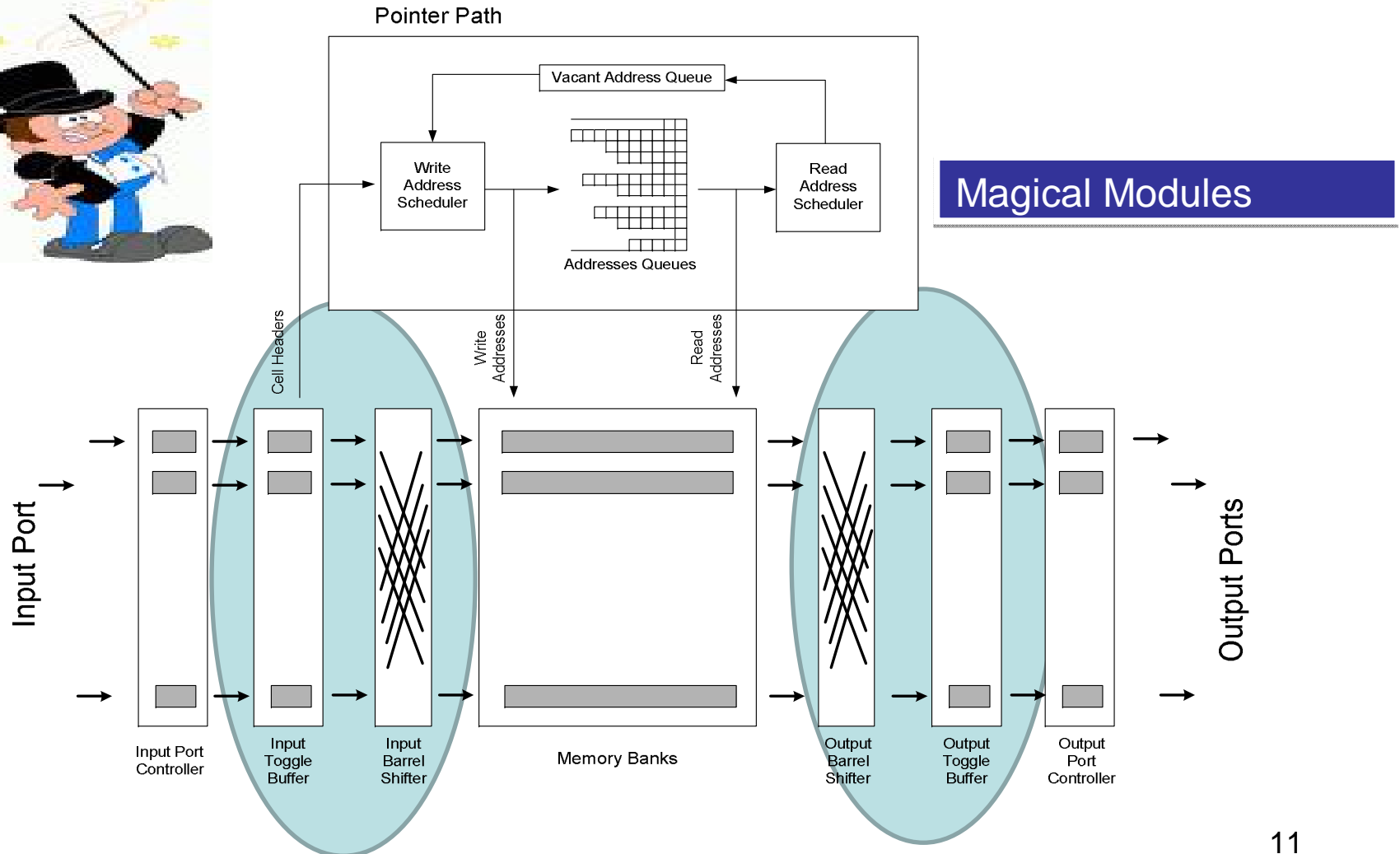


Higher Capacity Switch Fabric

$$\text{Switch Capacity} = \text{Memory Frequency} \times \text{Port Width} \times \text{No Ports}$$

- Example
 - Memory port width = 32 bit
 - Memory Access Frequency = 100 MHz
 - No of Ports = 4
 - Switch Capacity = 12.8 Gbps

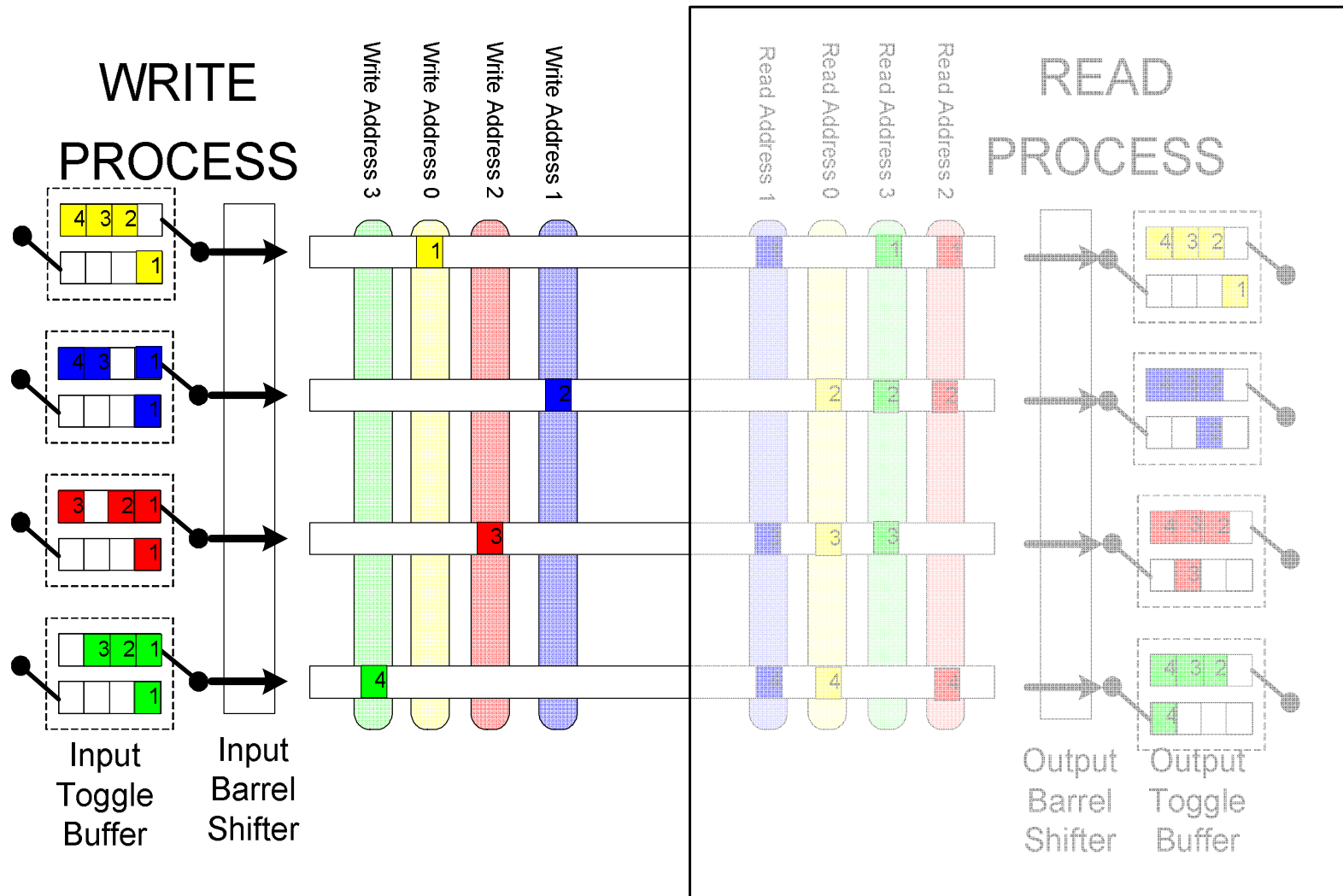
Distributed Shared Memory Block Diagram



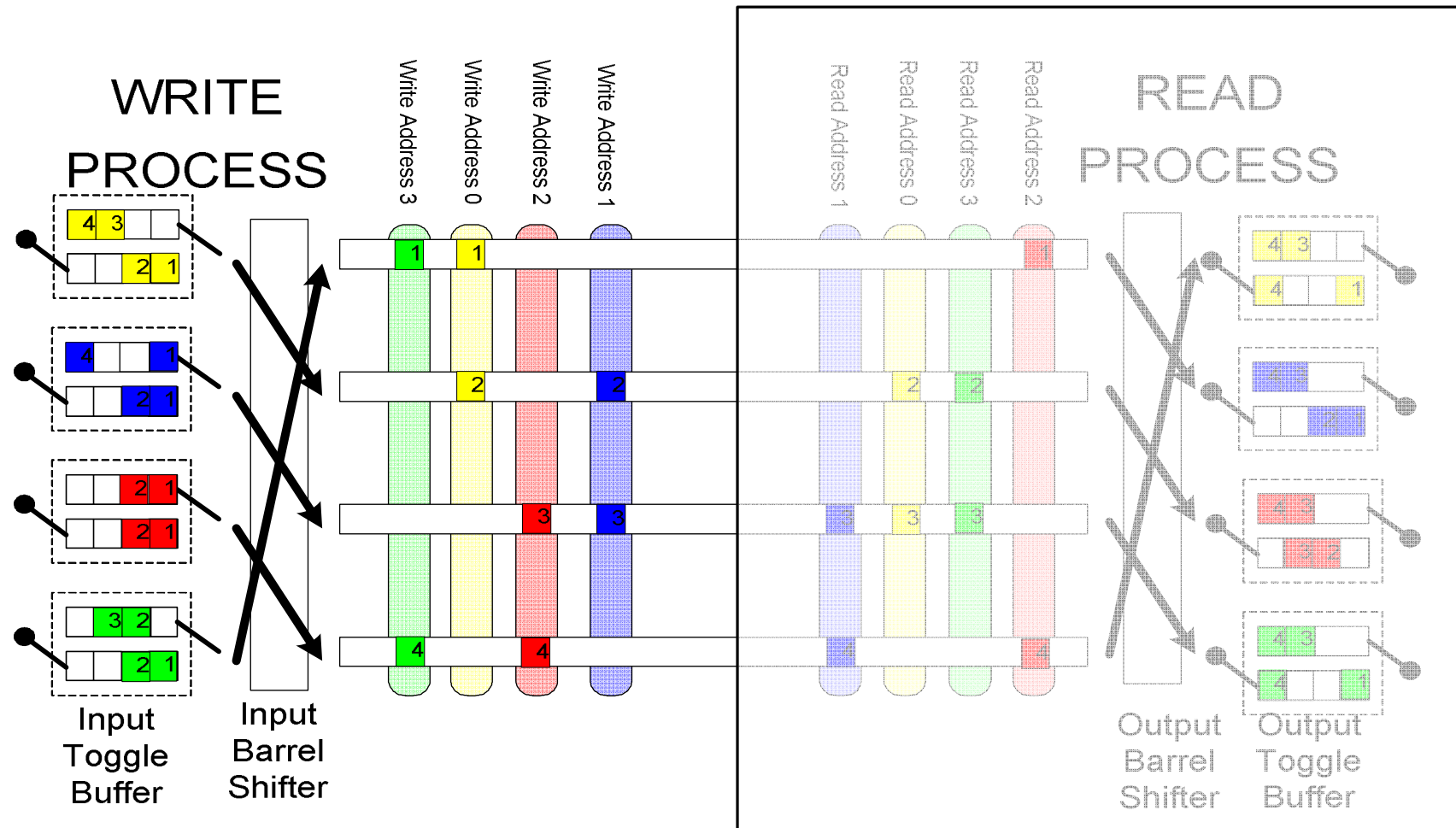
Clarifying Concept with an Example

- 4 x 4 switch
- Each cell consists of 4 words
- Cell Time Slot = 4 clock cycles

First Clock Cycle (WRITE PROCESS)

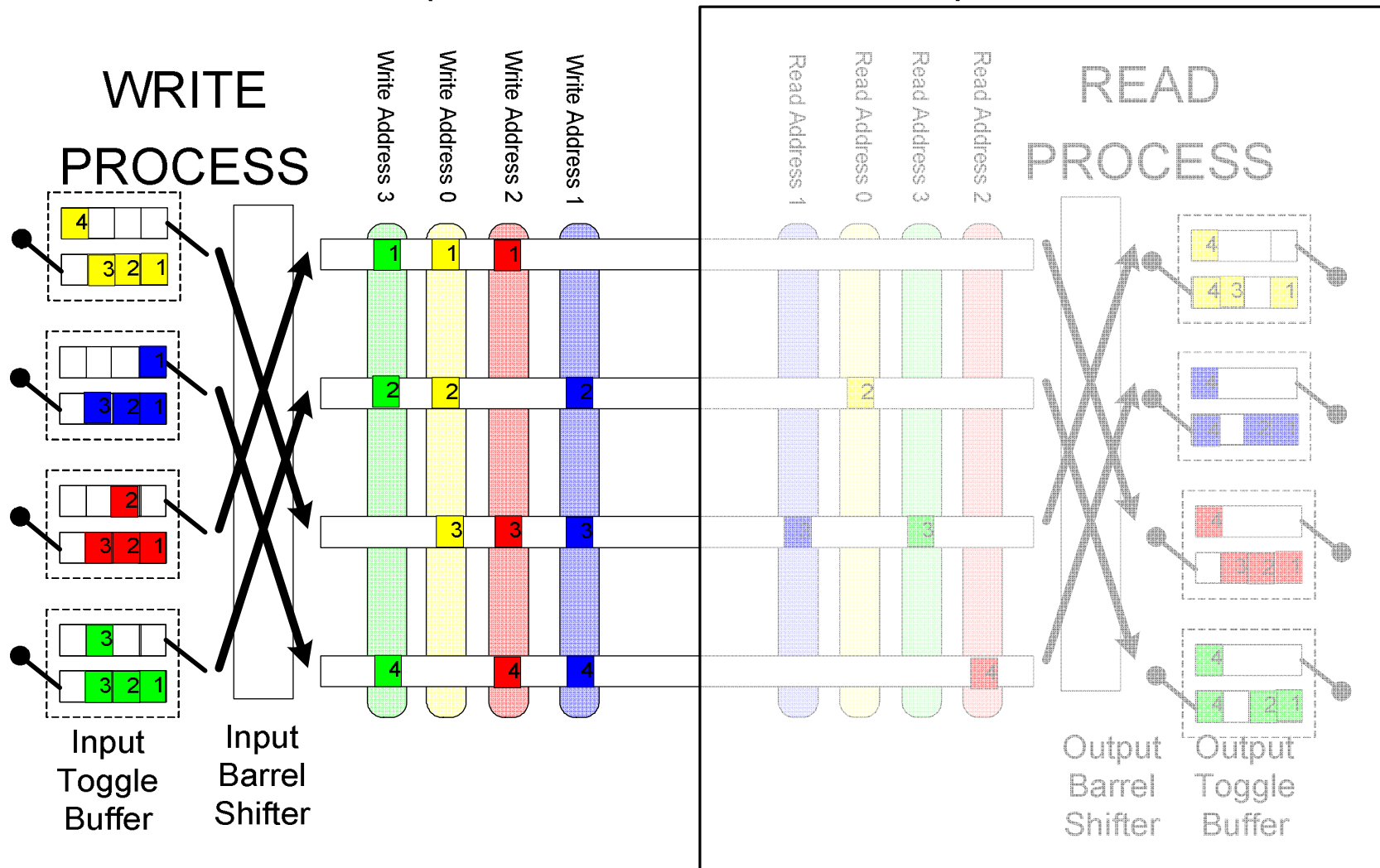


Second Clock Cycle (WRITE PROCESS)



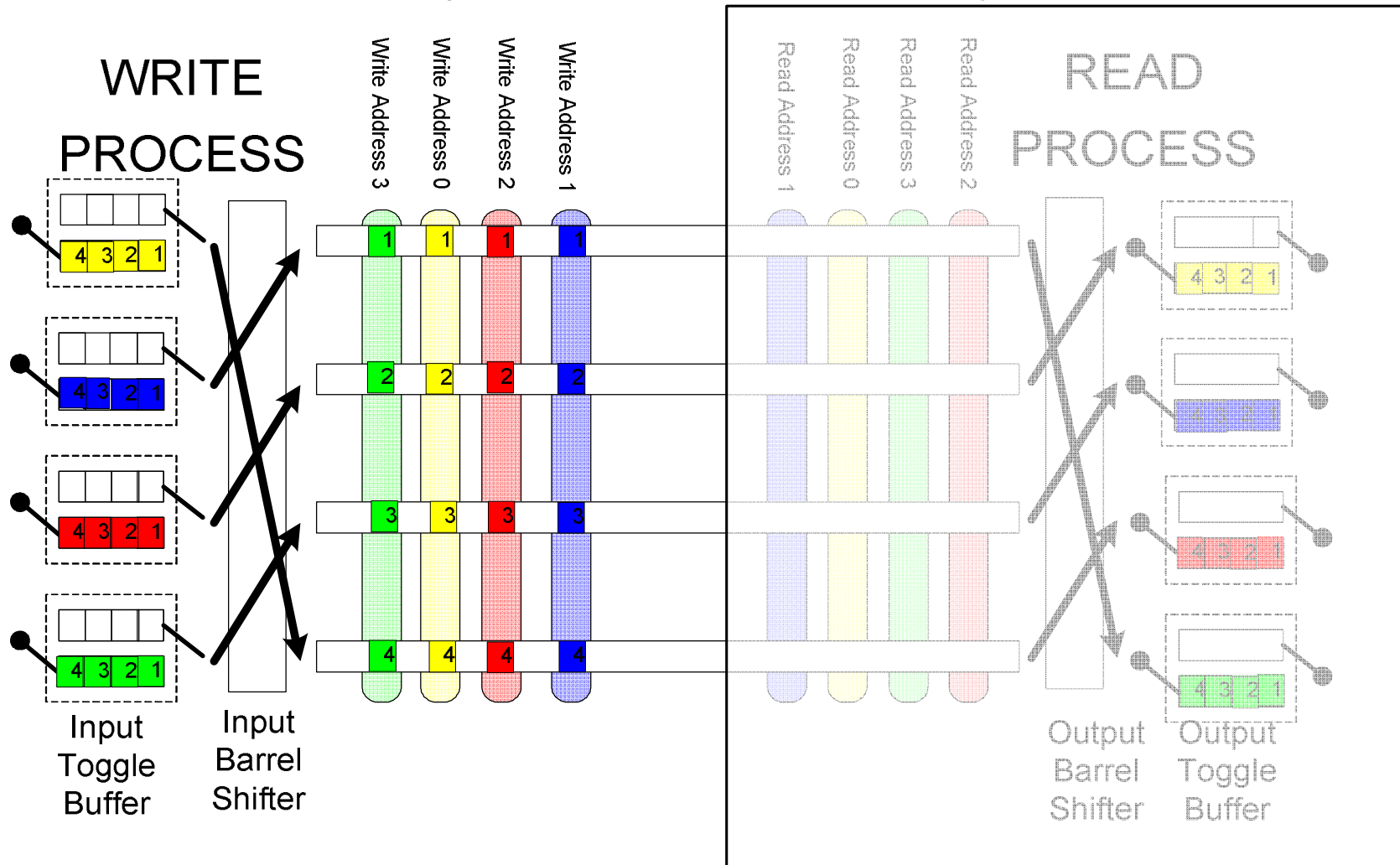
Third Clock Cycle

(WRITE PROCESS)

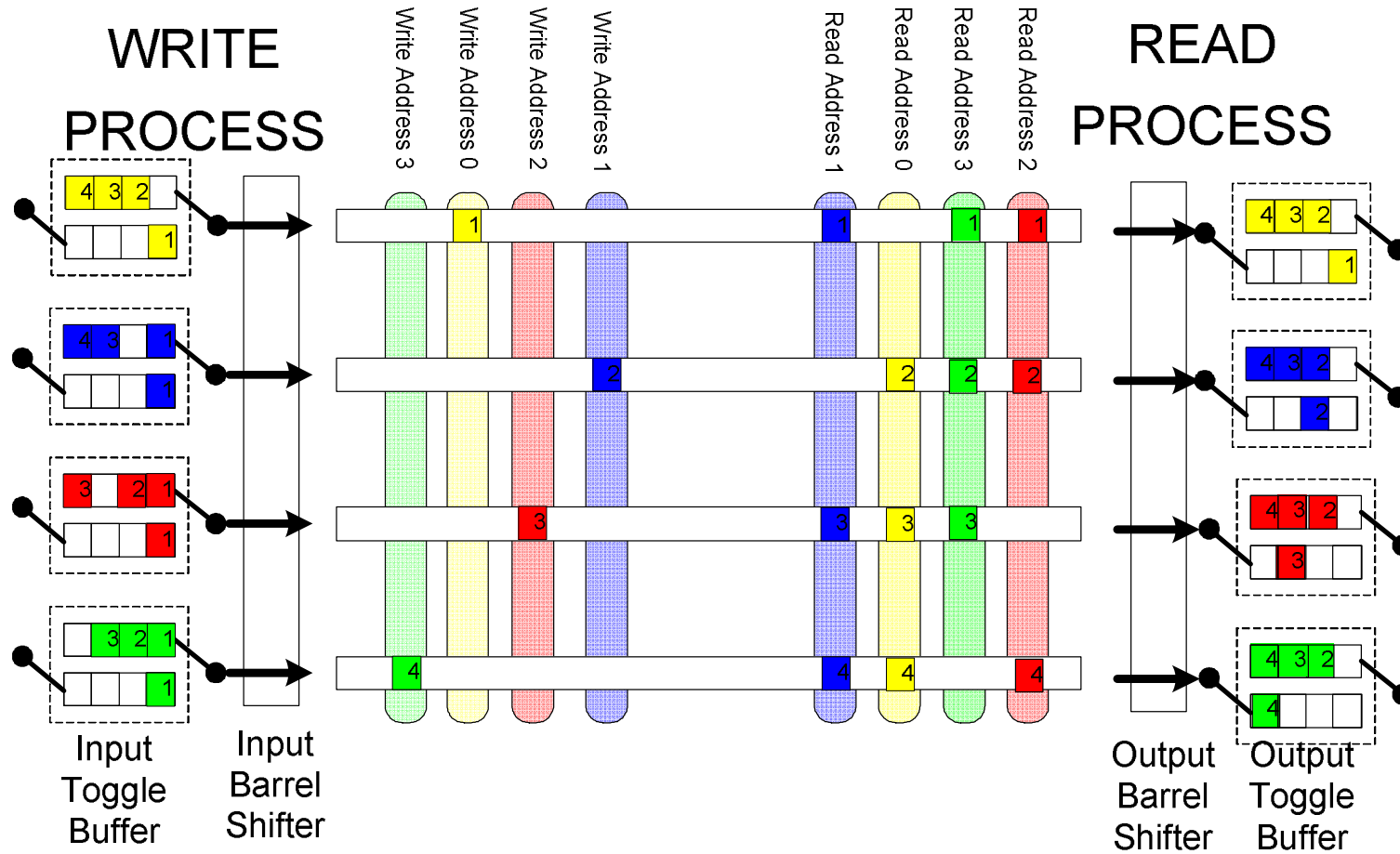


Fourth Clock Cycle

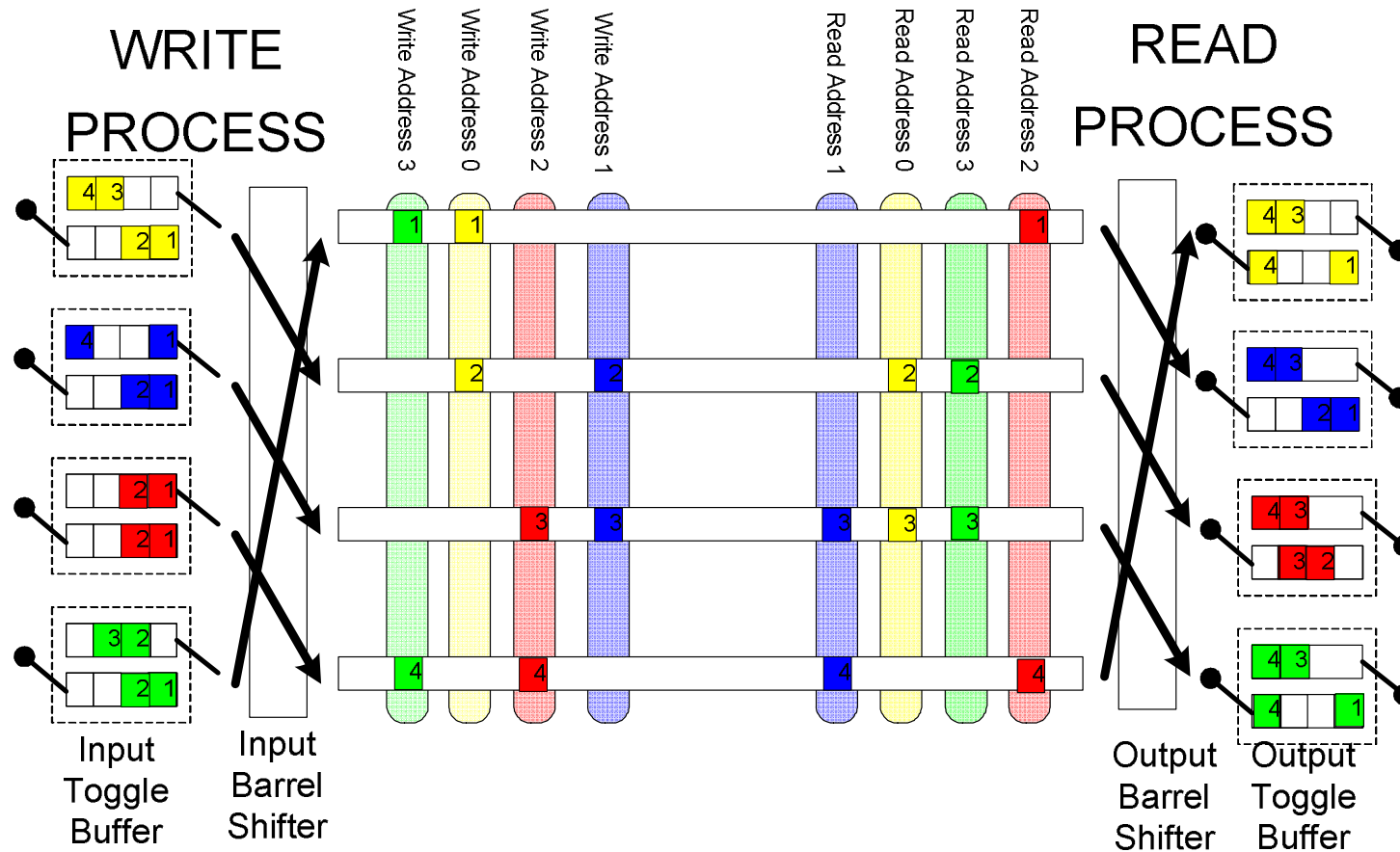
(WRITE PROCESS)



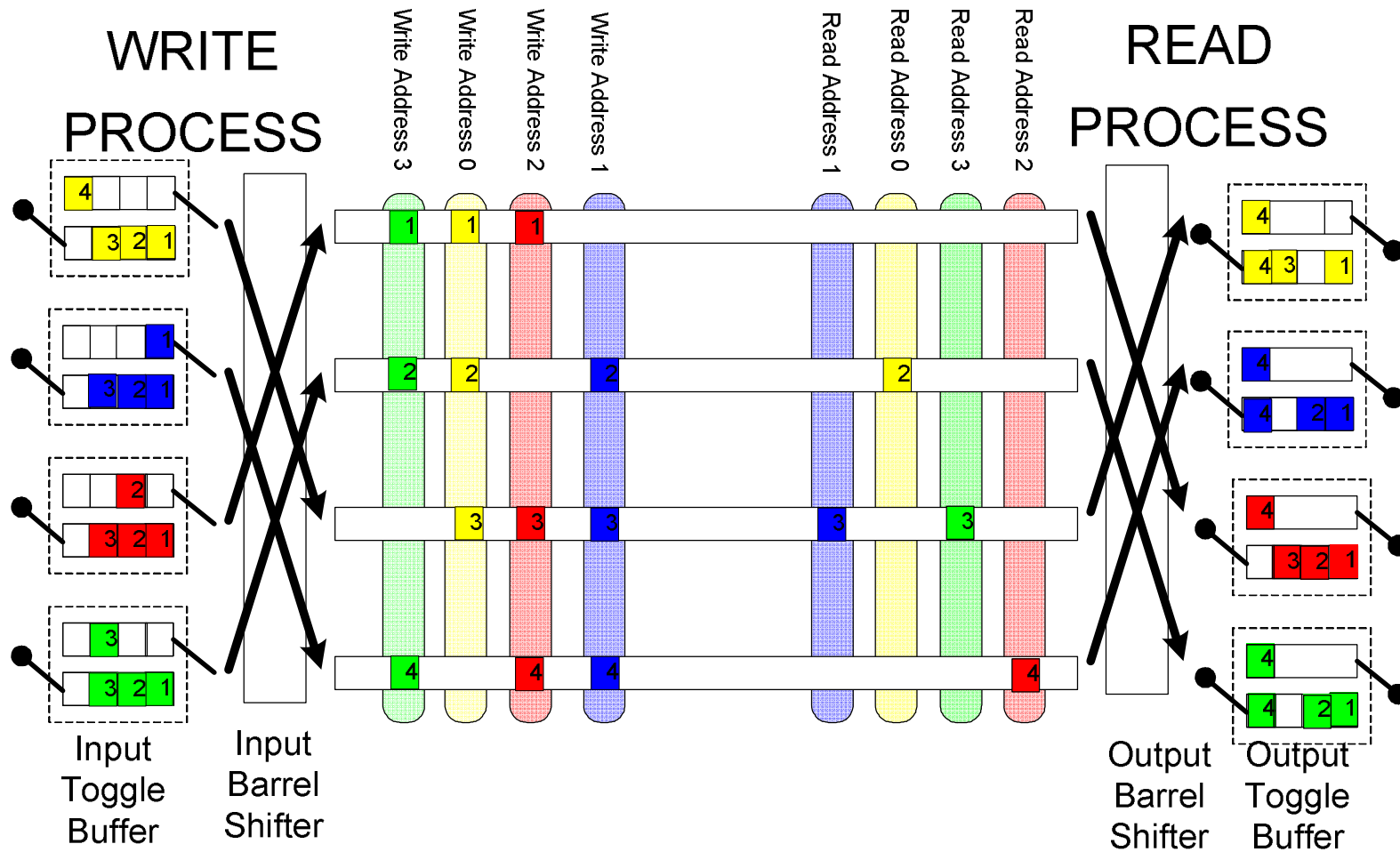
First Clock Cycle



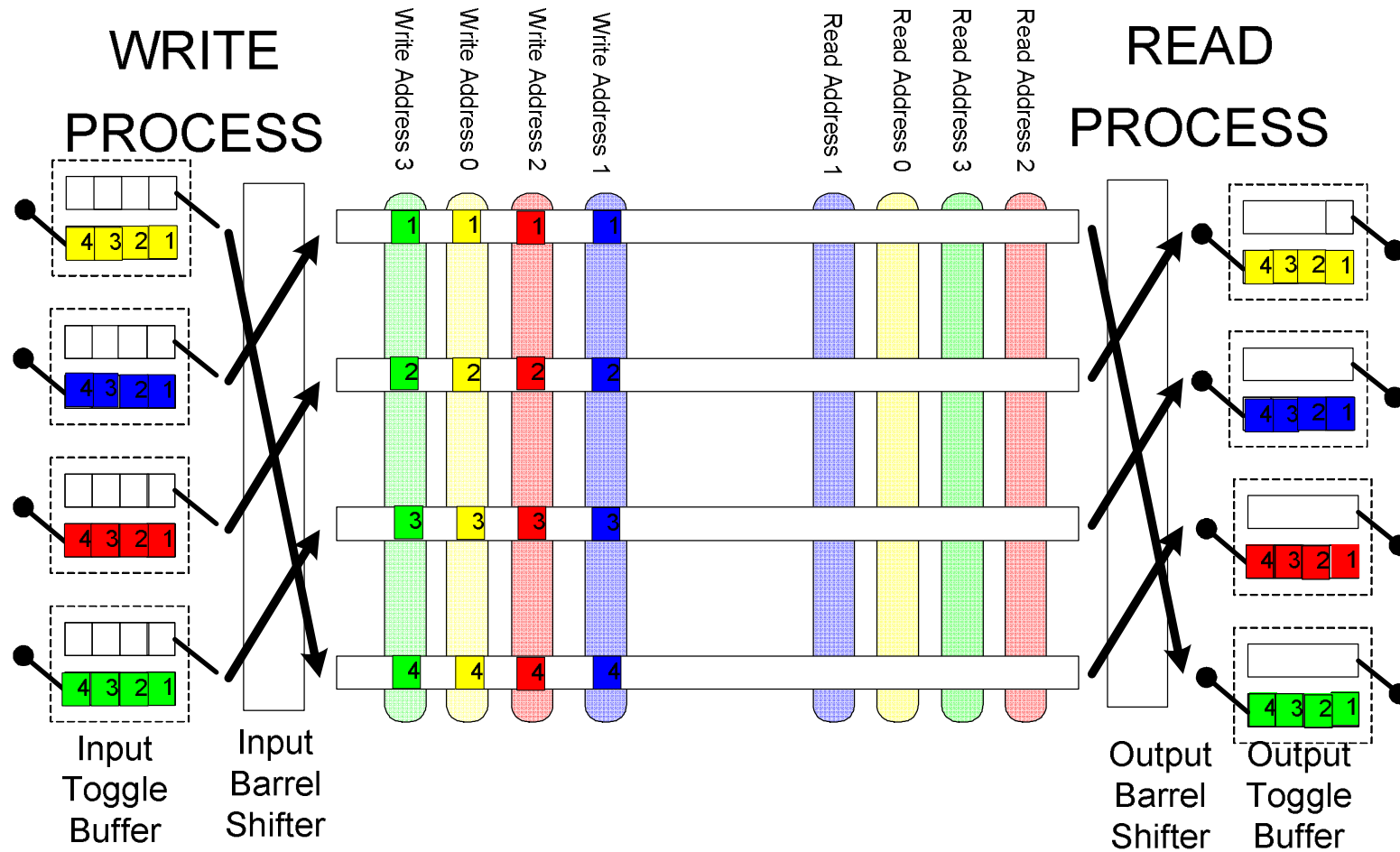
Second Clock Cycle



Third Clock Cycle



Fourth Clock Cycle



Implemented Switch Fabric Specification in 2005

- Virtex II 8000 from Xilinx
- 16 * 16 switch Fabric
- Working Clock = @ 70 M Hz
- Verilog HDL
- Cell Size = 81 Bytes (with 65 bytes payload size)
- Switching Capacity = 20 Giga Bit per second

Thanks

Question? / Answer



FPGA-based Correlator for Random Signal Processing in Noise Radar

**Konstantin Lukin, Olena Melnykova,
Sergey Lukin and Pavlo Vyplavin**



Outline

- Time domain correlator for continuous signals
- Frequency domain correlator
- Correlator processing results
- Conclusions

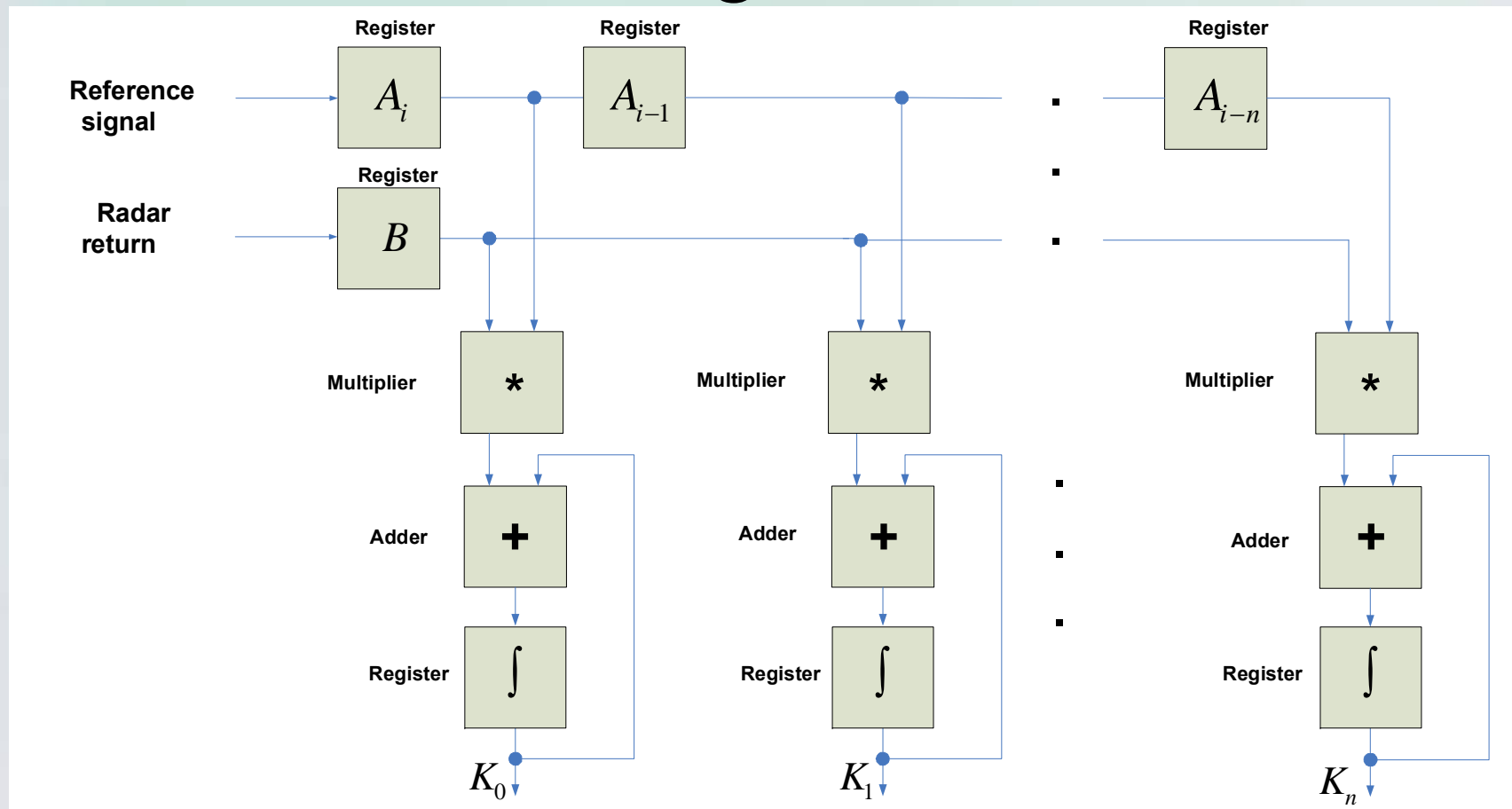


Estimation of cross-correlation using FPGA

$$K_{\tau} = \sum_{n=1}^N A_{n-\tau} B_n$$

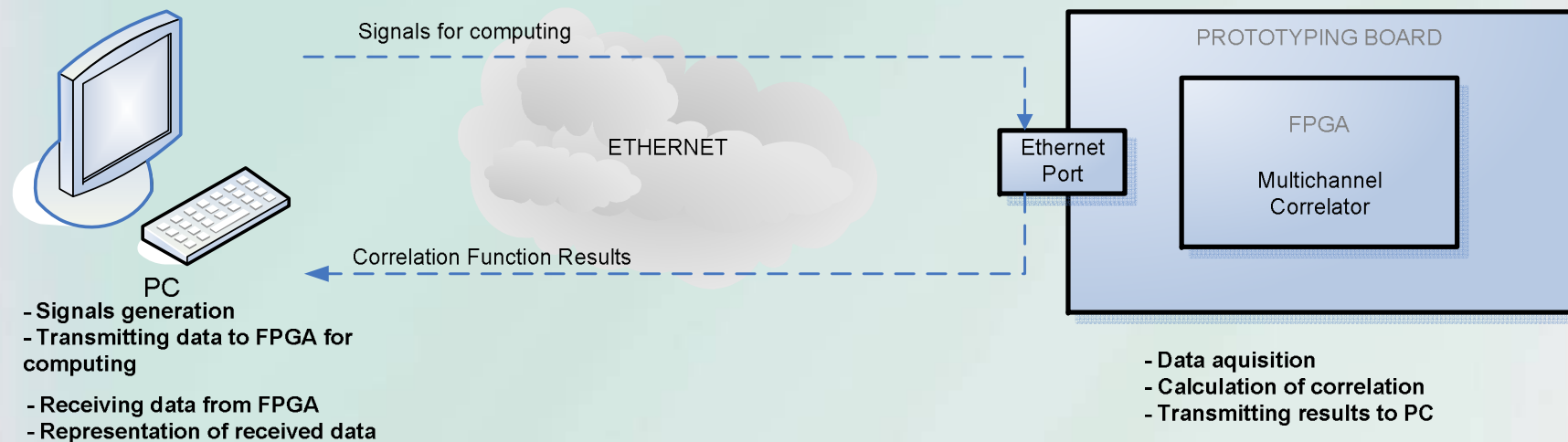


Diagram of correlator for CW signals



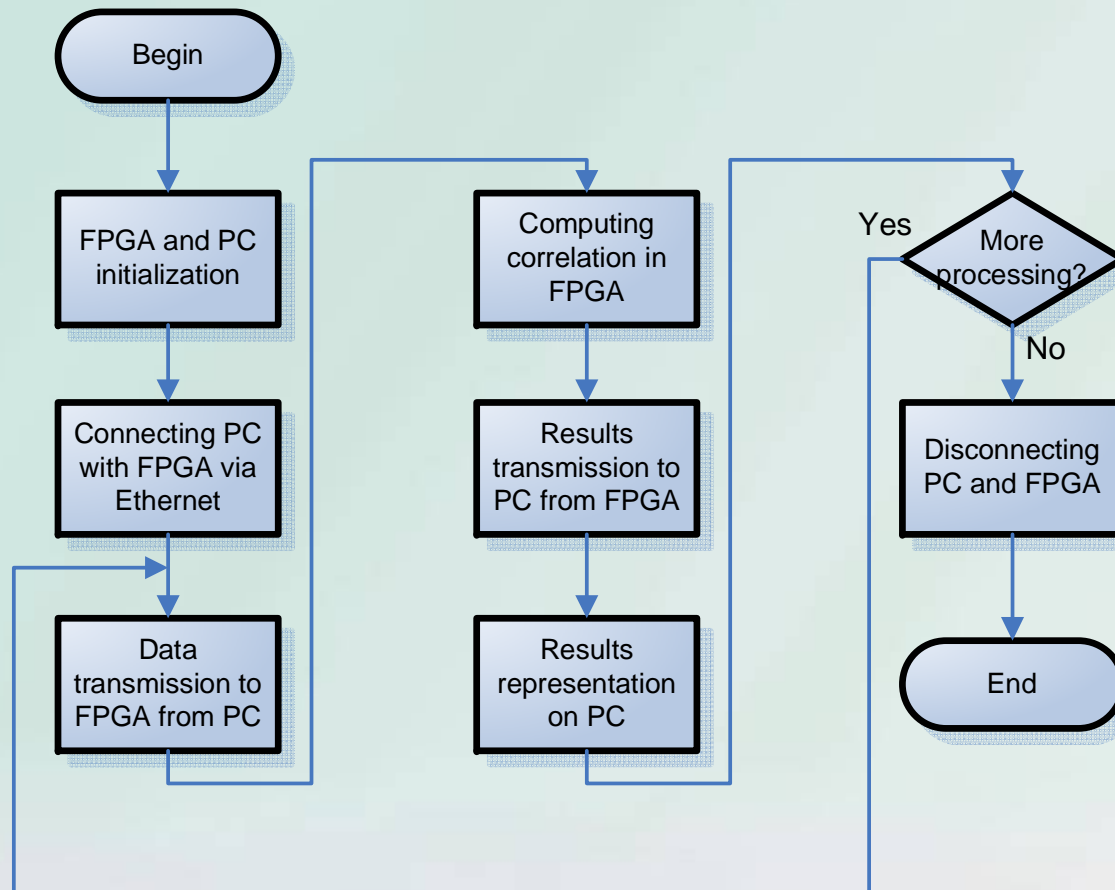


Flow diagram of cross-correlation computing model



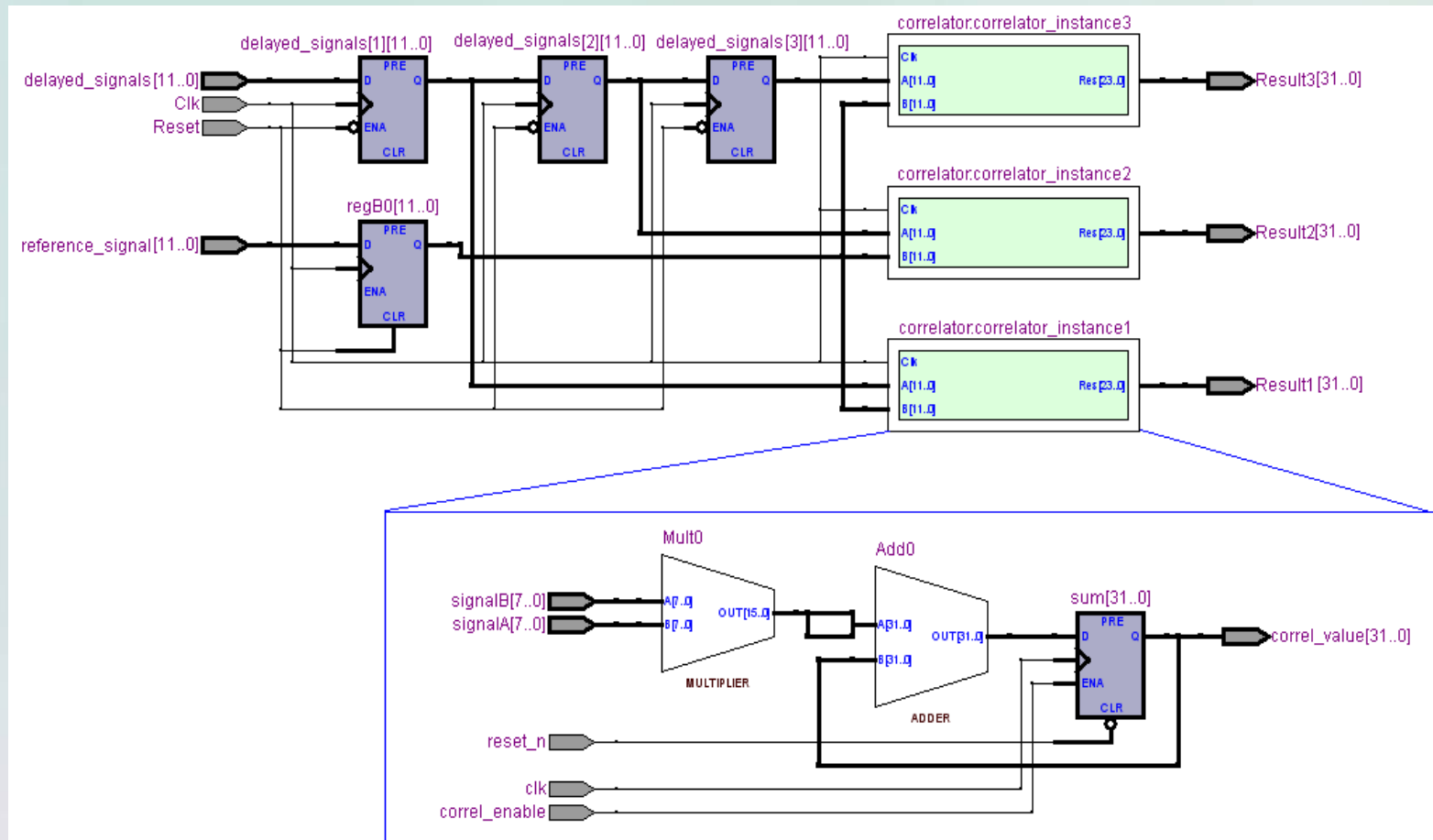


Cross-correlation computing algorithm



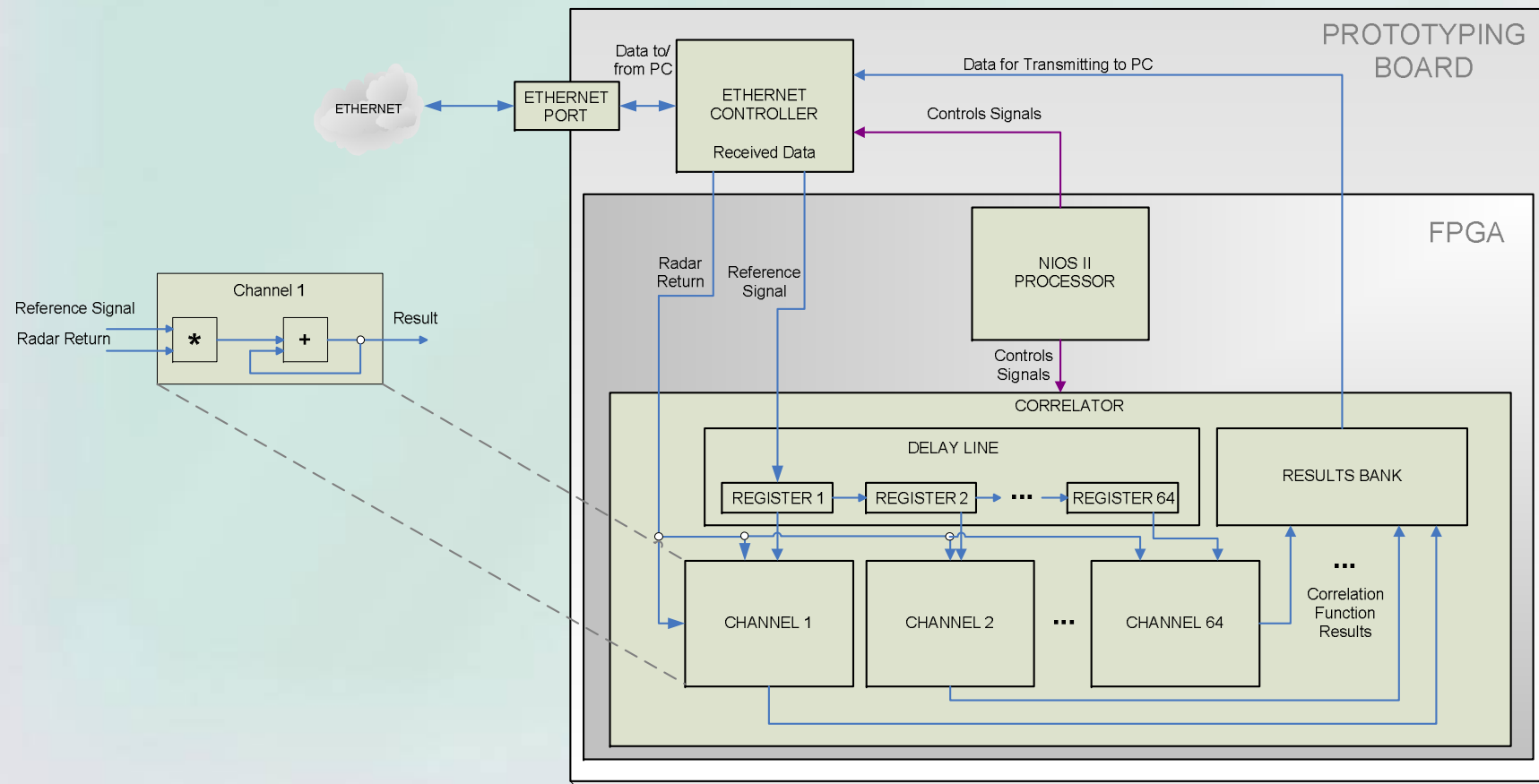


Flow diagram of multichannel correlator using FPGA





Flow diagram of multichannel correlator using FPGA

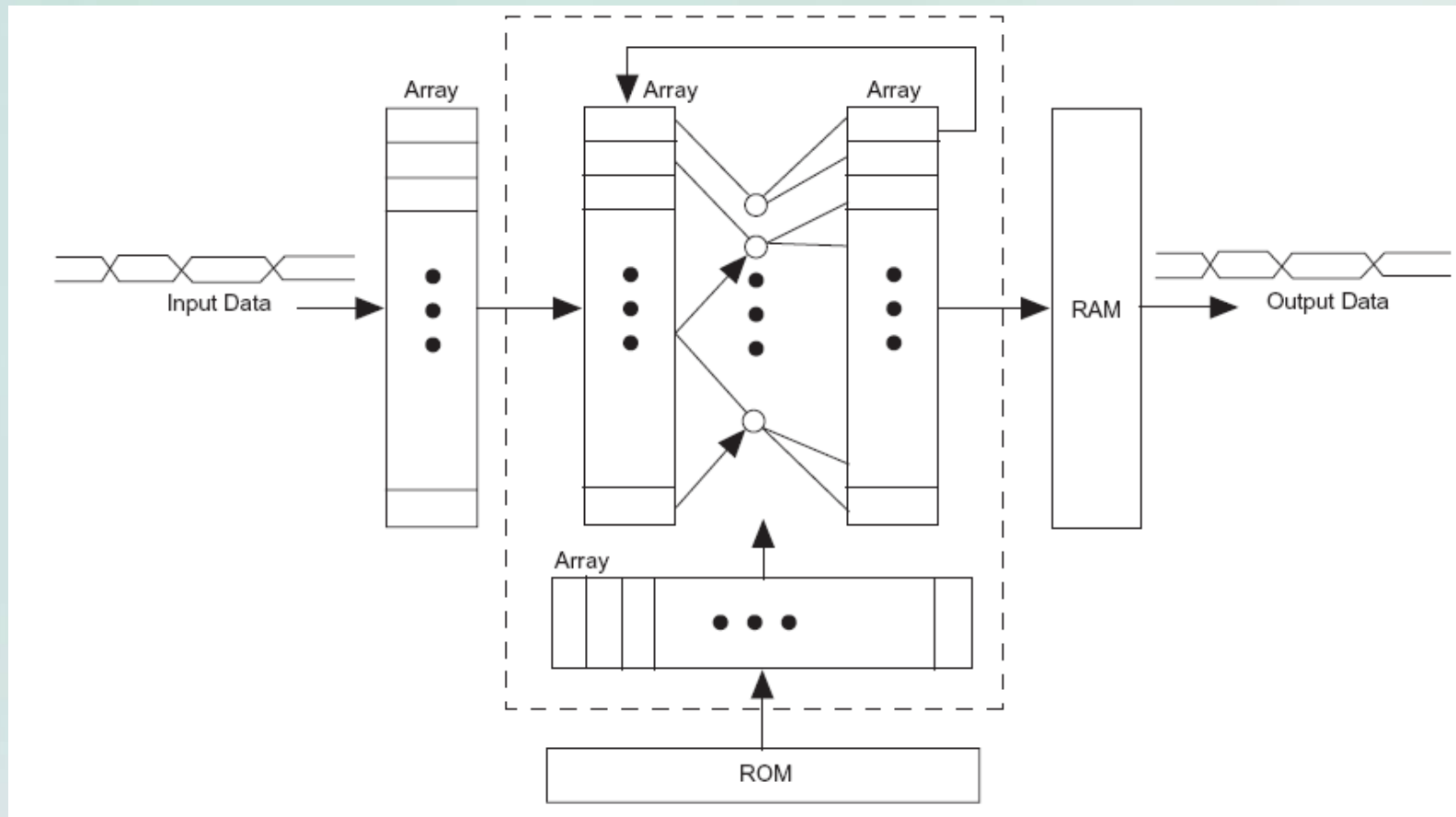


FFT of a length- N sequence

$$X[k] = \sum_{n=0}^{N-1} x(n)e^{(-j2\pi nk)/N}$$

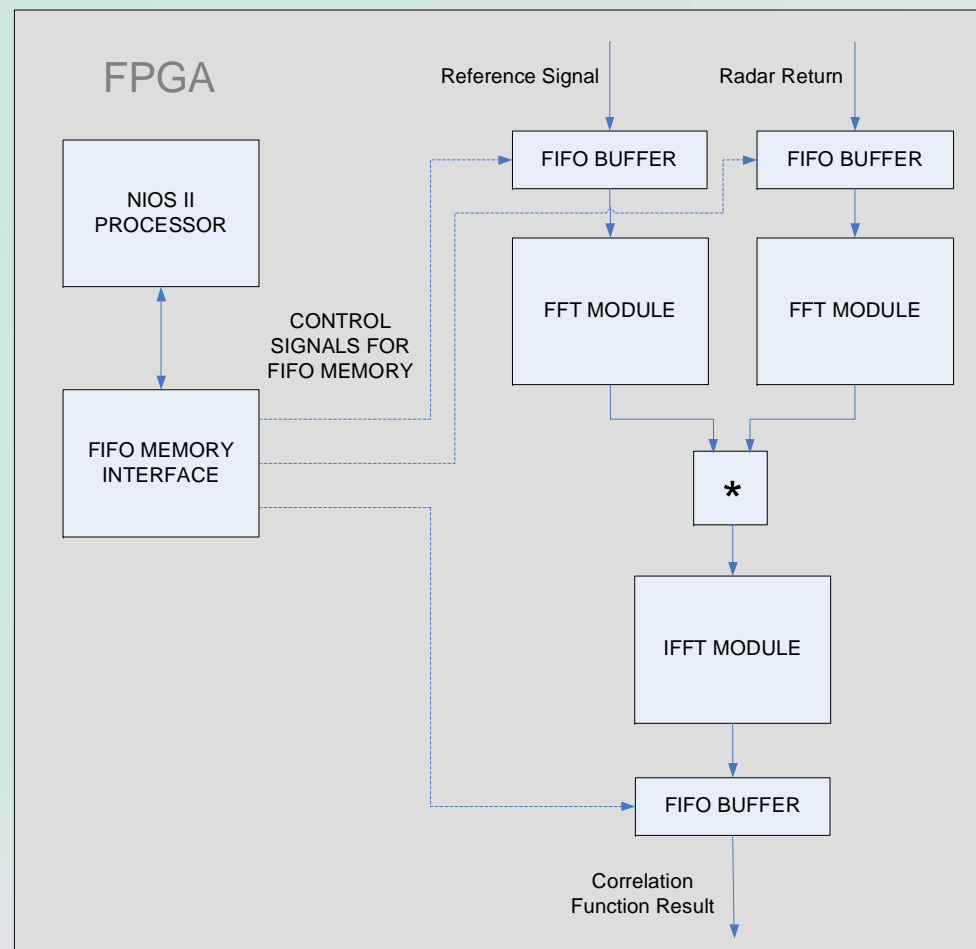
where $k = 0, 1, \dots, N-1$

Diagram of FFT Module



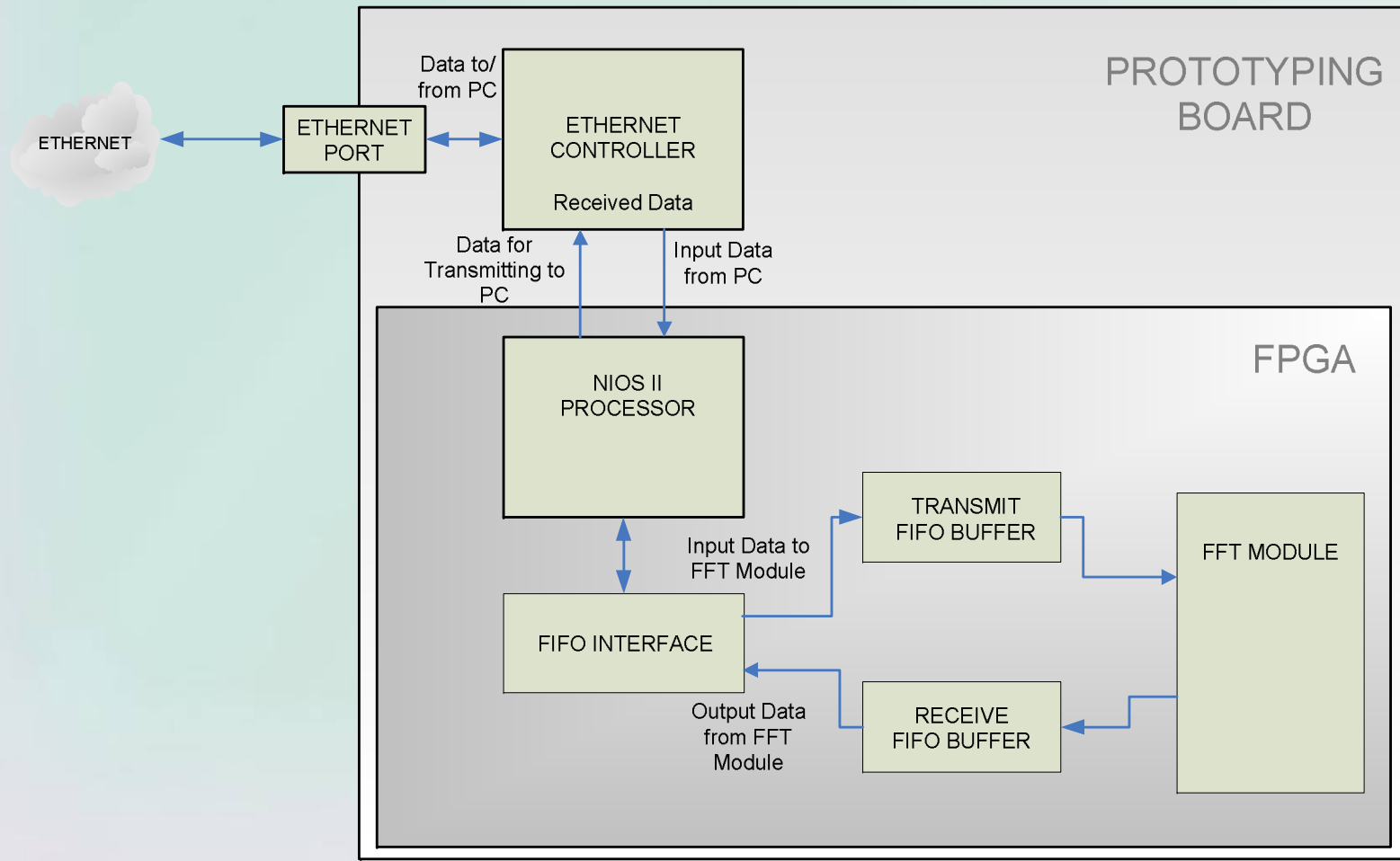


Flow diagram of FFT-based model using FPGA





Flow diagram of FFT-based correlator model using FPGA





User Interface

Correlator client

Controls

IP-address of FPGA
Server:
IP-address of PC
Client:
LNDES
Quantity of integration
Samples: Encode to base64
 Status: Status & log on

Log

Debugging information

Source 1

Transmission source: Text File

Sequence generator
Expression:
T min: T max: Samples:
Start time Finish time

Source 2

Transmission source: Text File

Sequence generator
Expression:
T min: T max: Samples:
Start time Finish time

Results

Diagram of computed in PC correlation function

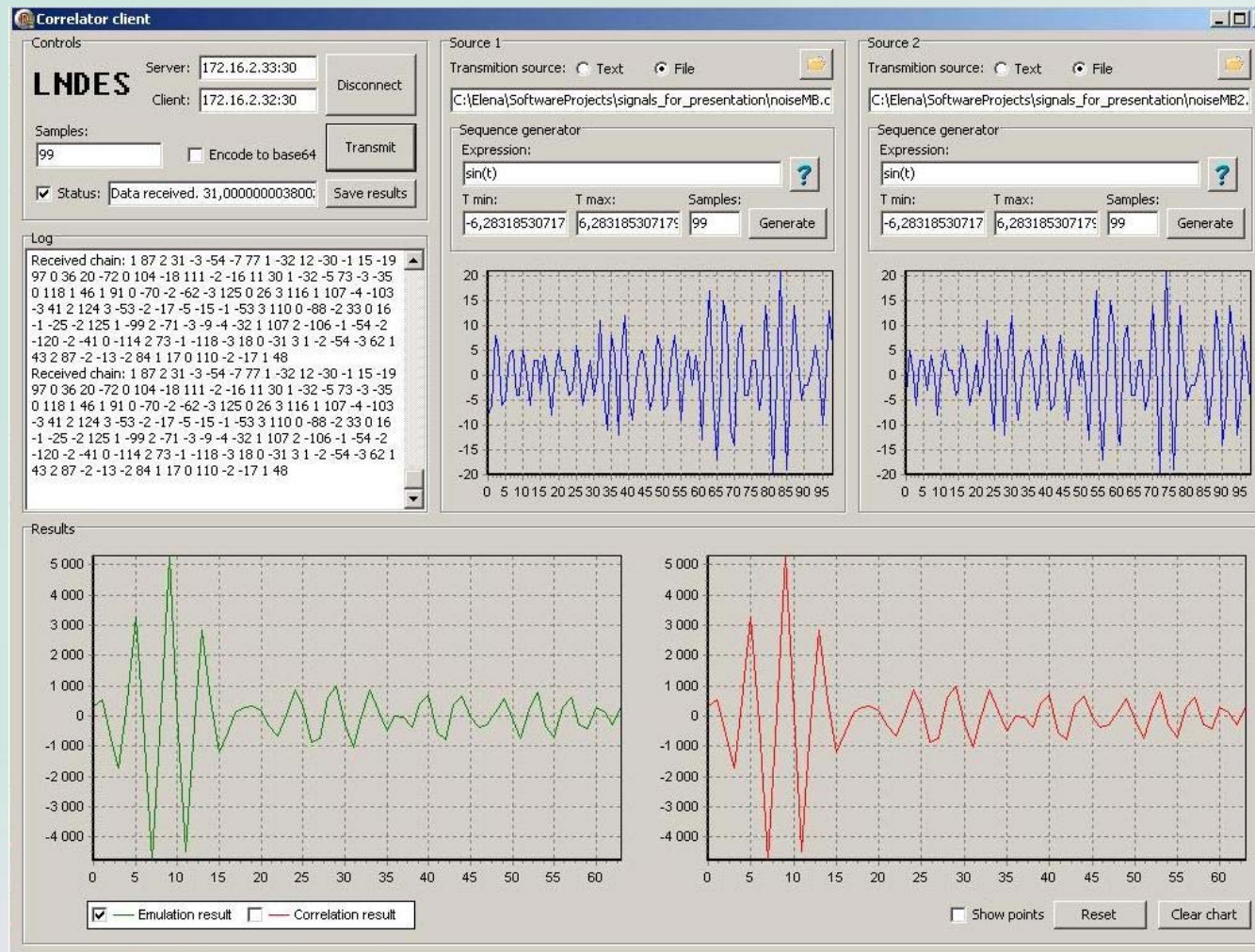
Diagram of computed in FPGA correlation function

Emulation result Correlation result

Show points



Results of processing of noise signals





Results of processing of noise signals

Correlator client

Controls
Server: 172.16.2.33:30
Client: 172.16.2.32:30
Samples: 99
Encode to base64
Status: Data received. 32,000,000,000,0209

Source 1
Transmission source: File
Sequence generator Expression: $\sin(t)$
T min: -6,28318530717 T max: 6,28318530717 Samples: 99

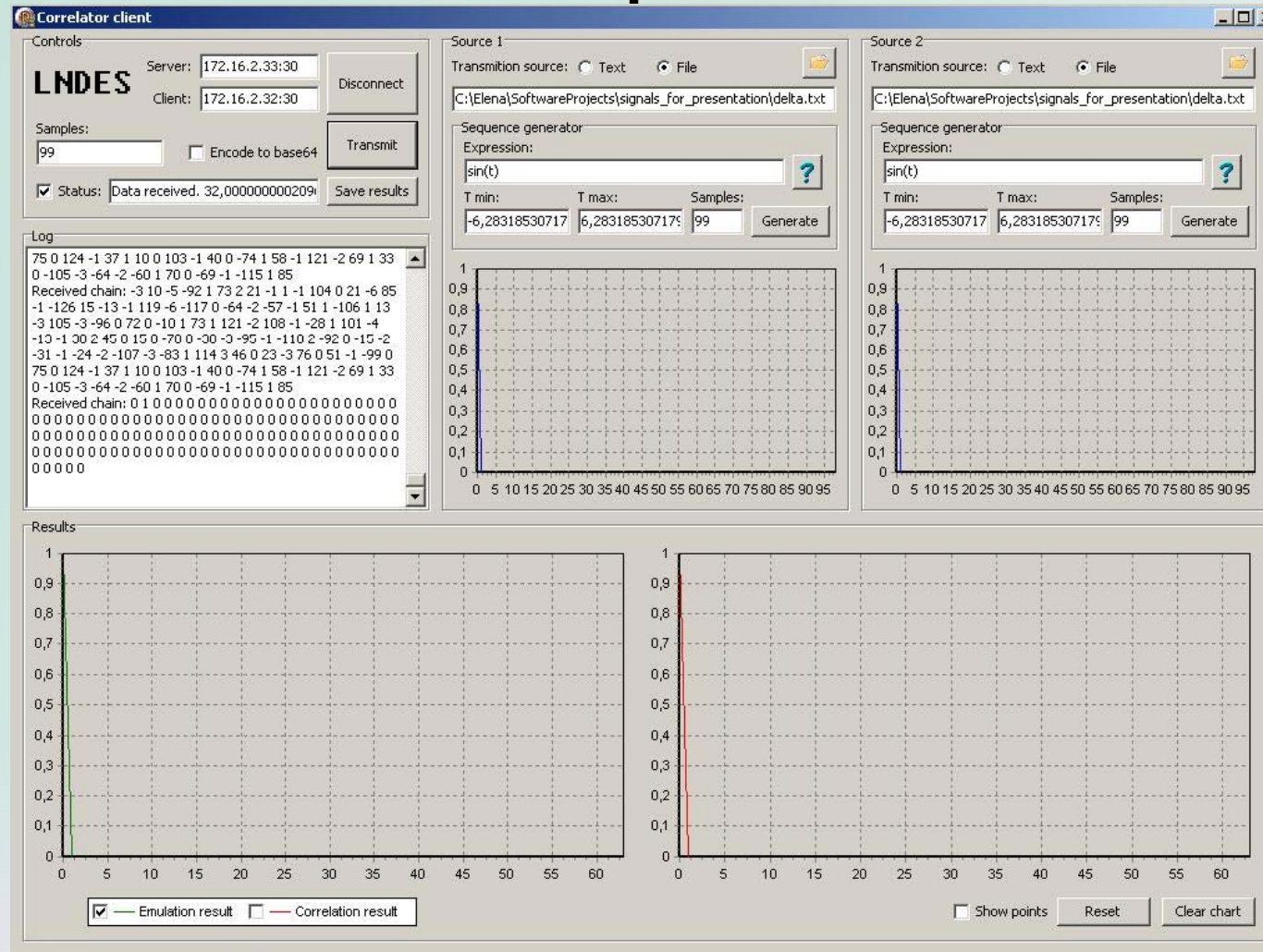
Source 2
Transmission source: File
Sequence generator Expression: $\sin(t)$
T min: -6,28318530717 T max: 6,28318530717 Samples: 99

Log
Received chain: 1 87 2 31 -3 -54 -7 77 1 -32 12 -30 -1 15 -19
97 0 36 20 -72 0 104 -18 111 -2 -16 11 30 1 -32 -5 73 -3 -35
0 118 1 46 1 91 0 -70 -2 -62 -3 125 0 26 3 116 1 107 -4 -103
-3 41 2 124 3 -53 -2 -17 -5 -15 -1 -53 3 110 0 -88 -2 33 0 16
-1 -25 -2 125 1 -99 2 -71 -3 -9 -4 -32 1 107 2 -106 -1 -54 -2
-120 -2 -41 0 -114 2 73 -1 -118 -3 18 0 -31 3 1 -2 -54 -3 62 1
43 2 87 -2 -13 -2 84 1 17 0 110 -2 -17 1 48
Received chain: -3 10 -5 -92 1 73 2 21 -1 1 -1 104 0 21 -6 85
-1 -126 15 -13 -1 119 -6 -117 0 -64 -2 -57 -1 51 1 -106 1 13
-3 105 -3 -96 0 72 0 -10 1 73 1 121 -2 108 -1 -28 1 101 -4
-13 -1 38 2 45 0 15 0 -70 0 -38 -3 -95 -1 -110 2 -92 0 -15 -2
-31 -1 -24 -2 -107 -3 -83 1 114 3 46 0 23 -3 76 0 51 -1 -99 0
75 0 124 -1 37 1 10 0 103 -1 40 0 -74 1 58 -1 121 -2 69 1 33
0 -105 -3 -64 -2 -60 1 70 0 -69 -1 -115 1 85

Results
Emulation result
Correlation result



Results of processing of delta-pulses





Results of processing of impulses

Correlator client

Controls
LNDES Server: 172.16.2.33:30 Client: 172.16.2.32:30
 Disconnect
 Samples: 99 Encode to base64 Transmit
 Status: Data received. 30,999999994207 Save results

Source 1
 Transmission source: Text File
 C:\Elena\SoftwareProjects\signals_for_presentation\impuls.txt
 Sequence generator
 Expression: sin(t) ?
 T min: -6,28318530717 T max: 6,28318530717 Samples: 99 Generate

Source 2
 Transmission source: Text File
 C:\Elena\SoftwareProjects\signals_for_presentation\impuls.txt
 Sequence generator
 Expression: sin(t) ?
 T min: -6,28318530717 T max: 6,28318530717 Samples: 99 Generate

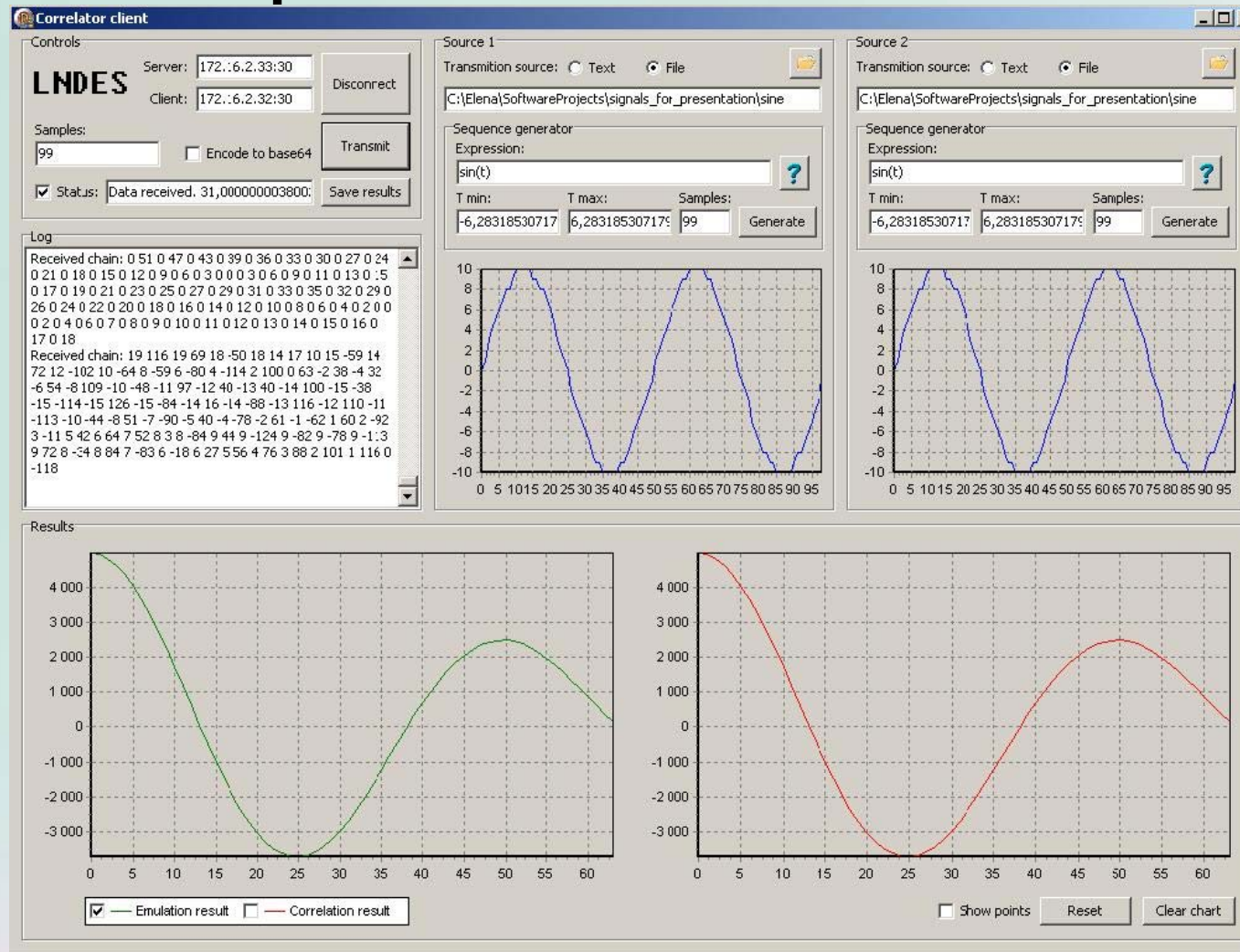
Log
 -1 -43 -1 -40 -1 -34 -1 -26 -1 -16 -1 -5 0 7 0 20 0 32 0 45 0
 57 0 68 0 78 0 87 0 93 0 98 0 102 0 105 0 105 0 104 0 101 0
 97 0 91
 Received chain: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 1 0 1 0 1 0
 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 1 0 1 0
 0 0 0 0
 Received chain: 0 51 0 47 0 43 0 39 0 36 0 33 0 30 0 27 0 24
 0 21 0 18 0 15 0 12 0 9 0 6 0 3 0 0 0 3 0 6 0 9 0 11 0 13 0 15
 0 17 0 19 0 21 0 23 0 25 0 27 0 29 0 31 0 33 0 35 0 32 0 29 0
 26 0 24 0 22 0 20 0 18 0 16 0 14 0 12 0 10 0 8 0 6 0 4 0 2 0 0
 0 2 0 4 0 6 0 7 0 8 0 9 0 10 0 11 0 12 0 13 0 14 0 15 0 16 0
 17 0 18

Results

Emulation result Correlation result
 Show points Reset Clear chart

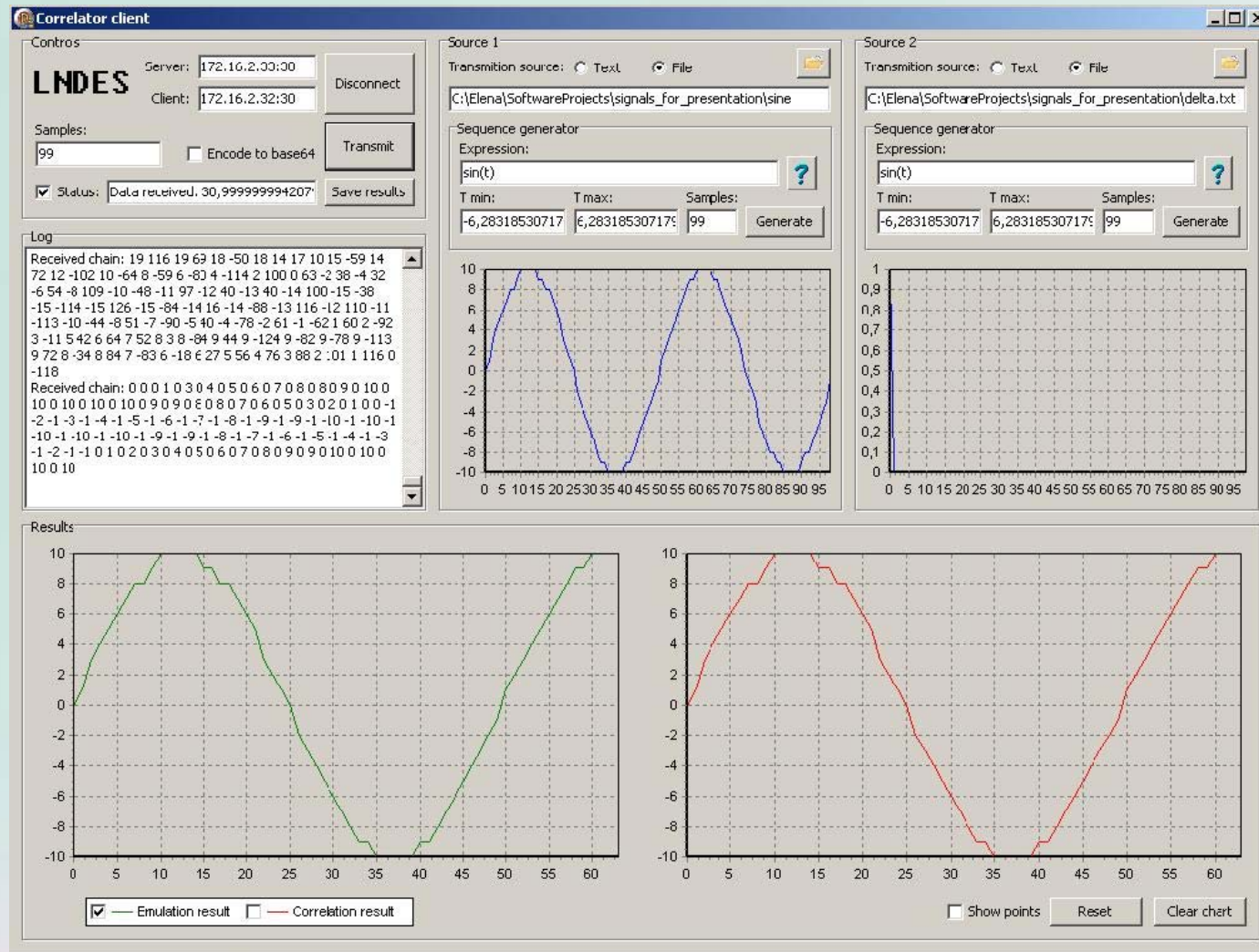


Results of processing of periodic functions





Results of processing of periodic signal and delta-pulse





Conclusions

1. We have implemented algorithms for correlation processing in FPGA in time domain and we are implementing algorithms in spectral domain.
2. The analysis shows that all correlators have rather good performance and each of them can be used in specific applications.
3. We realized correlator in the Altera/Stratix evaluation board.
4. In future we plan using these correlators in real radars.

Control system for solar tracking using a FPGA

L.A. Moreno-Coria¹, J.G. Pérez-Luna¹, B.S. Soto-Cruz¹, J.L. Sosa-Sanchez¹

¹ Center for Research in Semiconductor Devices, Benemérita Universidad Autónoma de Puebla, México.

*Advanced Training Course on FPGA Design and VHDL for Hardware
Simulation and Synthesis*

*26 October 2009-20 November 2009
Trieste-Italy*



Contents

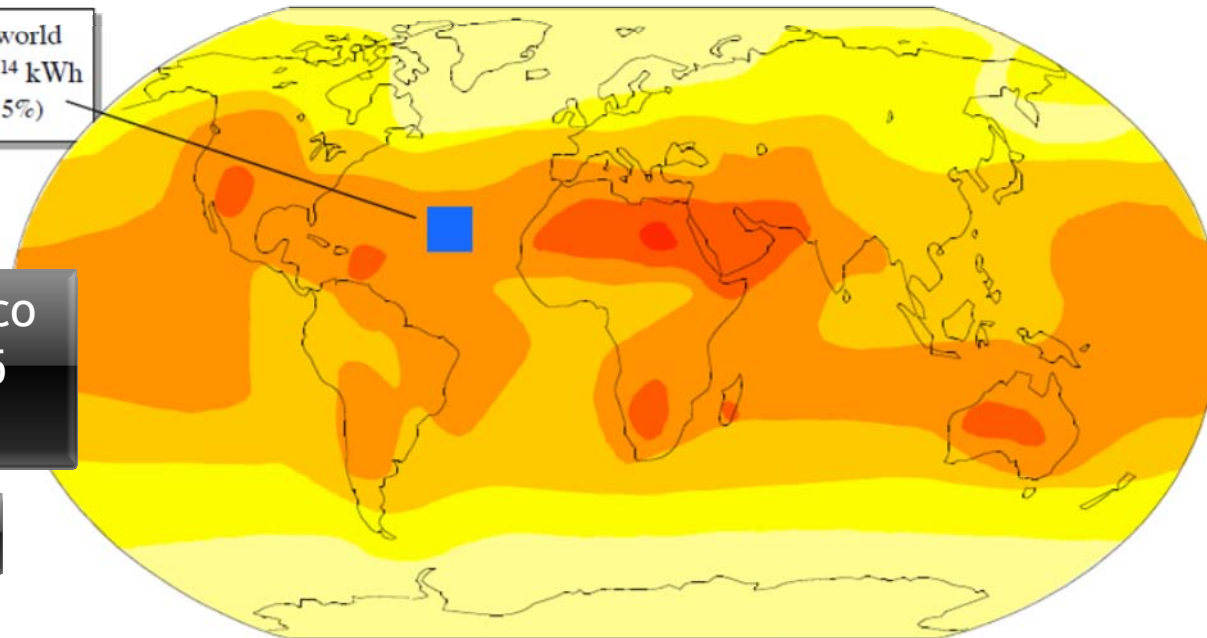
- *Introduction*
- *Method*
- *Discussion*

Introduction

Area required to cover world energy demand of 2×10^{14} kWh
($H = 2000 \text{ kWh m}^{-2} \text{ a}^{-1}$, $\eta = 5\%$)

Comparison of México : 1.86 respect to 2.55 of Sahara region

1750 kWh/m²



Solar irradiation

[kWh m⁻² a⁻¹]

< 876 876 1314 1752 2190 2628

[Principles of Solar Engineering, F. Krieth; J. Krieger, Mc Grow Hill, 1978].

Figure 1. Insolation for cover global demand for energy.

Introduction

- Are currently developing mechanisms for harnessing solar energy such as:

Solar Electricity

Solar Fuels

Solar thermal systems (thermoelectric cells, thermionic cells
but this requires high temperatures [1])

In this work, we are working on the implementation of a solar tracking control of polar and high precision from a set of geometric equations whose define the apparent position of the sun.

Using FPGA, we create conditions for the search of optimization of power generation, as is the case of thermionic-thermoelectric generators, where cogeneration is done with a better conversion efficiency.

[1] Pérez Luna J. G., *Desarrollo de un Generador Termoiónico de Corriente Alterna*, Tesis Doctoral, U.N.A.M., México 2001.



Introduction

- To locate the solar position, we use the geometric model that determines the position centered on the position of the observer (Fig 2):

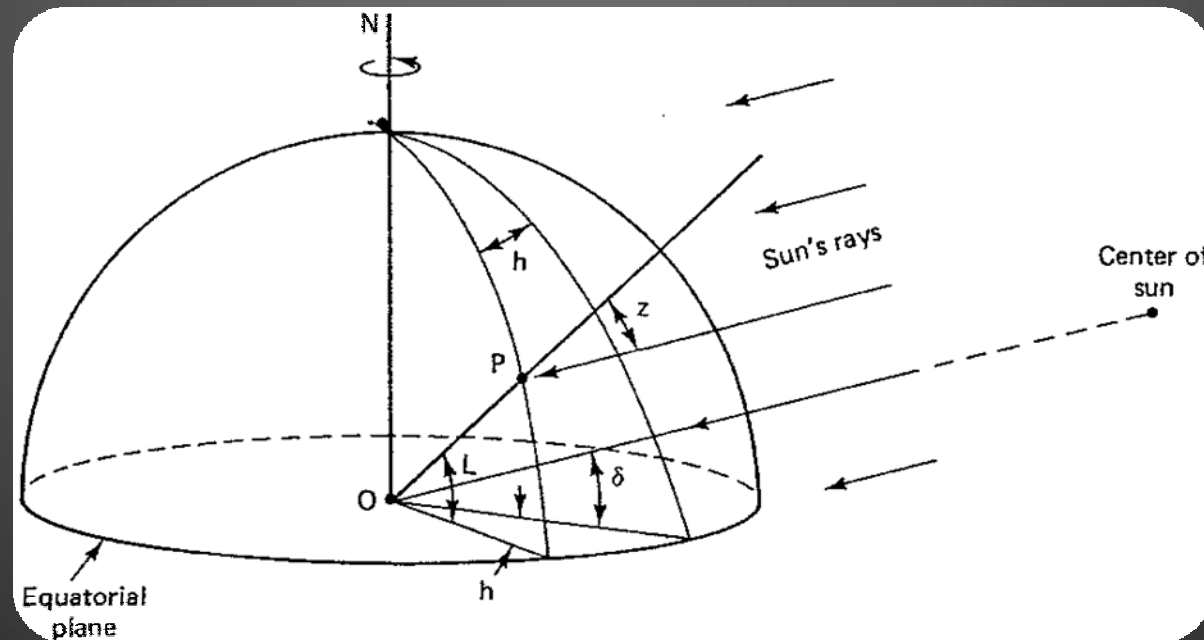


Figure 2. Definition of latitude L , hour angle h , a point P on the Earth's surface and solar declination δ .

Introduction

- To locate the solar position we use the geometric model, that determines its position based on the position of the observer (Fig 3):

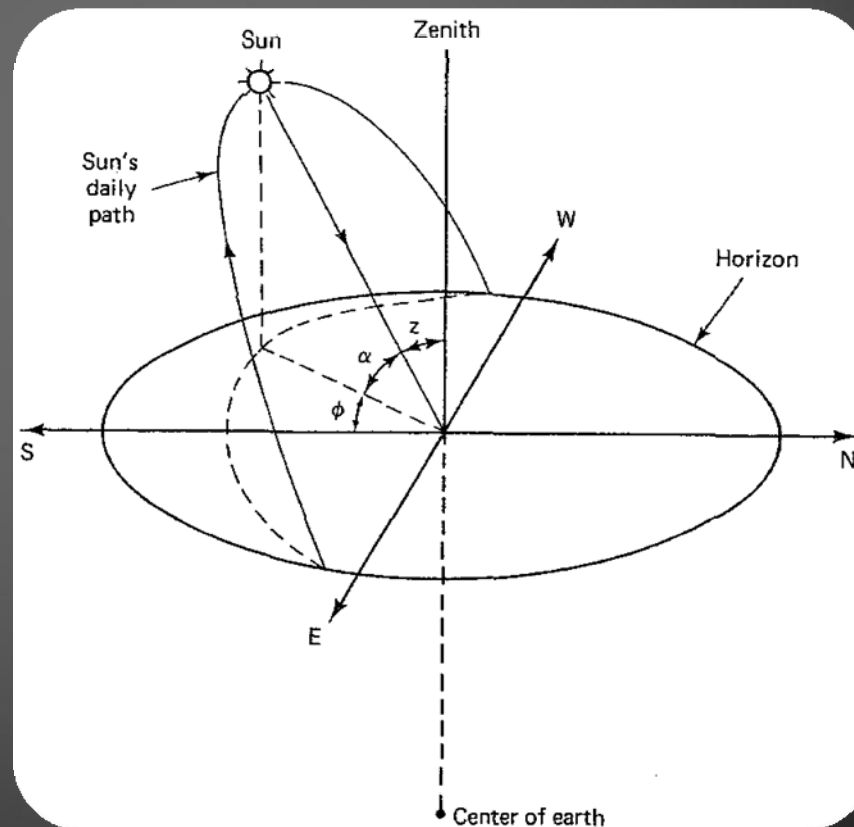


Figure 3. Angles of the solar position.

Introduction

From the Figures 1 and 2, we can obtain the angles involved in solar geometry. Their relationships through trigonometric inferences, are:

$$\delta = 23.45 \sin \left[\frac{360}{365} (284 + n) \right] \quad 1$$

$$\text{Solar Time} = \text{Official Time} + E + 4(L_{ref} - L_{loc}) \quad 2$$

$$E = 9.87 \sin(2B) - 7.53 \cos B - 1.5 \sin B \quad 3$$

$$B = \frac{360}{364} (n - 81) \quad 4$$

$$n = \text{Day of the year} (1 \leq n \leq 365) \quad 5$$

$$\omega = -(15^\circ/\text{hr})(\text{solar time}) + 180^\circ \quad 6$$

$$\sin \alpha = \cos \varphi \cos \delta + \sin \varphi \sin \delta \quad 7$$

$$\sin \gamma = \frac{\cos \delta \sin \omega}{\cos \alpha} \quad 8$$

E =equation of time

L_{ref} =length of the official
time reference
meridian for the zone
in question

L_{loc} =length of the
meridian of the place

Contents

- *Introduction*
- *Method*
- *Discussion*

Method

- Once established the restriction of movement to two degrees of freedom:

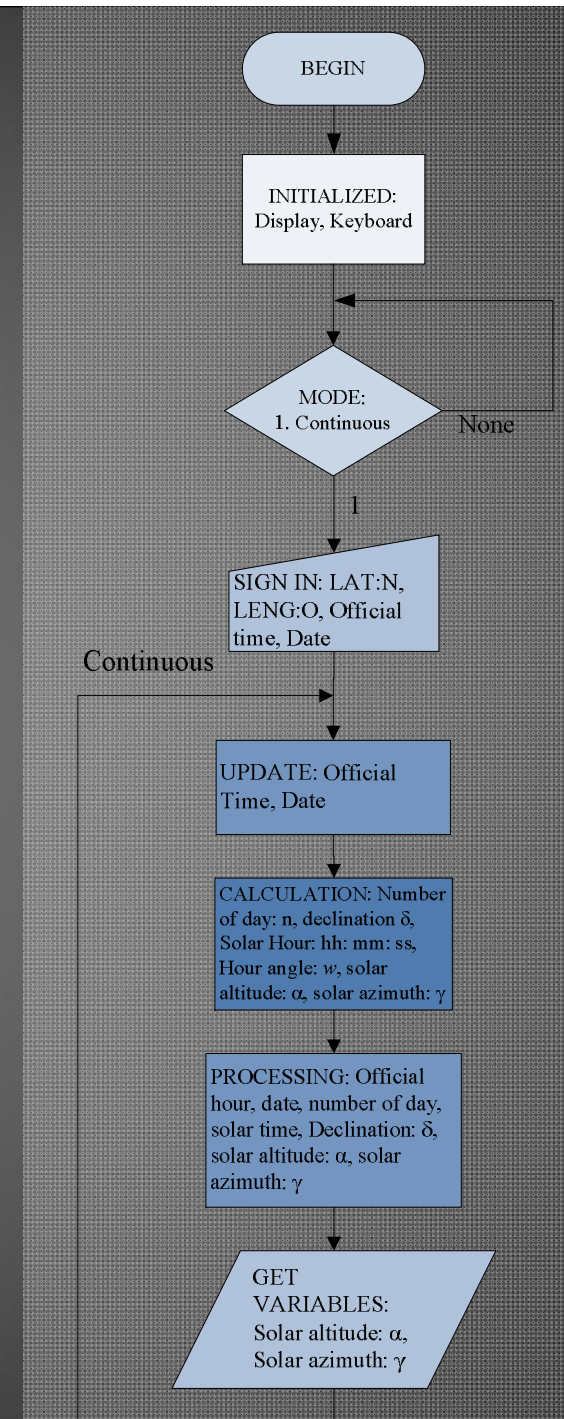
- *Solar altitude (a)*
- *Solar azimuth (γ)*

Method

Diagram flow

➤ Is part of an algorithm that calculates the solar position where data is entered and then you get the angles that define a particular position, finally, updates the time-data and obtains the solar position for each instant

Figure 5. Diagram flow.



Method

PID Control in closed loop

$$m(k) = k_p e(kT) + K_i \sum_{h=1}^k \frac{e((k-1)T) + e(kT)}{2} + k_d (e(kT) - e((k-1)T)) \quad 9$$

9

m(t) = output response in the control stage

e(t) = actuating error signal

k_p = proportional control action

K_i = integral control action

k_d = derivative control action

- The PID acts on the error of the variable, which will be manipulated through an appropriate combination of the three control measures

Method

Power stage

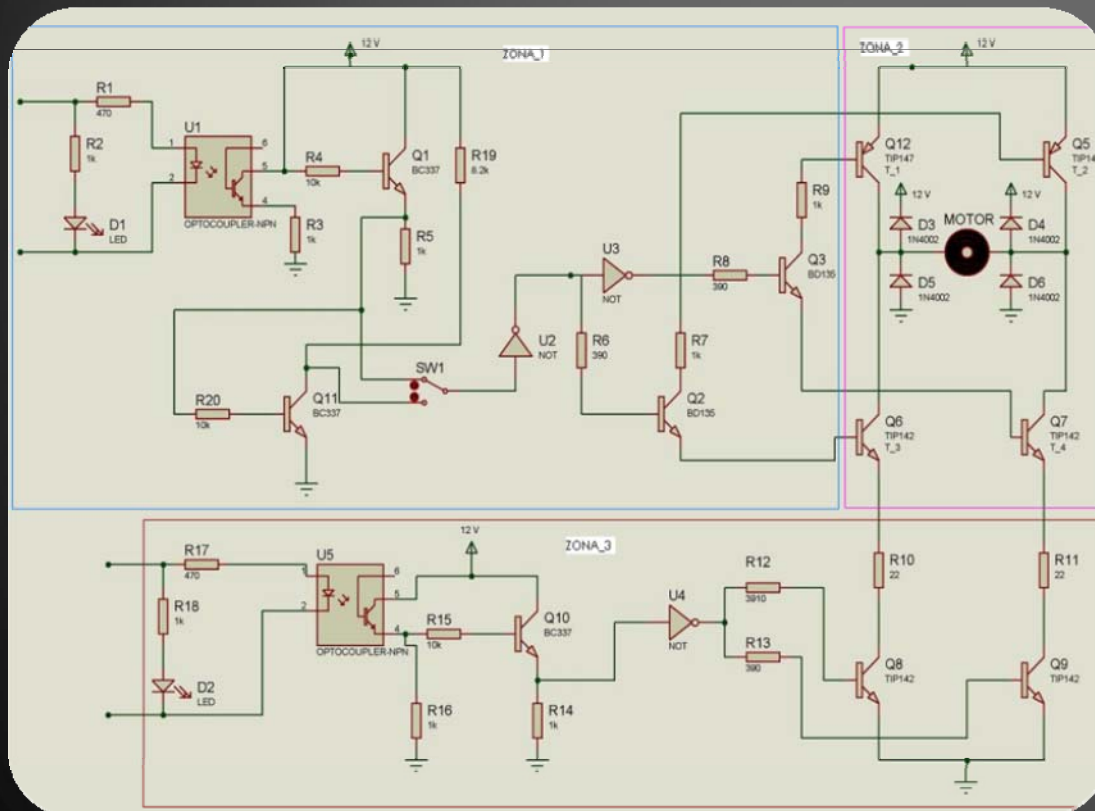


Figure 6. Schematic diagram of the power stage.

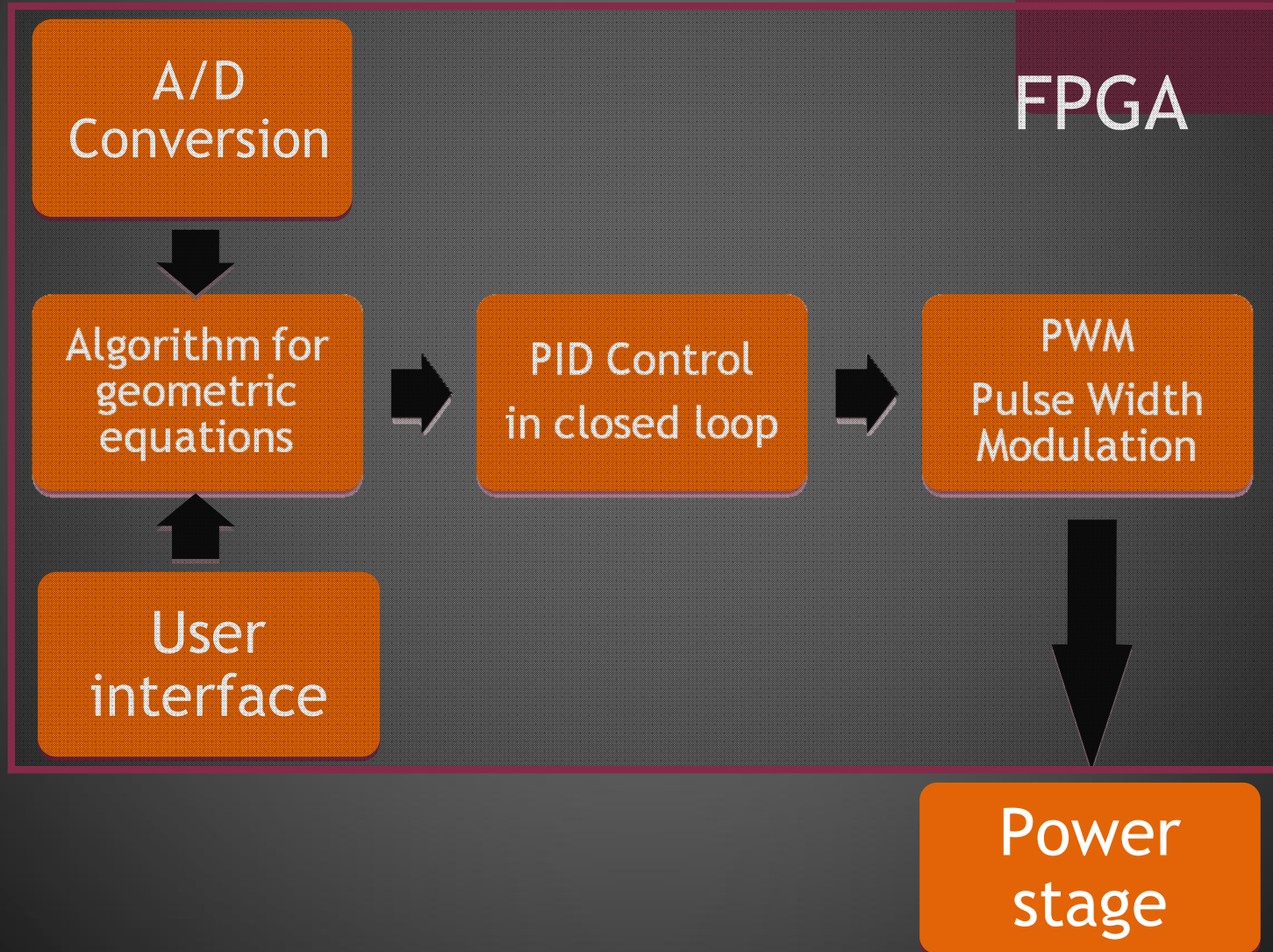
➤ The currents of the PWM signals (of the order of 25mA) are amplified to control *dc* motors

➤ It has 4 channels and provides up to 1A per channel

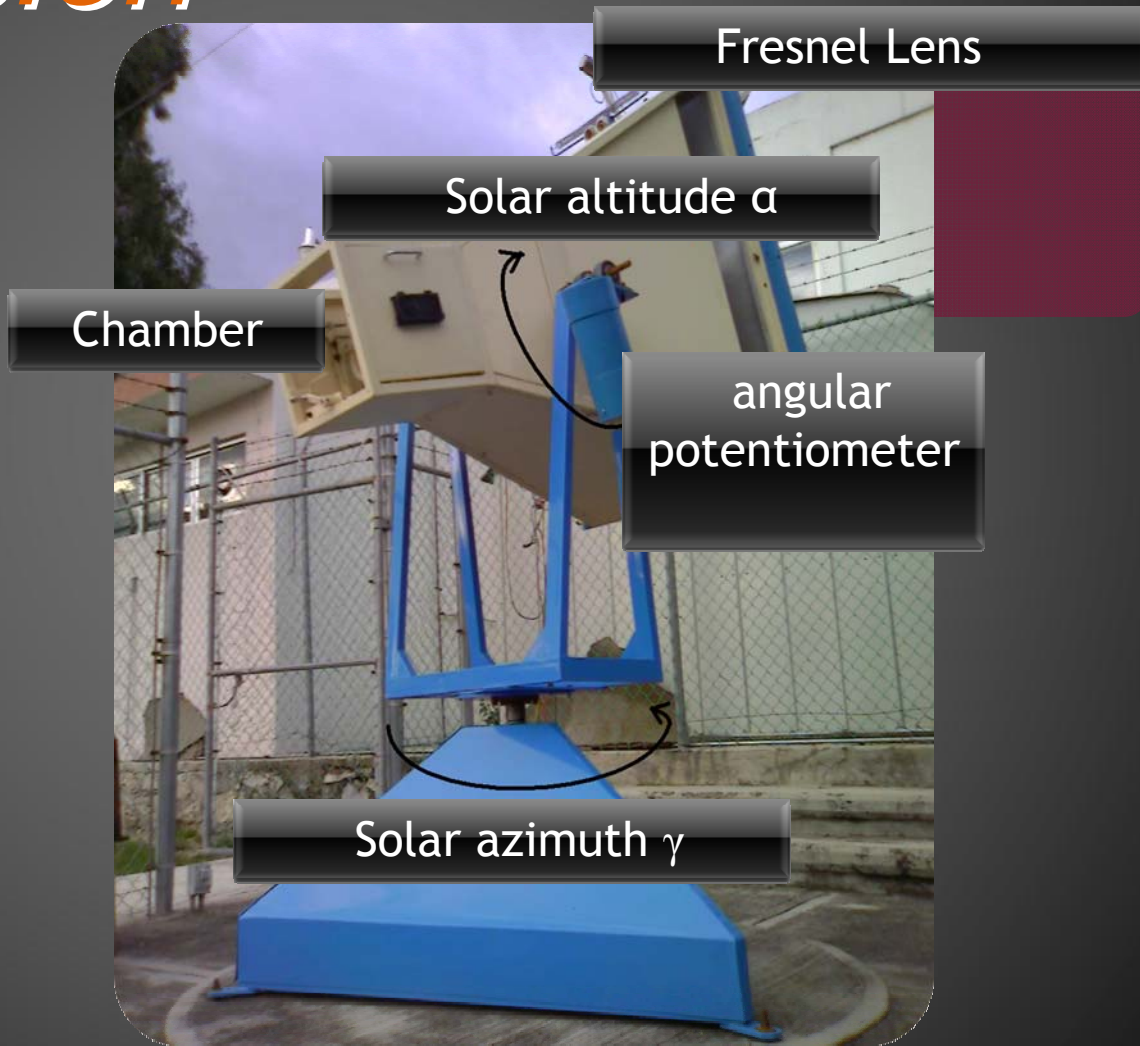
➤ Each channel is controlled by signals compatible with TTL input

➤ Each pair of channels has an enable signal that disconnects the output from them, well, feeding is independent of the control signals and dc motors fed from 5 to 12 Volts.

Method



Discussion



Solar concentrator and its main features

Discussion

Applications

Solar Concentrator is using for:

- synthesis of phthalocyanines,
- wafer oxidation,
- reactions of biofuels,
- thermoelectric generators,
- thermionic cells,
- and photovoltaic cells

CHEMICAL
REACTIONS

DIRECT
CONVERSION

Thank you !

CONTACT PERSON

L.A. Moreno-Coria
acoria@ece.buap.mx

Centro de Investigaciones en Dispositivos Semiconductores, Benemérita Universidad Autónoma de Puebla, 14 Sur y Av. San Claudio, Ciudad Universitaria, C.P. 72570, Puebla, México.

**Also: jgperezl@siu.buap.mx, blanca.soto@icbuap.mx, jose.sosa@icbuap.buap.mx*



Hardware Architecture for Particle Swarm Optimization using Floating-point Arithmetic

Daniel M. Muñoz Arboleda

Ph. D. Student - Mechatronic Systems
GRACO – Automation and Control Group
Mechanical Engineer Department
University of Brasília, D.F., Brazil

4 November 2009
ICTP – Trieste - Italy

About GRACO

The research activities are organized in three areas:

- Automation and Manufacturing process
 - Welding process
 - Manufacturing automation process
- Robotic systems
 - Sensing, mapping and navigation for autonomous mobile robots
 - Computer vision
- Composite and functional materials
 - Shape Memory Alloys – SMA
 - Piezoelectric ceramics sensors



About GRACO

- Staff
 - 6 Ph. D. Professors
 - 15 Master Students
 - 9 Ph. D Students
- Cooperation
 - University of Santa Catarina – Brazil
 - Pontifical Catholic University of Paraná – Brazil
 - University of Uberlandia – Brazil
 - Cranfield University - England
 - INRIA – France
 - ITIV - Universität Karlsruhe – Germany
 - Europäisches Zentrum für Mechatronik
 - PETROBRAS
 - FIAT Brazil
 - Rockwell Automation
 - Eletronorte



About GRACO

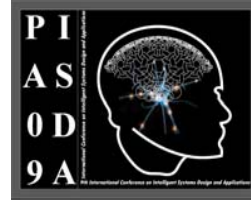
Reconfigurable Systems Laboratory

Research focuses

- High performance computer systems
- Architecture synthesis
- Hardware Synthesis and SoC Integration

Typical projects are from the fields

- Robotics (manipulators and or mobile robots)
- Automotive applications
- Welding processes
- Computer Vision applications
- **Intelligent Systems and Optimization processes**
- **Control systems**
- **Building Automation**



Hardware Architecture for Particle Swarm Optimization using Floating-point Arithmetic

Daniel M. Muñoz Arboleda

Ph. D. Student - Mechatronic Systems
GRACO – Automation and Control Group
Mechanical Engineer Department
University of Brasília, D.F., Brazil

4 November 2009
ICTP – Trieste - Italy

Summary

Introduction

The PSO operation

Hardware implementations

Synthesis results

Simulation results

Conclusions

Future works

Hardware Architecture for PSO using Floating- point Arithmetic

Summary

Introduction

The PSO operation

Hardware
Implementations

Synthesis results

Simulation results


Conclusions

Future works

Introduction

Particle Swarm Optimization (PSO) is a bio-inspired stochastic technique.

- Easy implementation 
- Less computational requirements 
- Non-gradient computation 
- Parallel capabilities 
- **Large elapsed time** 

Parallel implementations of PSO can increase the throughput and improve their performance ! 

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

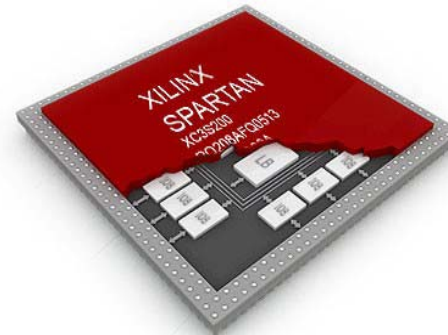
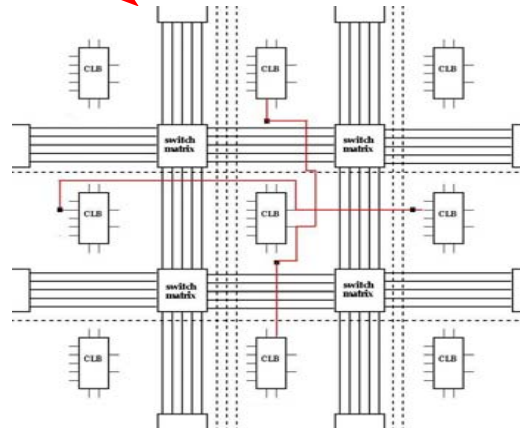
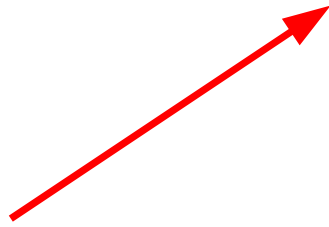
Synthesis results

Simulation results

Conclusions

Future works

Introduction



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Introduction

FPGA implementation of the PSO is a feasible and cheap solution, exploring the parallel capabilities by implementing on a single chip:

- Parallel particles
- Simultaneous computations

HARDWARE IMPLEMENTATION PROBLEM:

FPGAs only provides integer or fixed-point arithmetic, whereas an CPU usually works with a floating-point arithmetic.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Introduction

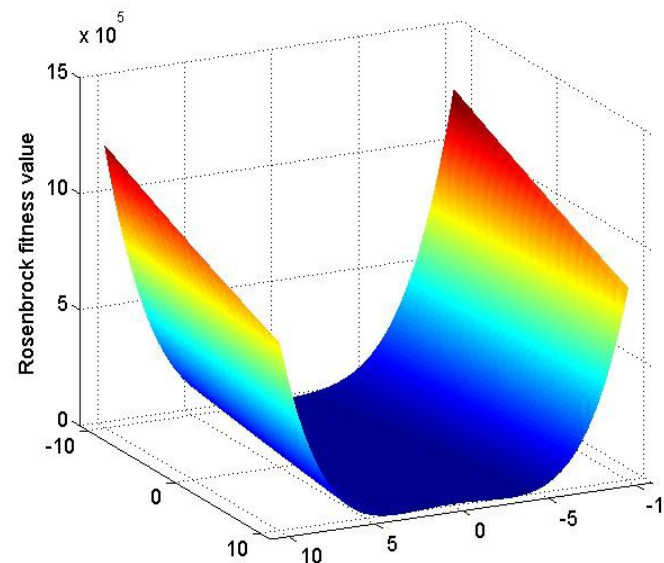
Fixed-point arithmetic can work with high precision by using large bit-width representations.

Floating-point arithmetic is an essential component of high-performance computer systems.

- **High precision** computation. Capabilities to retain its resolution
- Large **dynamic range** to represent small and large real numbers

The choice of using a fixed or floating-point arithmetic depends on whether **the floating-point is needed by the data set?**

Many optimization problems require to work with large and small real numbers.



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

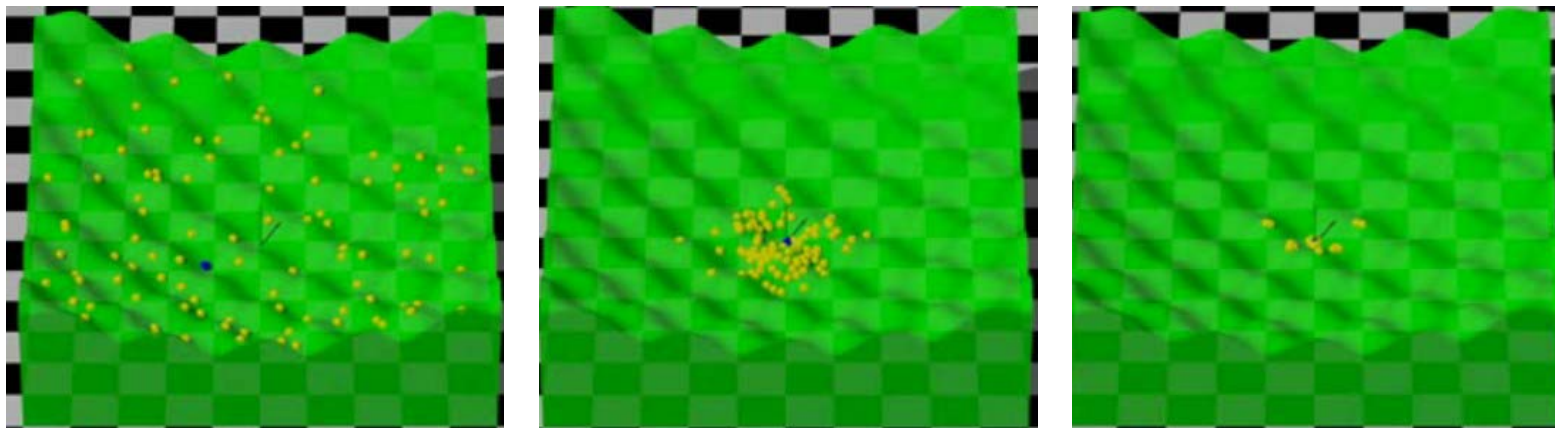
Simulation results

Conclusions

Future works

The PSO Operation

- Population is called swarm.
- Each individual is called particle.
- Mass-less and volume-less.
- Each particle i has a current vector position x_i , a personal best position vector y_i and a velocity vector v_i .
- The PSO shares the global best position y_s among all the particles.



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

6 / 29

The PSO Operation

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)} \quad (1)$$

$$v_{ij}^{(t+1)} = wv_{ij}^{(t)} + c_1 r_1 (y_{ij}^{(t)} - x_{ij}^{(t)}) + c_2 r_2 (y_s^{(t)} - x_{ij}^{(t)}) \quad (2)$$

w : inertia weight. Linearly decrease [0.9 to 0.1]

c_1 : cognitive coefficient

c_2 : social coefficient

r_1, r_2 : uniform random numbers [0 to 1]

The velocities are clamped to the range $[v_{\min}, v_{\max}]$

Large c_1 values provide particles with a high self confidence in their experience.

Large c_2 values provide particles with high confidence in the swarm.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Hardware Implementations

The IEEE-754 standard



Three binary components:

- Sign S
- Exponent E
- Mantissa M

This standard allows the user to work not only with single (32 bits) or double (64 bits) precisions, but also with the suitable bit-width according to the applications requirements.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

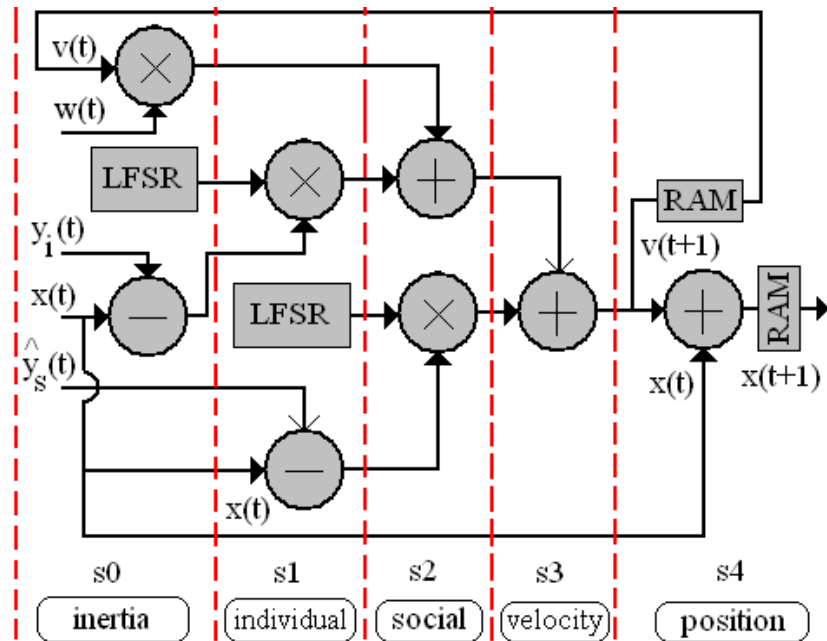
Synthesis results

Simulation results

Conclusions

Future works

Particle architecture



$$v_{ij}^{(t+1)} = wv_{ij}^{(t)} + r_1 (y_{ij}^{(t)} - x_{ij}^{(t)}) + r_2 (y_s^{(t)} - x_{ij}^{(t)})$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}$$

- One floating-point addition/subtraction (*FPadd*)
 - One floating-point multiplier (*FPmul*)
 - One floating-point Linear Feedback Shift Register
- The FP operators have been previously developed.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

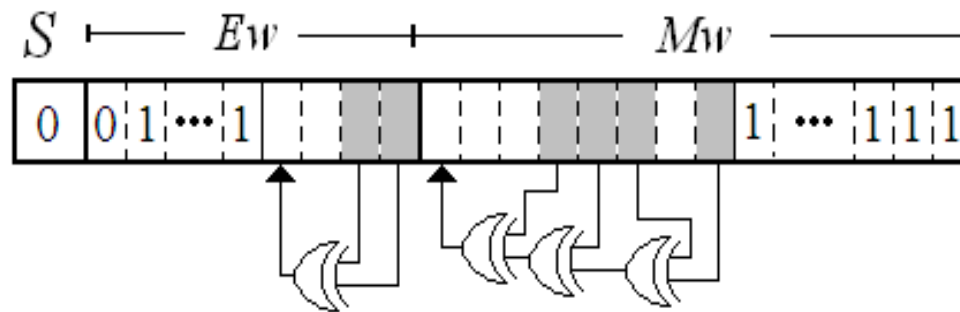
Synthesis results

Simulation results

Conclusions

Future works

Floating-point LFSR



A Linear Feedback Shift Register (LFSR) is a shift register based technique.

- Several bits (*taps*) are chosen as feedback function.
- By selecting the appropriate *taps* the bit sequence works as a pseudo-random number generator.
- By selecting the appropriate changing bits in both the exponent and mantissa words, the LFSR works in the desired range. $U[0, c_1]$ or $U[0, c_2]$

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

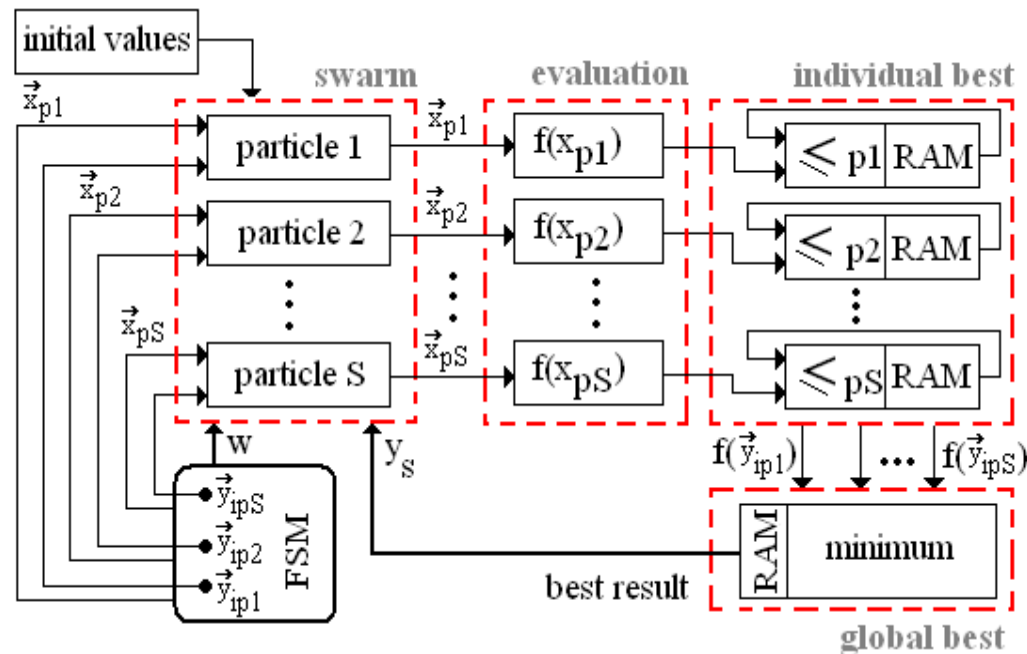
Synthesis results

Simulation results

Conclusions

Future works

General HPSO architecture



The HPSO was developed using a Finite State Machine (FSM) approach. It is composed of:

- A swarm unit (S parallel particles)
- An evaluation unit (S parallel fitness functions)
- An individual best detection unit
- A global best detection unit

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Fitness functions

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

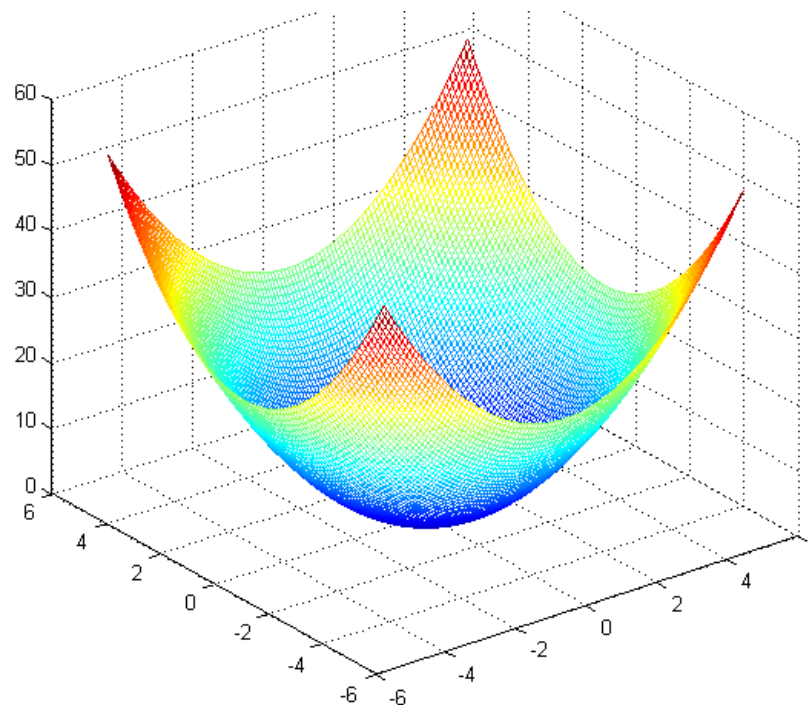
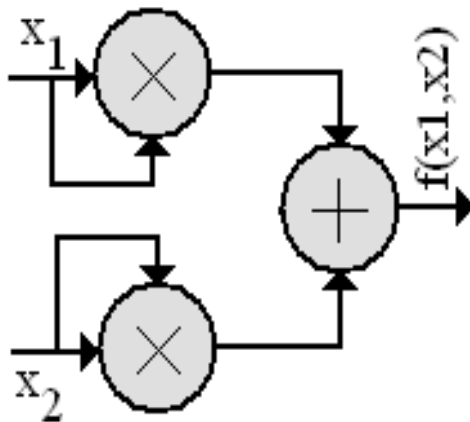
Simulation results

Conclusions

Future works

The proposed HPSO architecture has been validated using a swarm composed of 4 particles ($S=4$) and four well-know benchmark problems for a two-dimensional situation ($N=2$), also implemented in hardware.

- Sphere: $f(\vec{x}) = \sum_{i=1}^N x_i^2$ (4)

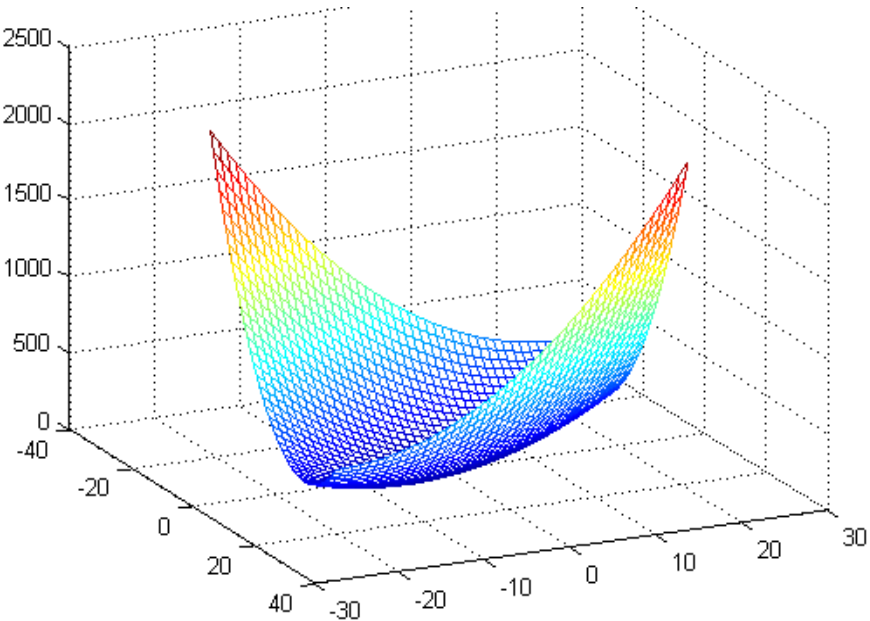
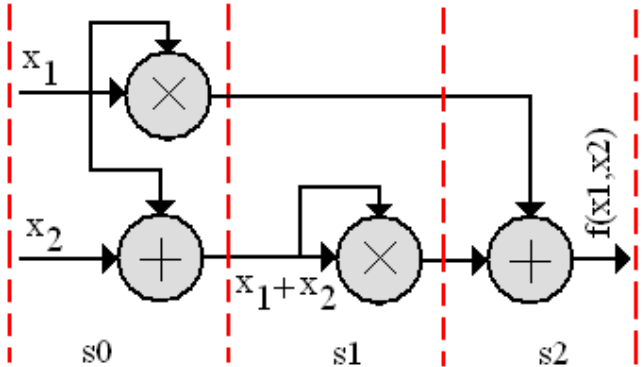


Fitness functions

- Quadric:

$$f(\vec{x}) = \sum_{i=1}^N \left(\sum_{j=1}^i x_j \right)^2$$

(5)

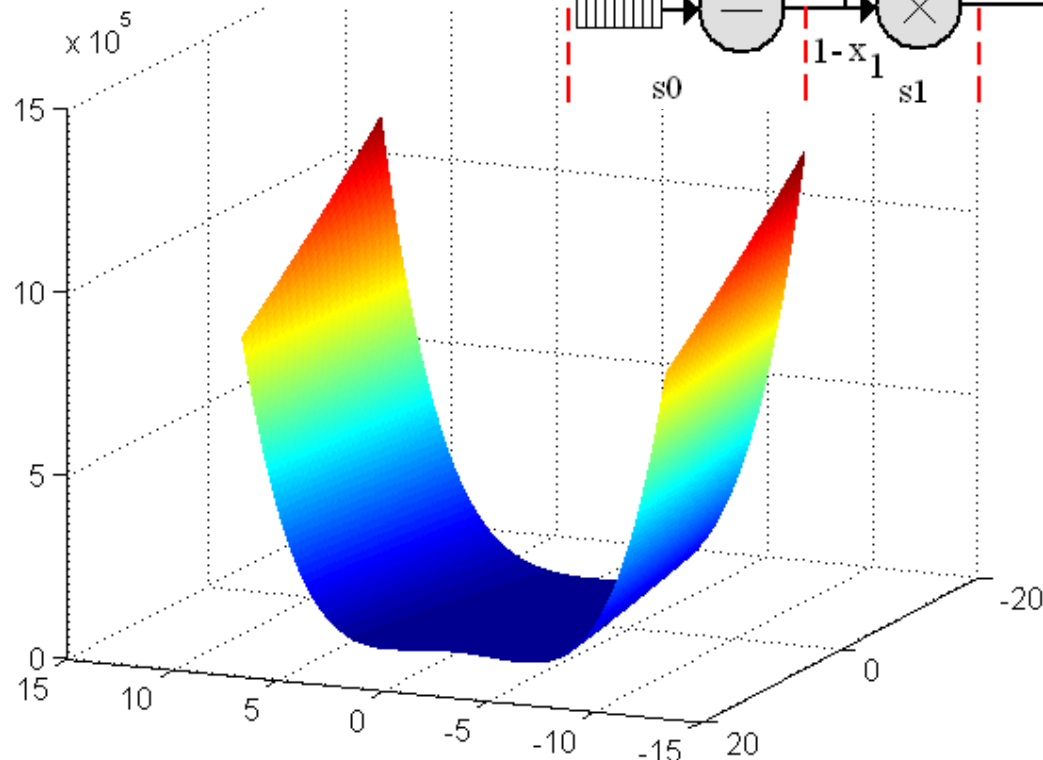
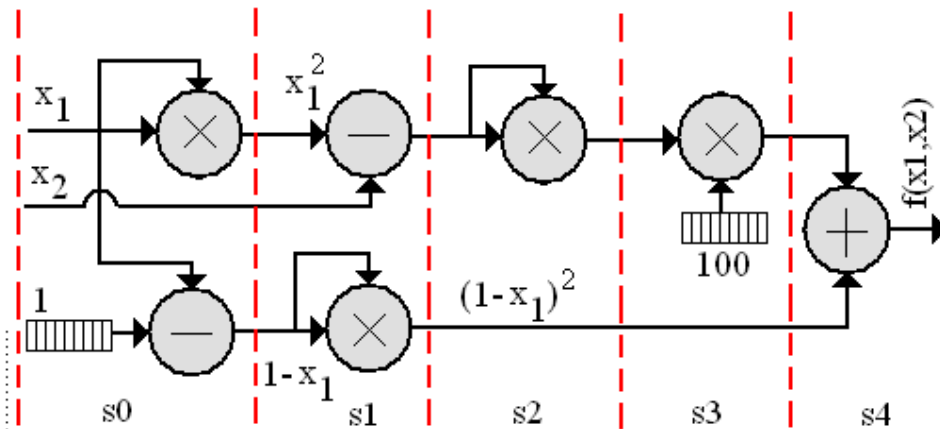


Hardware Architecture for PSO using Floating-point Arithmetic

- Summary
- Introduction
- The PSO operation
- Hardware Implementations**
- Synthesis results
- Simulation results
- Conclusions
- Future works

Fitness functions

- Rosenbrock:
$$f(\vec{x}) = \sum_{i=1}^{N/2} 100 \cdot (x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1})^2 \quad (6)$$



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

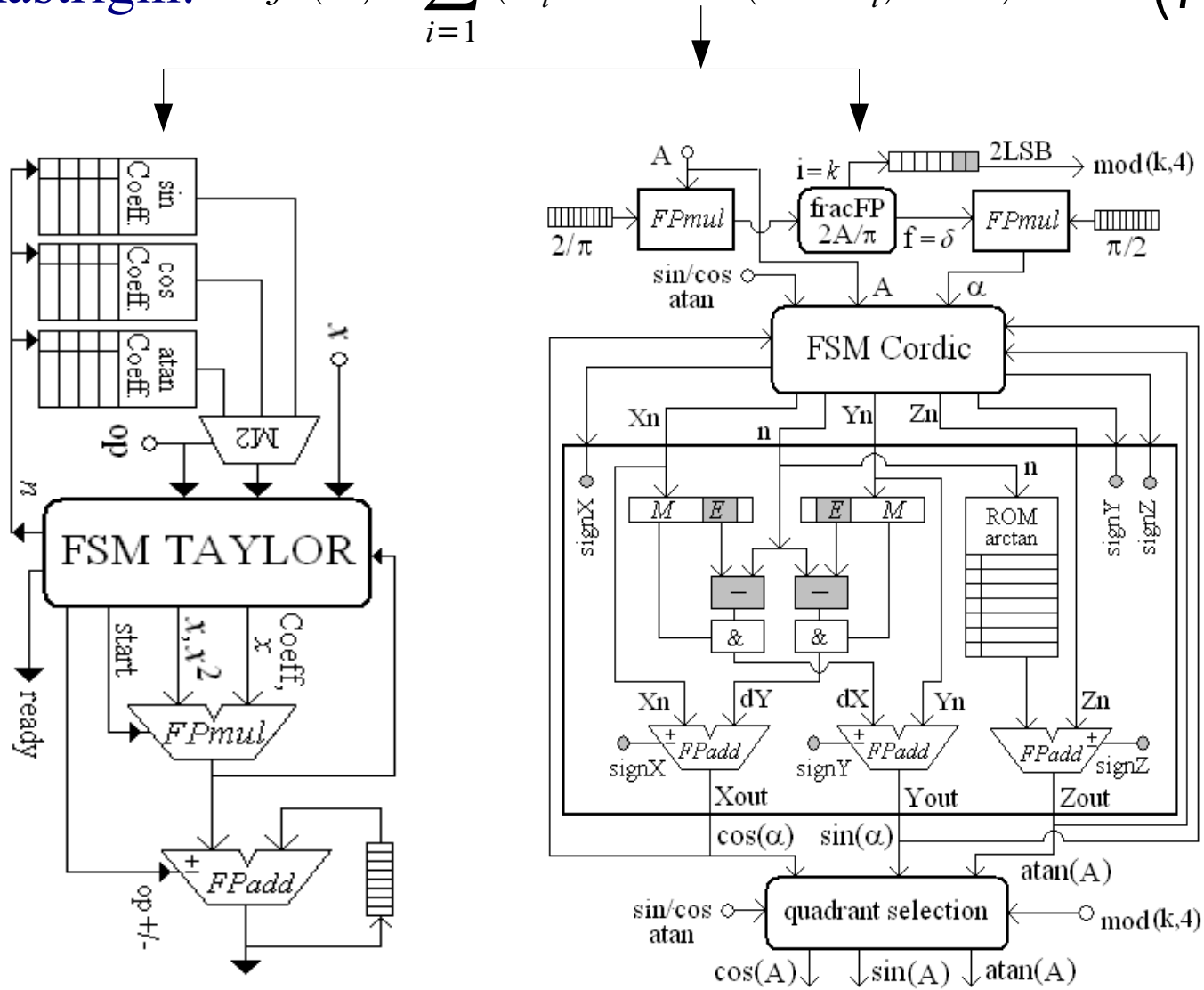
Simulation results

Conclusions

Future works

Fitness functions

• Rastrigin:
$$f(\vec{x}) = \sum_{i=1}^N (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (7)$$



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

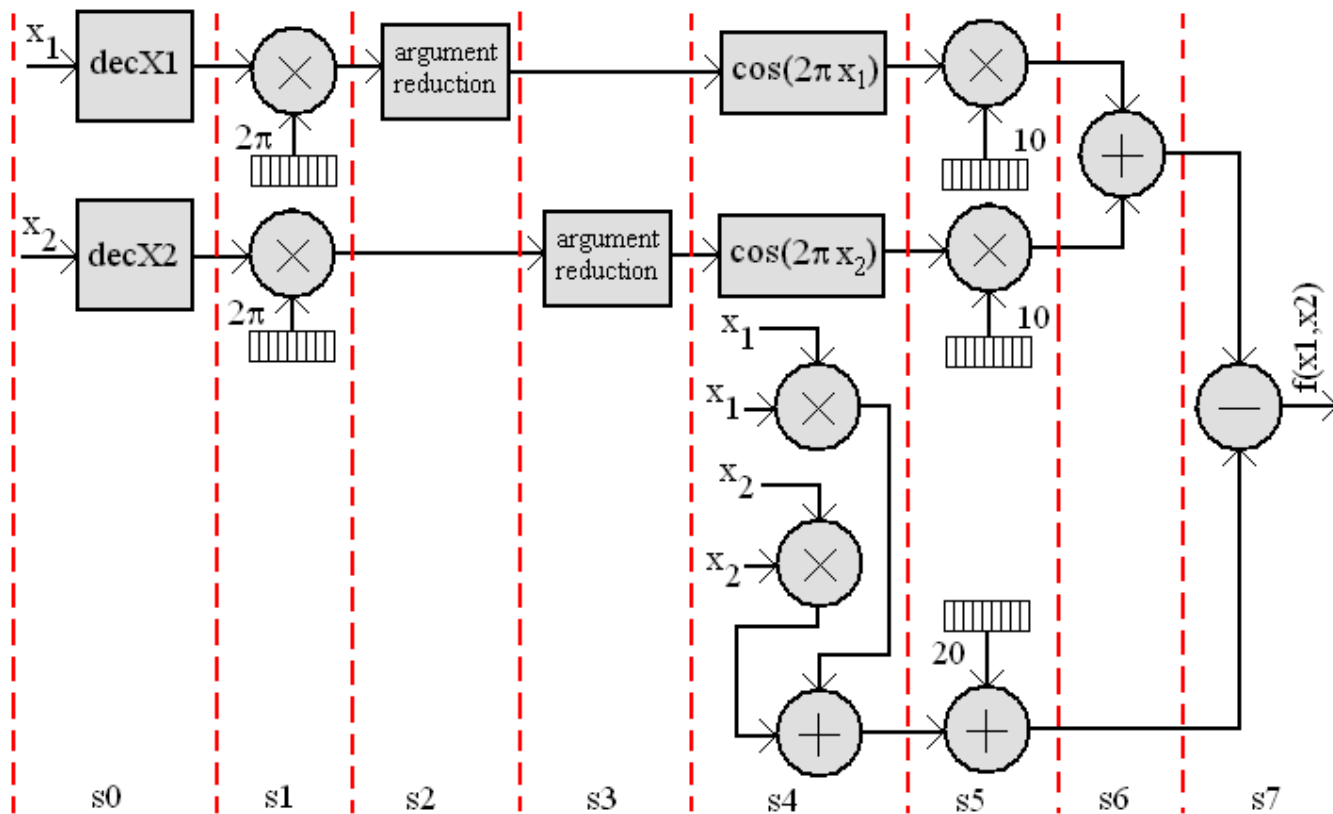
Simulation results

Conclusions

Future works

Fitness functions

- Rastrigin:
$$f(\vec{x}) = \sum_{i=1}^N (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (7)$$



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

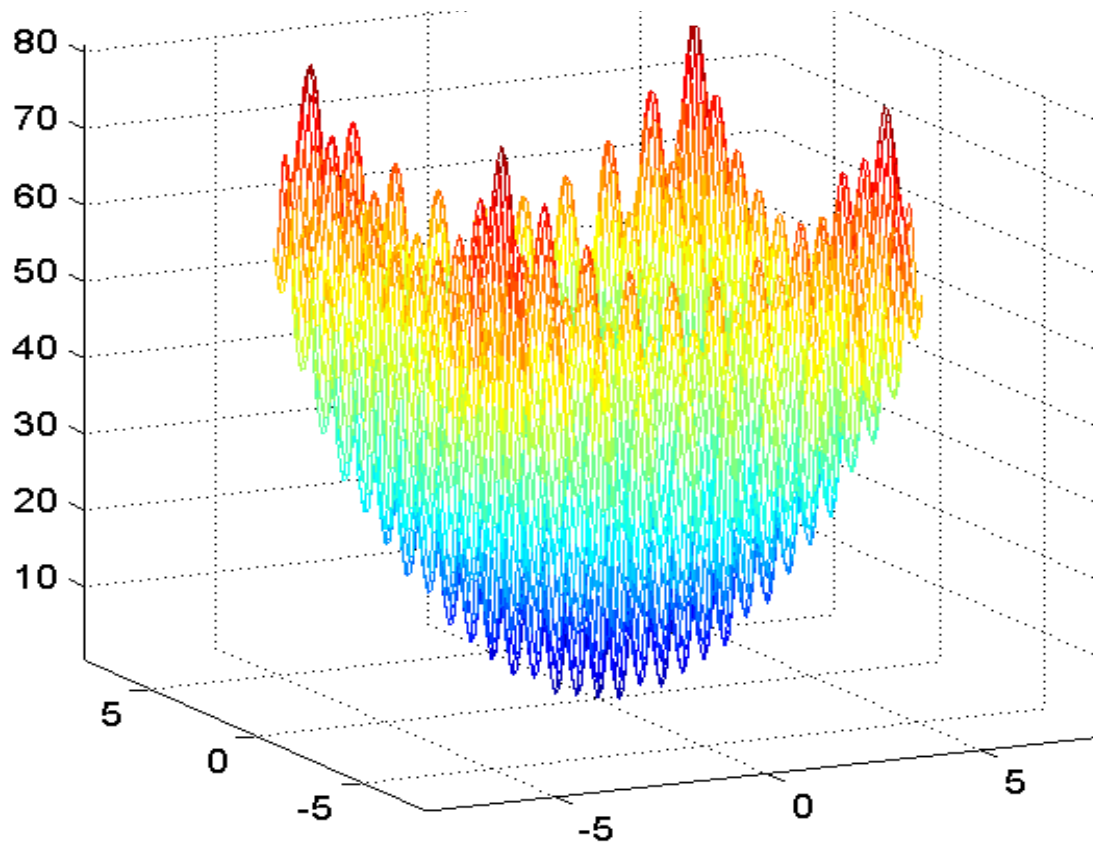
Simulation results

Conclusions

Future works

Fitness functions

• Rastrigin:
$$f(\vec{x}) = \sum_{i=1}^N (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (7)$$



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Results

The same experiment conditions were used for both the hardware and software implementations.

Parameter	Value
Max. number iterations	500
Cognitive Coefficient c_1	2.0
Social Coefficient c_2	2.0
Inertia weight	[0.9 to 0.1]
Initial velocity	0.5
Maximum velocity	3.0
Domain range Rosenbrock	[-16.02 16.02]
Domain range remain functions	[-5.12 5.12]

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Synthesis Results

The proposed HPSO architecture was described in VHDL and synthesized for a FPGA Virtex5 family (chip xc5vlx330).

HPSO single precision (32 bits)

Implemented FP-core	Flip-Flops Max 207360	LUTs Max 207360	DSP MUL Max 192	Freq. MHz
Particle	358 (0.17%)	1298 (0.63%)	2 (1.04%)	97.5
PSO-Sphere	4246 (2.05%)	12058 (5.81%)	26 (13.5%)	90.5
PSO-Quadric	4441 (2.14%)	12578 (6.07%)	18 (9.38%)	91.8
PSO-Rosenbrock	3377 (1.63%)	8423 (4.06%)	10 (5.21%)	94.7
PSO-Rastrigin	8474 (4.09%)	27067 (13.1%)	42 (21.9%)	90.6

The HPSO is feasibility implemented for all bit-width. The HPSO requires a large number of embedded multipliers. The complexity of the fitness function affects directly the scalability of the HPSO architecture.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

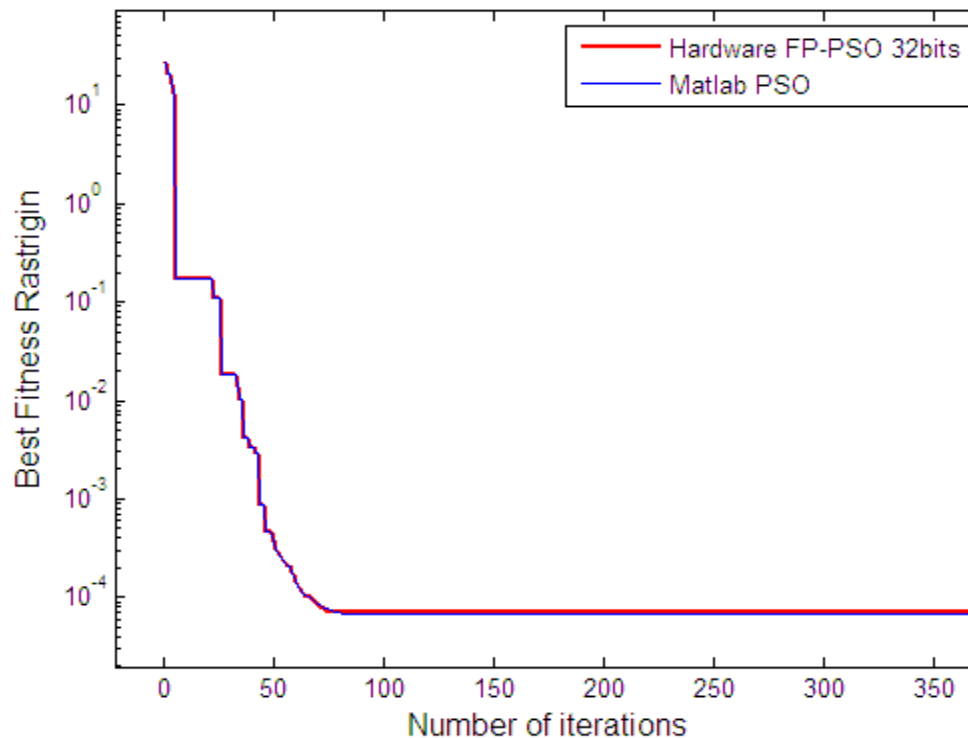
Validation of the FPLib

PSO for a Rastrigin function without random factor

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)} \quad (1)$$

$$v_{ij}^{(t+1)} = wv_{ij}^{(t)} + r_1(y_{ij}^{(t)} - x_{ij}^{(t)}) + r_2(y_s^{(t)} - x_{ij}^{(t)}) \quad (3)$$

$$f(\vec{x}) = \sum_{i=1}^N (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (7)$$



Hardware
Architecture for
PSO using Floating-
point Arithmetic

Summary

Introduction

The PSO operation

Hardware
Implementations

Synthesis results

Simulation results

Conclusions

Future works

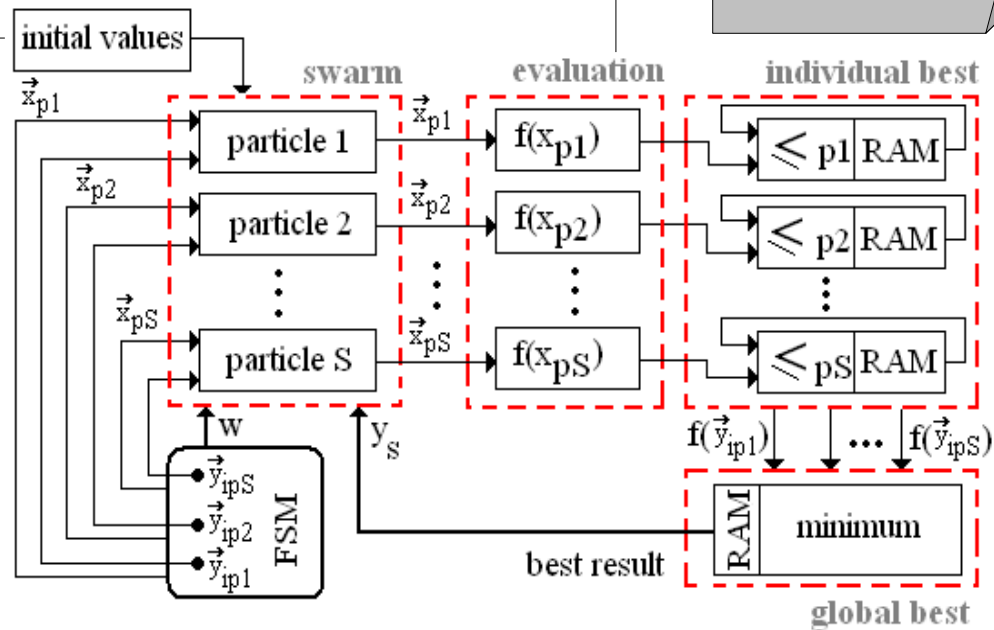
Simulation Results

For each benchmark problem the HPSO 64 bits were simulated using different initial positions conditions.

The same initial position for each single run was used in a Matlab[®] implementation (64 bits) and the best fitness results were compared.

- Sphere
- Quadric
- Rastrigin
- Rosenbrock

10 different initial positions



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

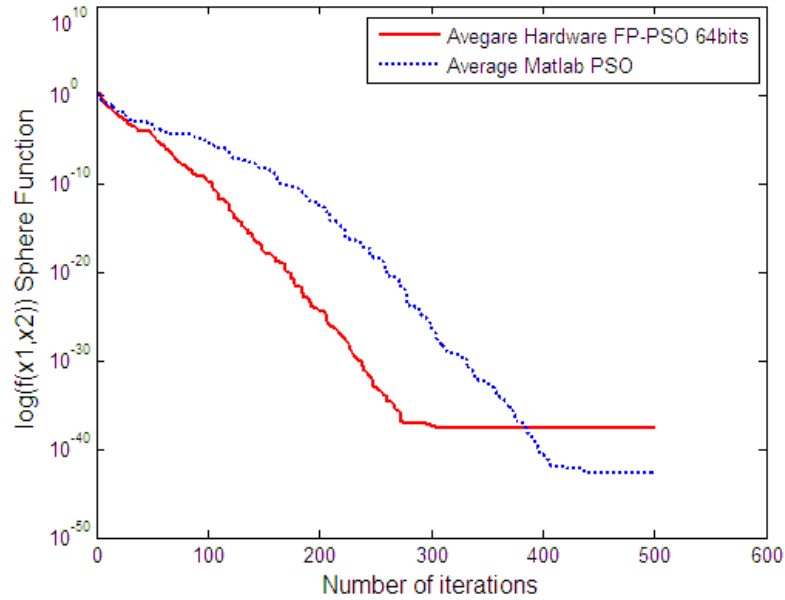
Synthesis results

Simulation results

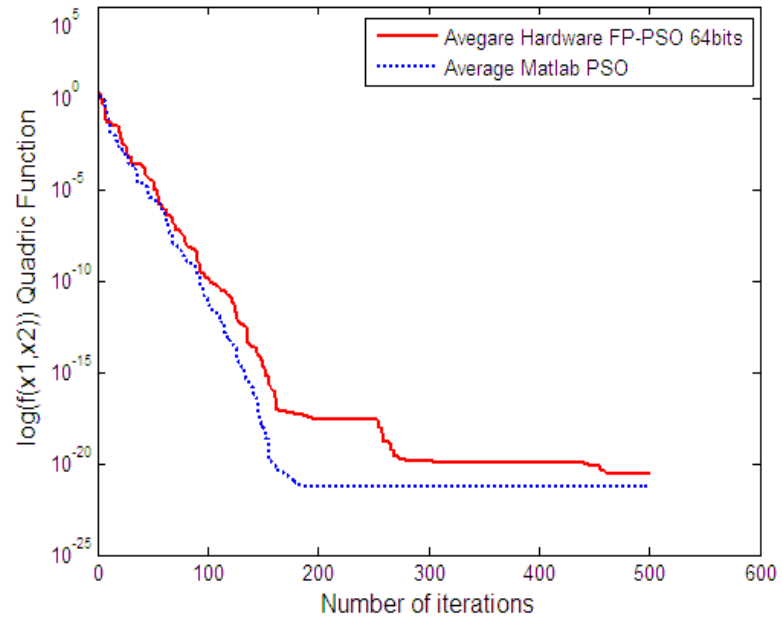
Conclusions

Future works

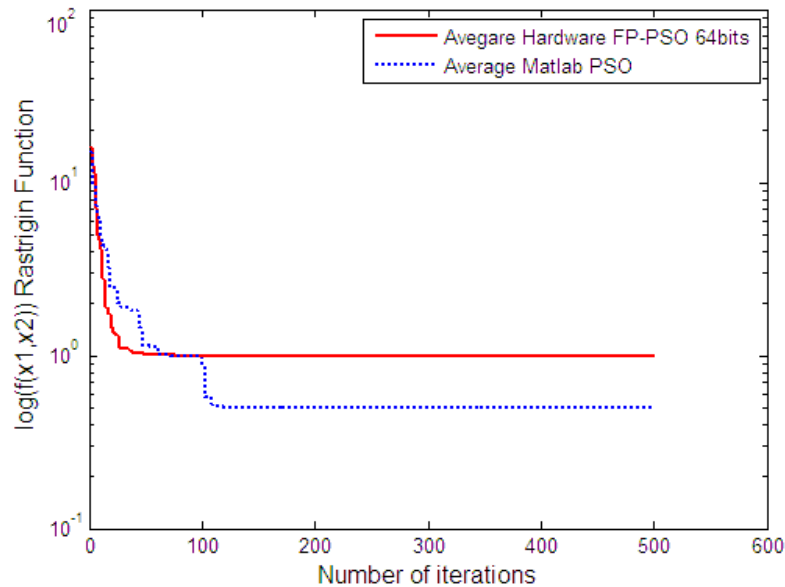
Sphere function



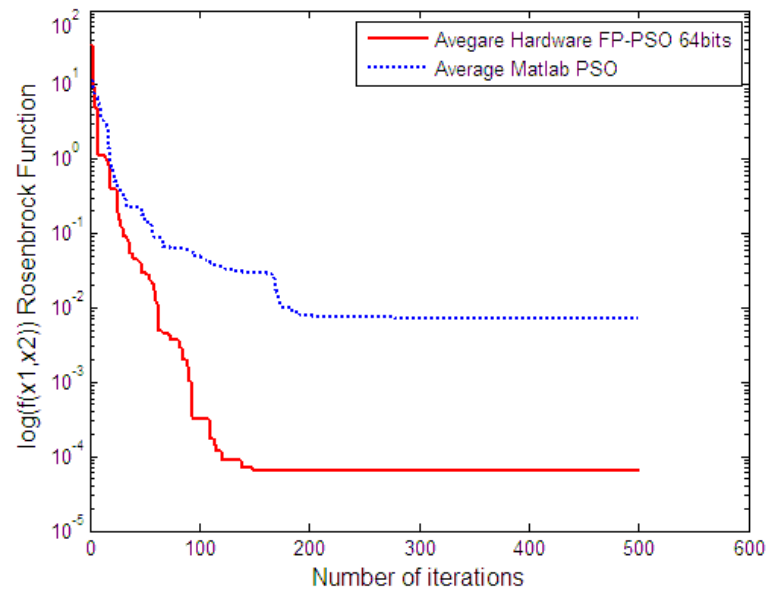
Quadric function



Rastrigin function



Rosenbrock function



Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

HPSO implementation:
FPGA Xilinx at **50MHz**



Matlab[®] implementation:
Intel Core Duo at **1.6 GHz**, 1GB
of RAM, Windows XP OS.



Elapsed time per iteration

Case problem	Hardware	Software	Times faster
PSO-sphere	0.94 us	120.20 us	127
PSO-quadric	0.98 us	127.03 us	129
PSO-rosenbrock	1.10 us	105.74 us	96
PSO-rastrigin	1.64 us	128.40 us	78

HPSO achieves similar fitness values than software implementations; however, at the worst case, the HPSO is around 78 times faster than Matlab[®] implementation.

**Hardware
Architecture for
PSO using Floating-
point Arithmetic**

Summary

Introduction

The PSO operation

Hardware
Implementations

Synthesis results

Simulation results

Conclusions

Future works

Conclusions

- A hardware architecture of the PSO algorithm (HPSO) have been implemented using a floating-point arithmetic.
- The floating-point arithmetic allows the algorithm to work with very large and small real numbers with a high precision.
- The area cost has a significant increase with the number of particles. The number of dimensions increase the elapsed time and has a minor effect in the area cost.
- The HPSO achieves similar fitness values than a software implementation. The main difference is due to the random number generator technique.
- The best improvement case is for the Quadric function (127 times faster per iteration). The worst improvement case is for Rastrigin problem (around 78 times faster per iteration).

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Future works

- Scalability study for different number of particles, bit-width representations and number of dimensions.
- Application to different engineering problems: Robotics, Control and Automation, System Identification.
- A hardware implementation of an **adaptive PSO** algorithm could improve the performance, guarantying convergence.
- In order to accelerate the design process, a **PSO VHDL CODE GENERATOR** is a suitable implementation.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

References

- J. Schuttle, J. Reinblot, B. Fregly, R. Haftka and A. George, "Parallel global optimization with the particle swarm algorithm," Int. J. Numer. Methods Eng. Vol. 6, No. 13, pp. 2296-2315, 2004.
- P. Reynolds, R. Duren, M. Trumbo, and R. Marks, "FPGA implementation of particle swarm optimization for inversion of large neural networks," Proc. IEEE Swarm Intelligence Symp., 2005, pp. 389-392.
- P. Palangpour, G. Venayagamoorthy and S. Smith, "Particle Swarm Optimization: A Hardware Implementation", in International Conference on Computer Design, 2009, pp. 134-139.
- D. Sanchez, D. Muñoz, C. Llanos, M. Ayala-Rincón, "Parameterizable Floating-point Library for Arithmetic Operations in FPGAs", in Proc. ACM Inter. Symp. on Int. Circuits and Syst. Design, 2009, pp. 253-258.
- D. Muñoz, D. Sanchez, C. Llanos, M. Ayala-Rincón, "Tradeoff of FPGA Design of Floating-point Transcendental Functions", in Proc. IEEE Int. Conf. on Very Large Scale Integration, 2009, pp. 1-4.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Acknowledgments



The authors gratefully acknowledge the support from Conselho Nacional de Desenvolvimento Científico e Tecnológico /PNM.



We acknowledge the financial support of this presentation from Fundação de Empreendimentos Científicos e Tecnológicos.

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Questions ?

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

Hardware Implementations

Synthesis results

Simulation results

Conclusions

Future works

Hardware Architecture for PSO using Floating-point Arithmetic

Summary

Introduction

The PSO operation

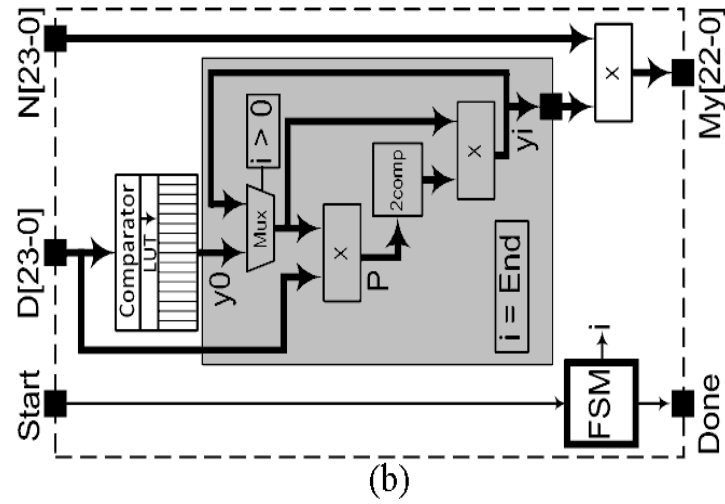
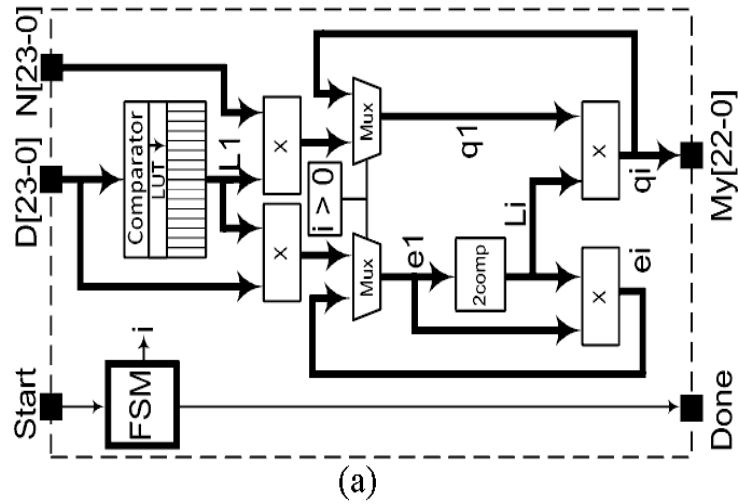
Hardware Implementations

Synthesis results

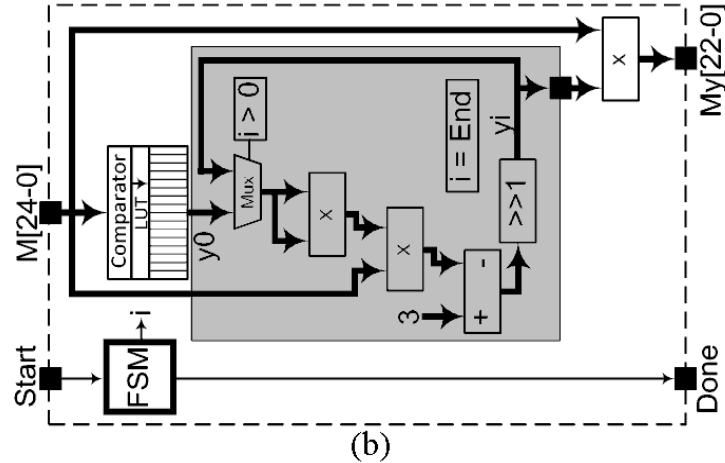
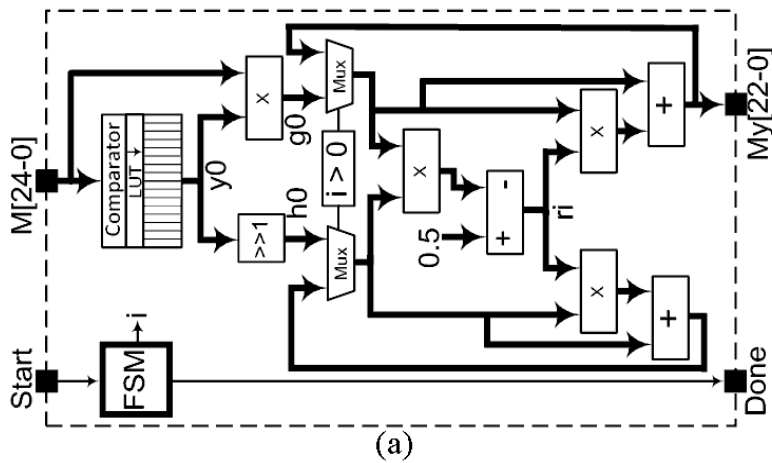
Simulation results

Conclusions

Future works



Division: (a)Goldschmidt (b) Newton-Rapson



Square root: (a)Goldschmidt (b) Newton-Rapson

FPGA leap from micro- controllers

Presented by: Arnold Bett
Kenya

Institution affiliated to

- University of Nairobi
- Physics department

Type of institution

- Educational

Institution goals/ projects

- Teach students
- Provide instrumentation services to the country

Specialization

- Analogue and digital circuit
- Micro-controller applications programming
- Microcontroller development kit production
- Circuit board design and production
- Scientific instruments development

challenges

- As a country
 - Level of digital electronics at a low level
 - There are few specialists in FPGA/ microcontrollers
 - Lack of funding by government and institution
- As an institution
 - curriculum change takes time to be change
 - Analogue electronics is preferred to be taught
 - Few lecturers that teach embedded programming

Physics department programs

- Develop microprocessor lab
- Build capacity in the following areas:
 - Multi layer circuit boards
 - Microcontroller
 - Digital signal processors
 - FPGA

Projects directed towards FPGA

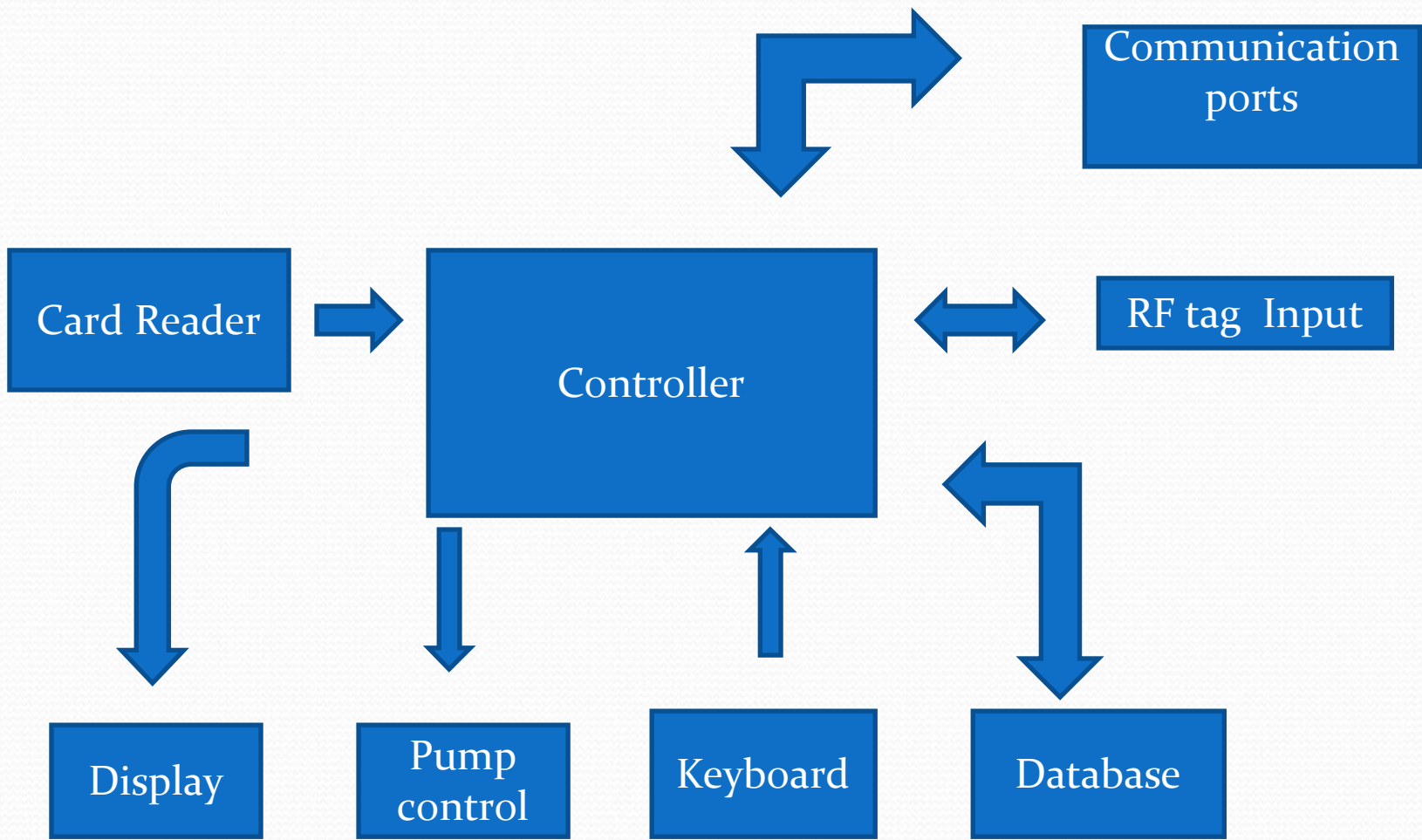
- Fuel pump Dispenser with embedded distributed system

Details.....

- Dispenser controller board
- Memory for capturing and reading data
- RF – tag for car identification
- Magnetic card reader or smart card reader
- Ethernet link. Wired and wireless

Prototype board characteristics

- Microcontroller
 - Z8 encore xp 64k
 - 80 pins
- 4 x 4 Matrix Keyboard
- Led display
- 4 seven segment display
- rs-232 communication
- I₂C interface



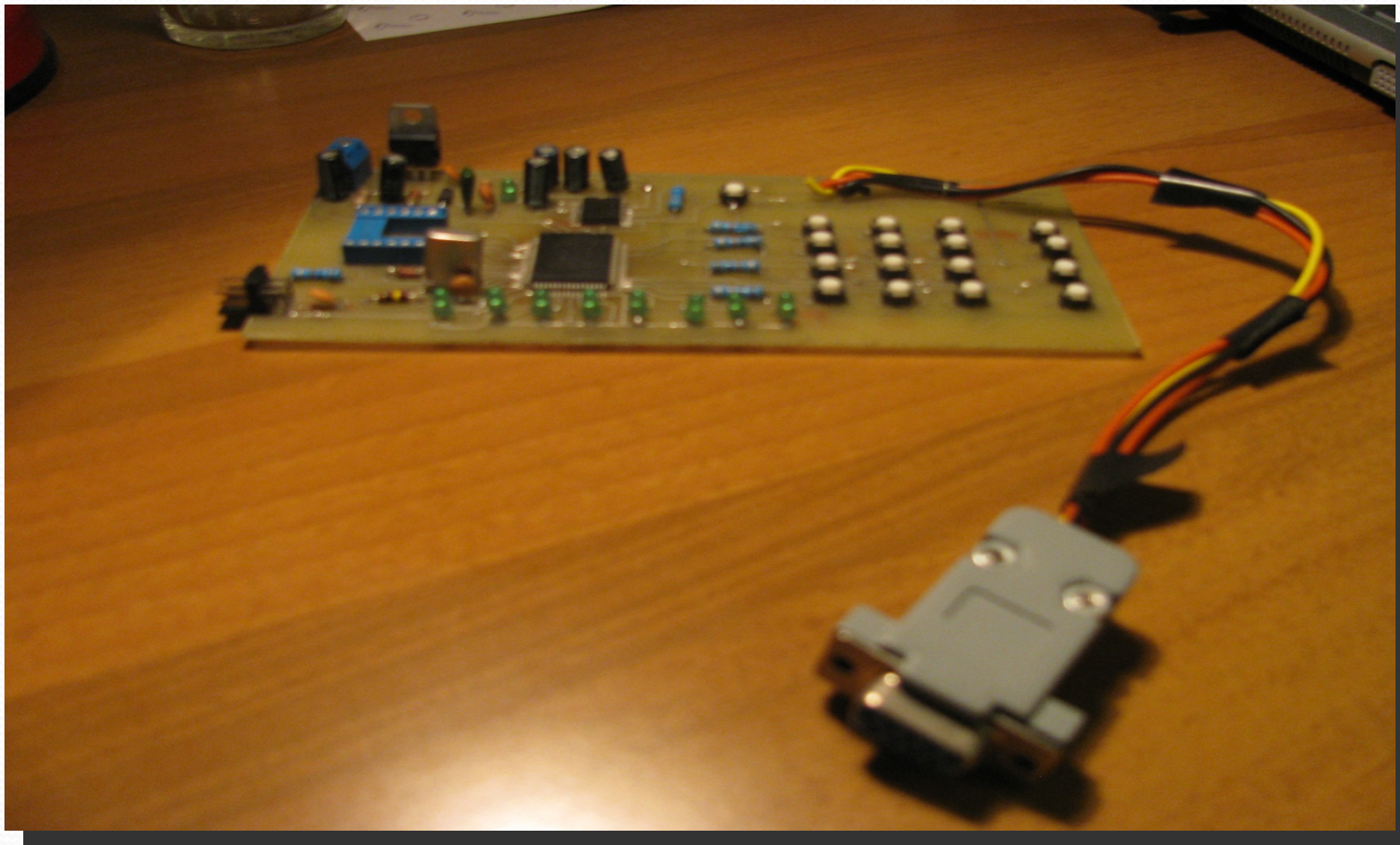
Final Product

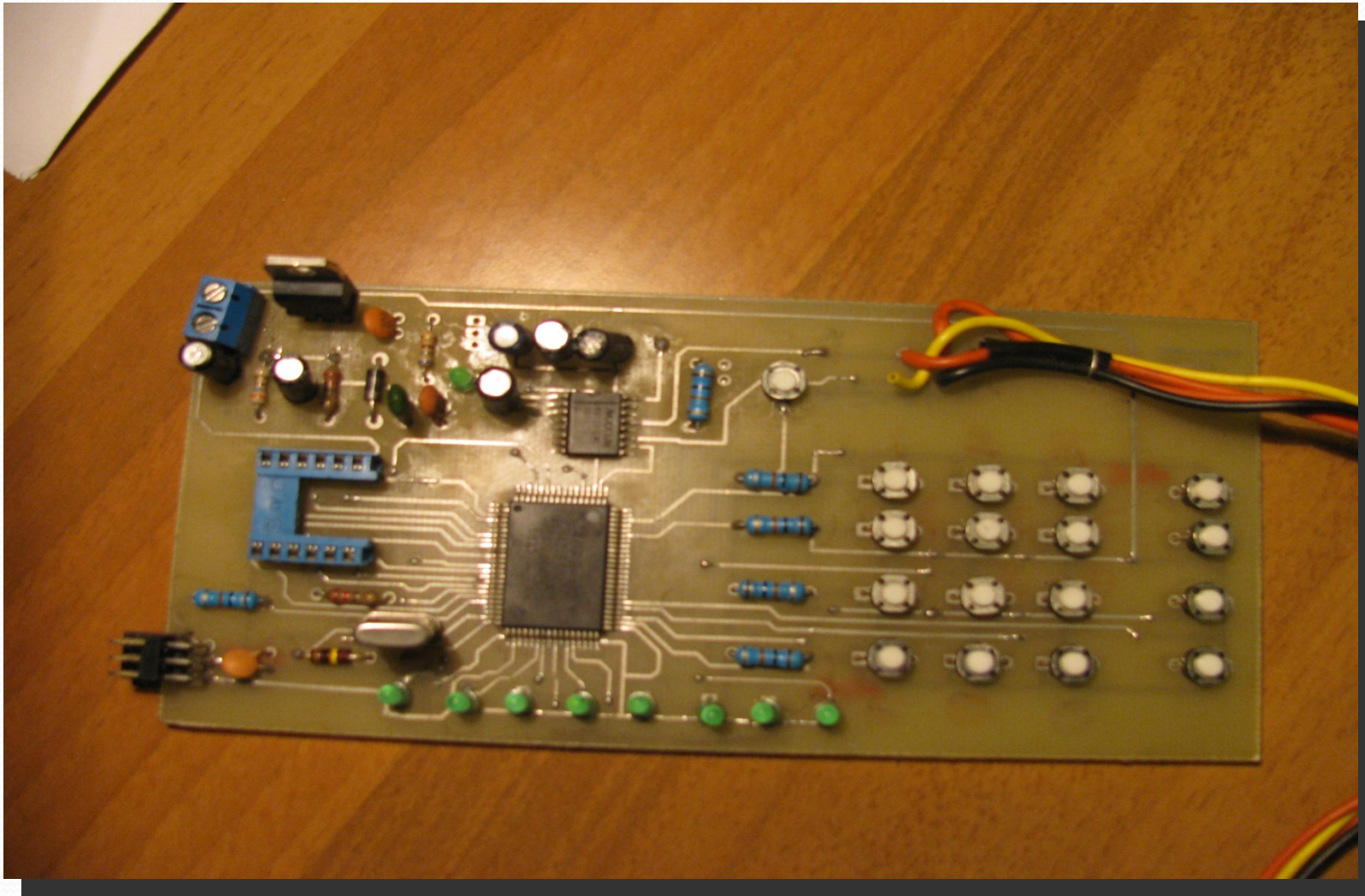
- Cost efficient
 - Very few external parts
 - Minimal hardware material use
- Easy to implement
 - Plug and play if possible
- Distributed system
 - Control
 - Information read and write

How to achieve better results

- Outsource circuit board production
- Use of FPGA
- Reduce external components

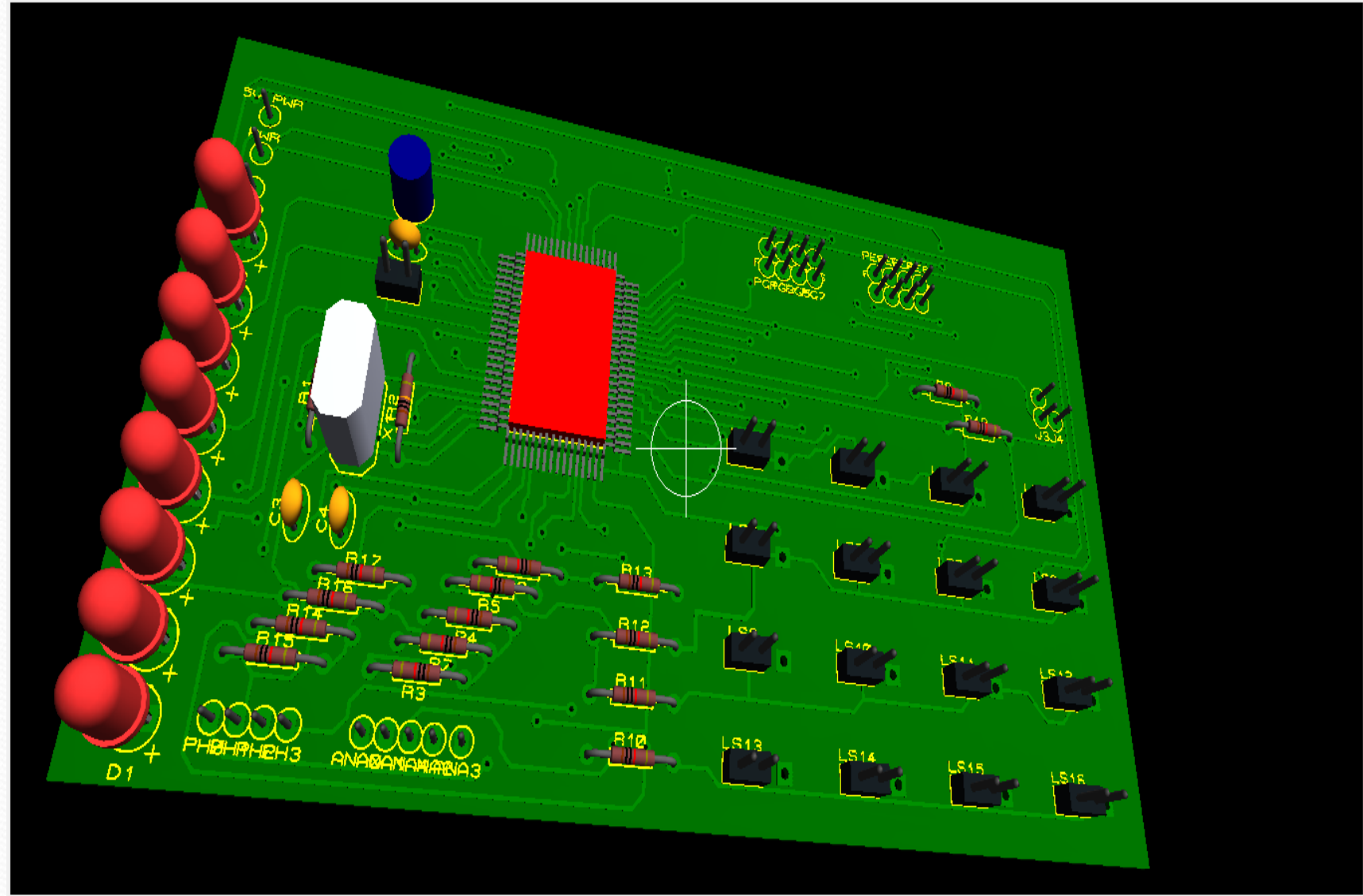
Prototype photos

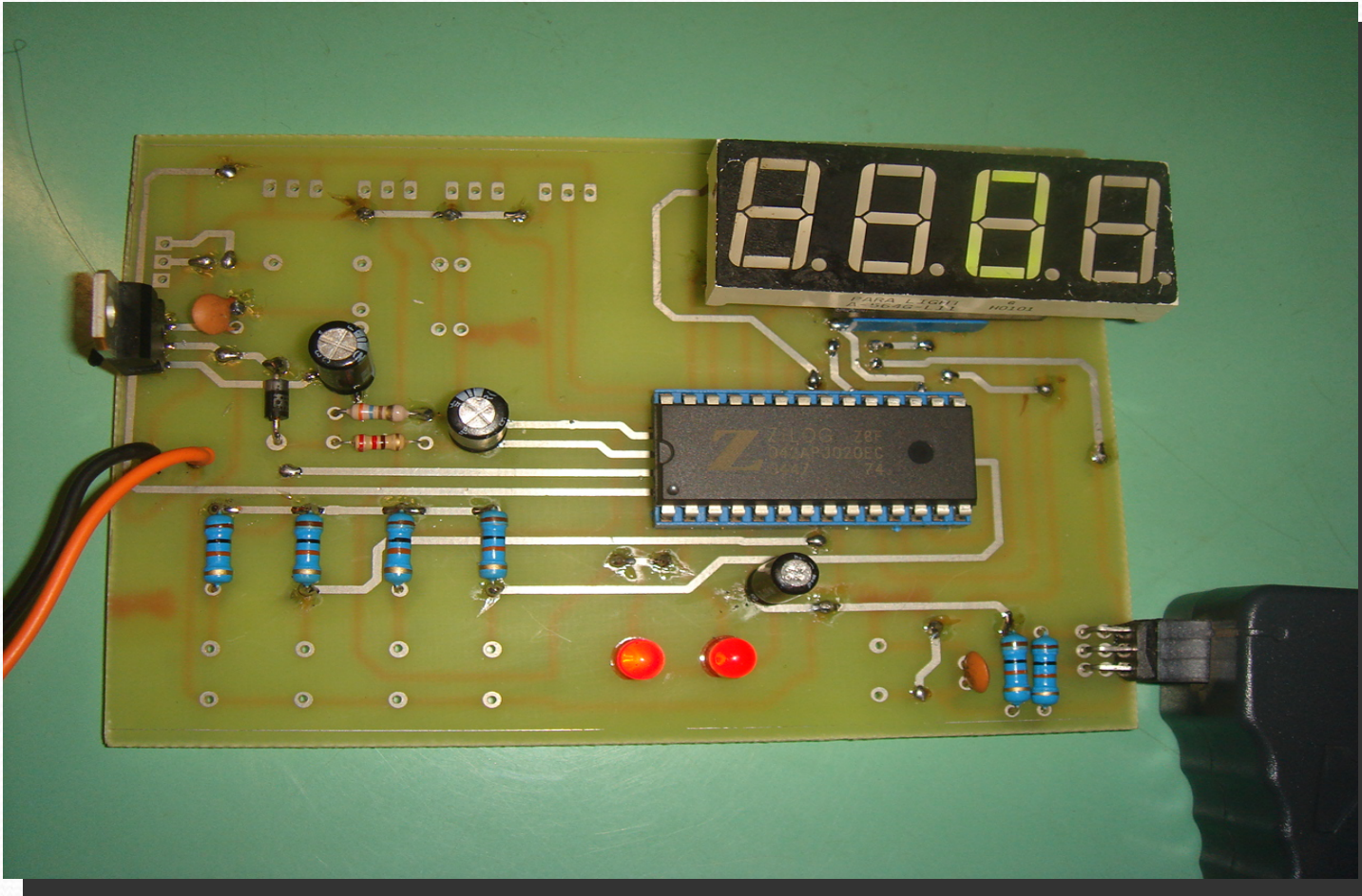




zilog Z8F643 V3 - 3D Visualization

File View Settings Help





The Big jump

- With the realization of the good side of life in the FPGA field its time to make the big jump

What I feel about Actel Fusion



It Matters!.....



Something to think about



Time to take a big jump into FPGA





Thank you for your attention



M. V. LOMONOSOV MOSCOW STATE UNIVERSITY
FACULTY OF PHYSICS

Methodology of FPGA Design

Lock-In Amplifier

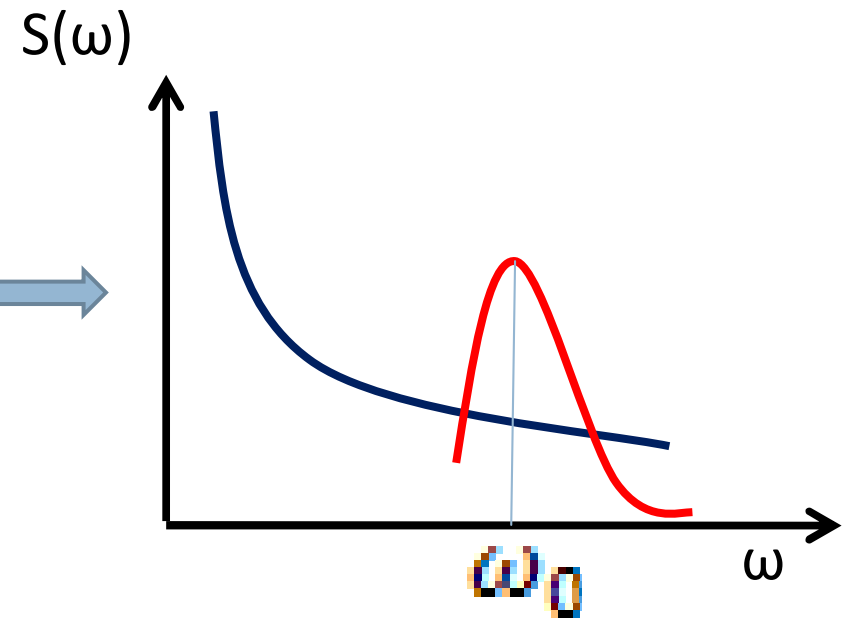
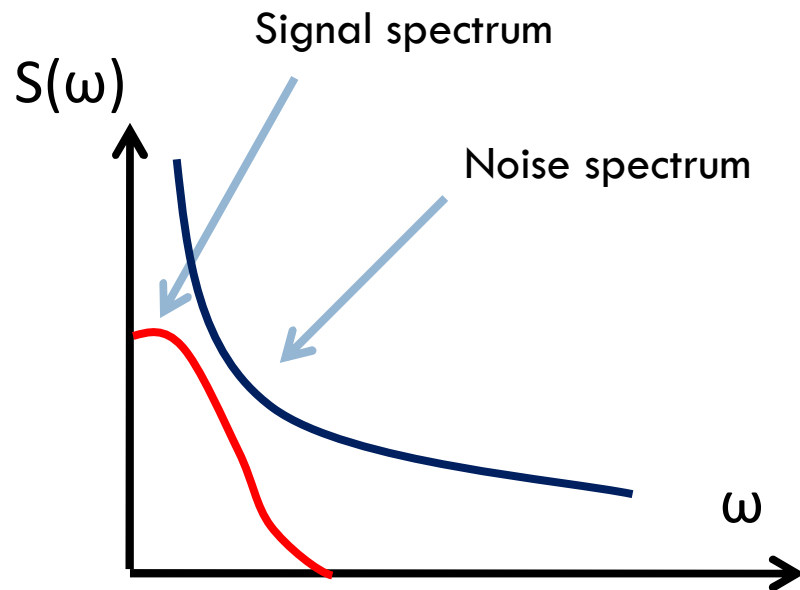
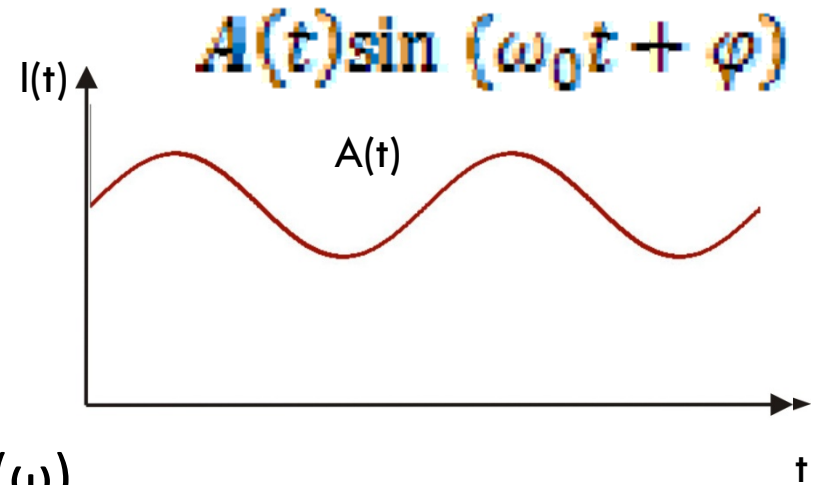
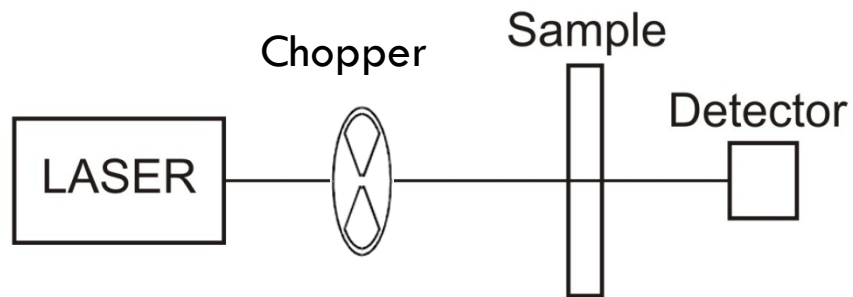
Yuri Rumyantsev
yarumyantsev@gmail.com

Lock-In Amplifier

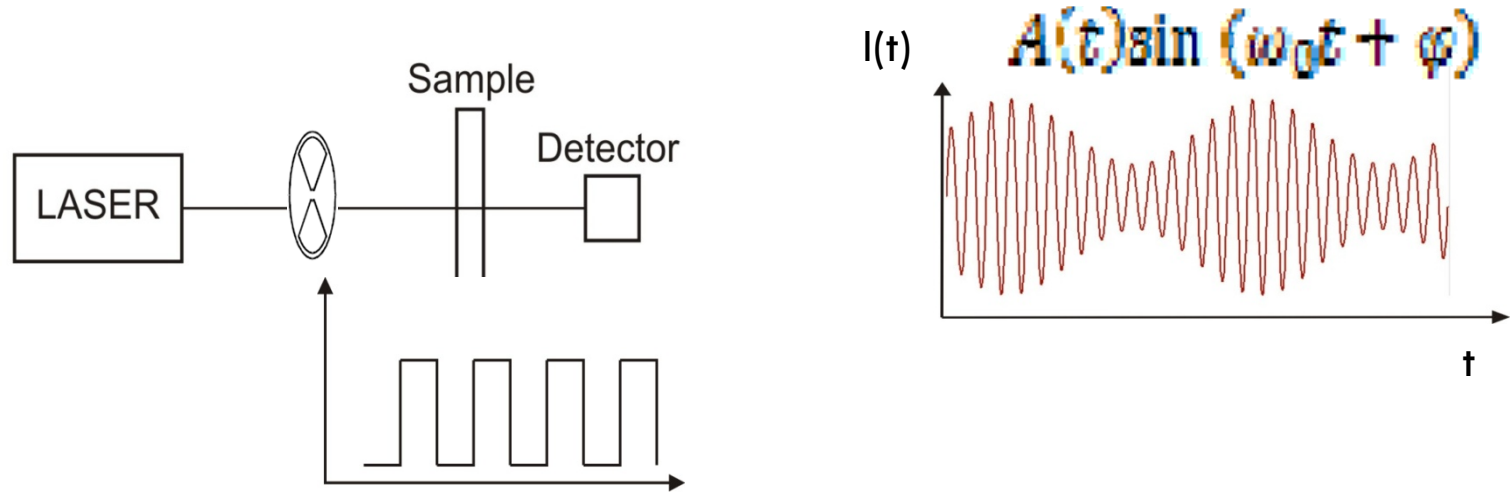


- What is it?
- Implementation

Lock-In Amplifier



Lock-In Amplifier



$$A(t)\sin(\omega t + \varphi) * \sin(\omega t) = \frac{A(t)}{2} [\cos \varphi - \cos(2\omega t + \varphi)]$$

$$A(t)\sin(\omega t + \varphi) * \cos(\omega t) = \frac{A(t)}{2} [\sin \varphi + \sin(2\omega t + \varphi)]$$

Lock-In Amplifier

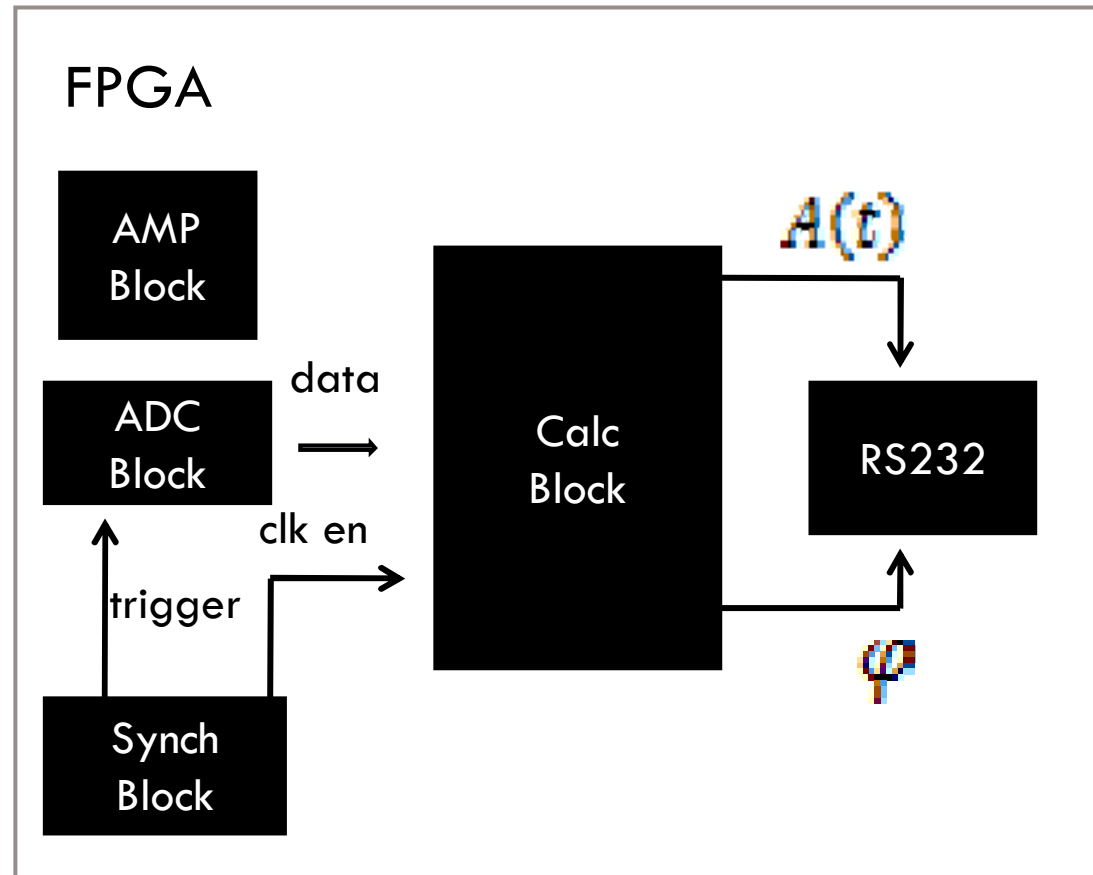
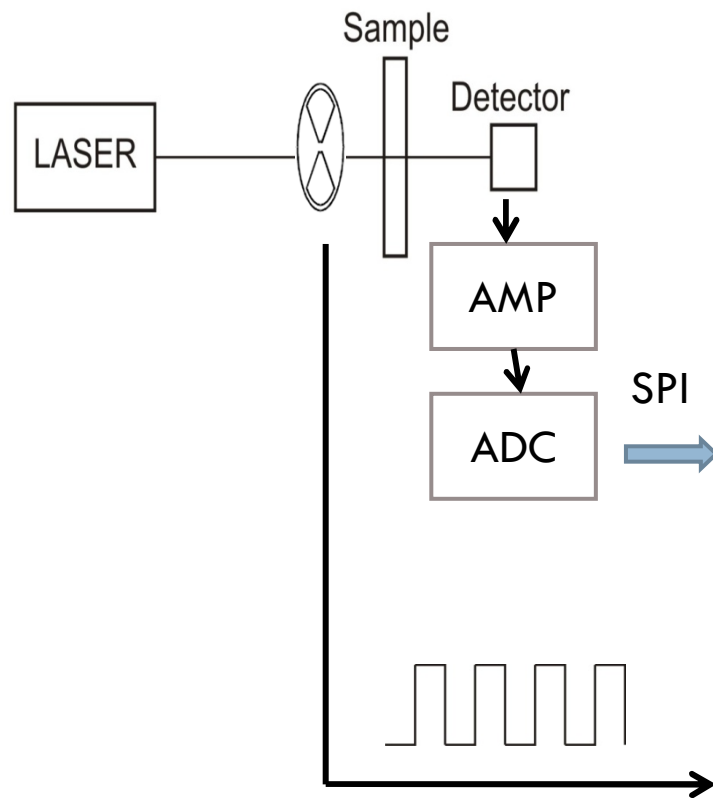
$$\langle A(t) \sin(\omega t + \varphi) * \sin(\omega t) \rangle = \frac{A(t)}{2} \cos \varphi = \alpha$$

$$\langle A(t) \sin(\omega t + \varphi) * \cos(\omega t) \rangle = \frac{A(t)}{2} \sin \varphi = \beta$$

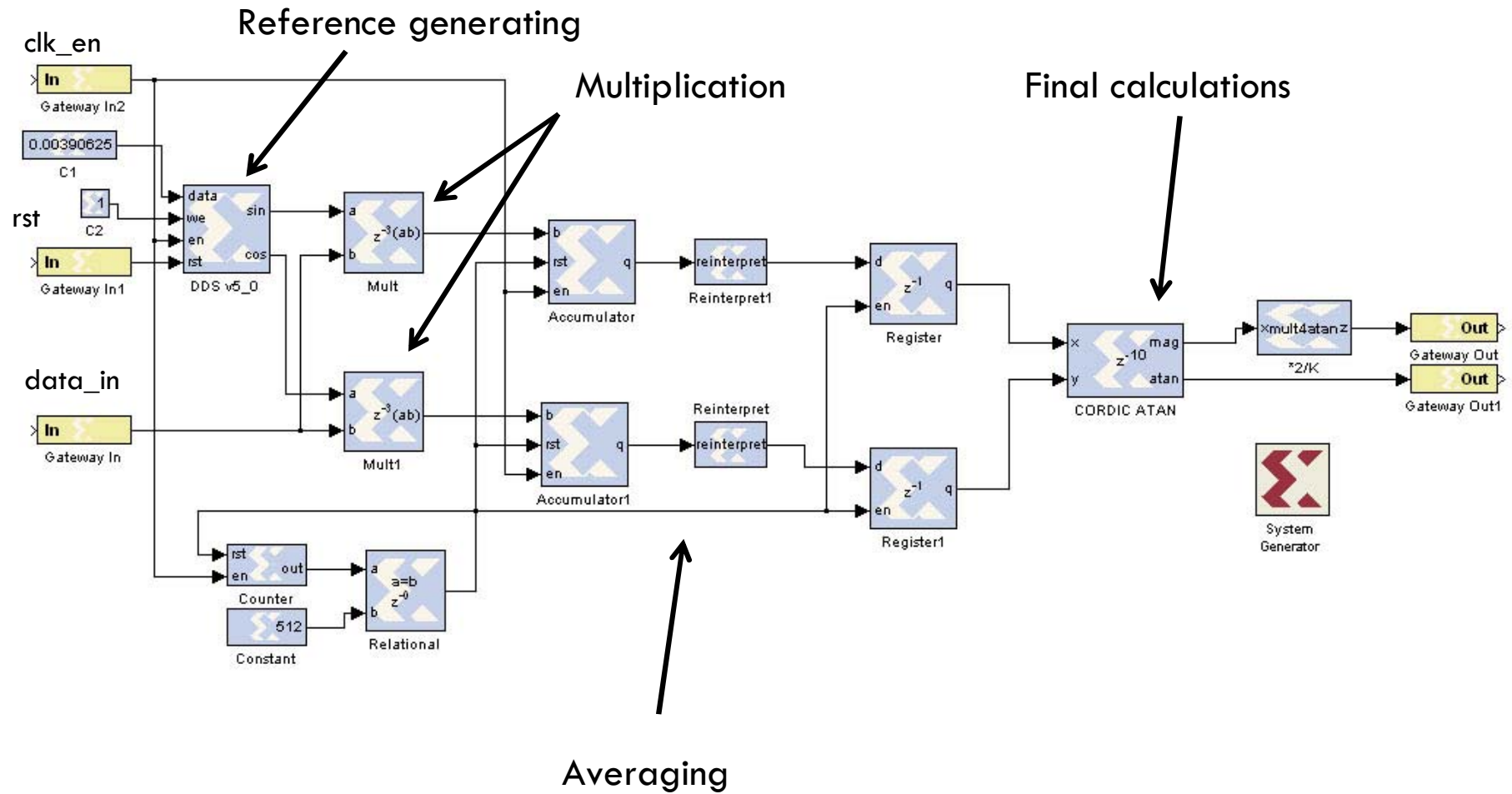
$$A(t) = 2 \sqrt{\alpha^2 + \beta^2}$$

$$\varphi = \operatorname{arctg} \left(\frac{\beta}{\alpha} \right)$$

Lock-In Amplifier: Implementation



Lock-In Amplifier





FPGA-based Single-Chip GPS Navigation System using SDR and SOC

By Ahmad Sadiq

Nigeria Communications Satellite Limited

– Research and Development

Department





Introduction

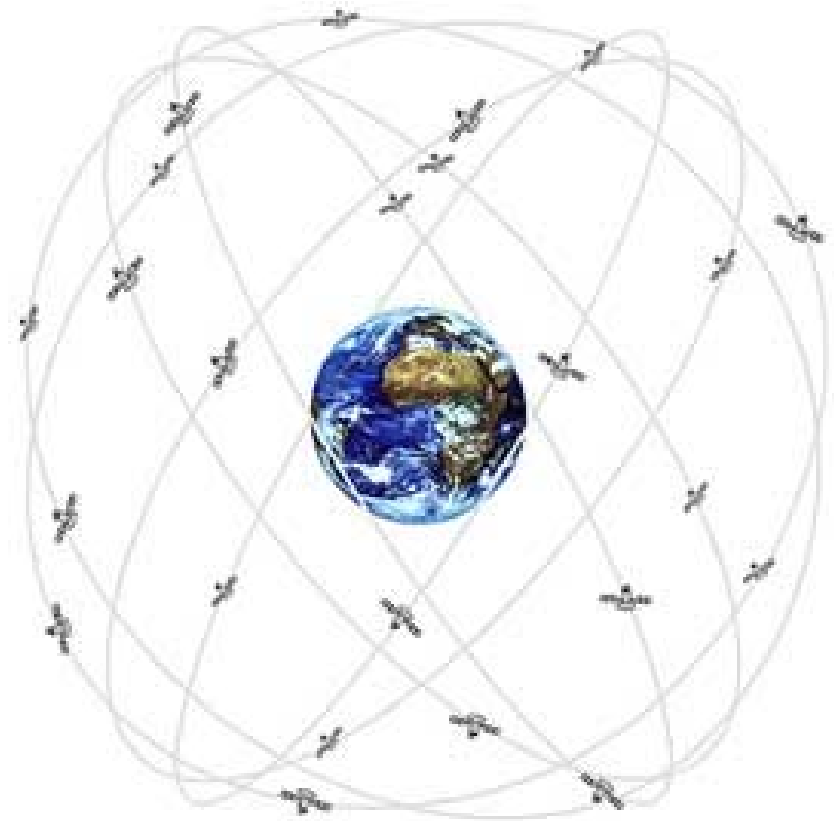
- The aim of this presentation is to propose an FPGA-based design of GPS Navigation system
- This design proposes a single chip solution to handle all the GPS receiver functions as well as a complete SOC design with LCD display interface capable of running the Map software

Table of Content

- Introduction
- About GPS
- A Look at Existing Systems
- Introducing the base technologies for this design: FPGA, SDR, SOC
- Preliminary (High-level) Design
- Design Considerations
- Summary
- References



ABOUT GPS



What is GPS

- GPS – global positioning system
- A system of 30 active satellites orbiting the earth
- The satellites transmits signals which enable the exact location of a GPS receiver on the surface of the earth to be determined (or the atmosphere; up to LEO)

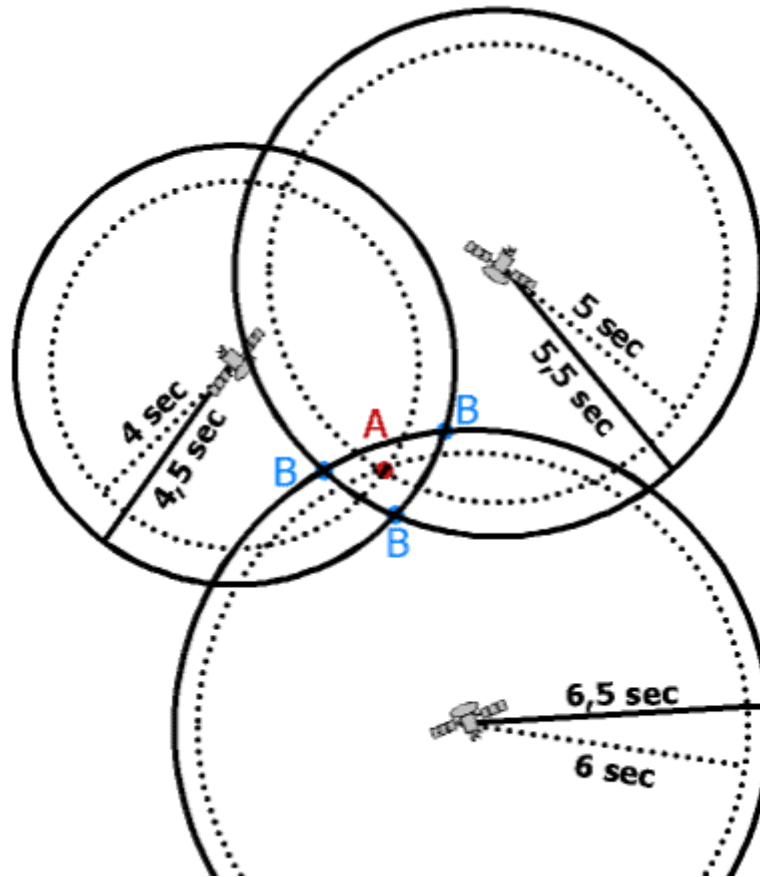
The setup of GPS system

- The GPS system can be divided into three basic segments:
- Space segment (satellites)
- Control segment (control stations)
- User segment (GPS receiver)

The GPS Receiver

- In order for a GPS receiver to work it must perform 4 tasks:
 1. Find GPS signals (frequency, code phase)
 2. Track/Demodulate the message from each GPS satellite (at the same time)
 3. Calculate the position based on distances to the satellites
 4. Calculate the correction to your local clock

Trilateration

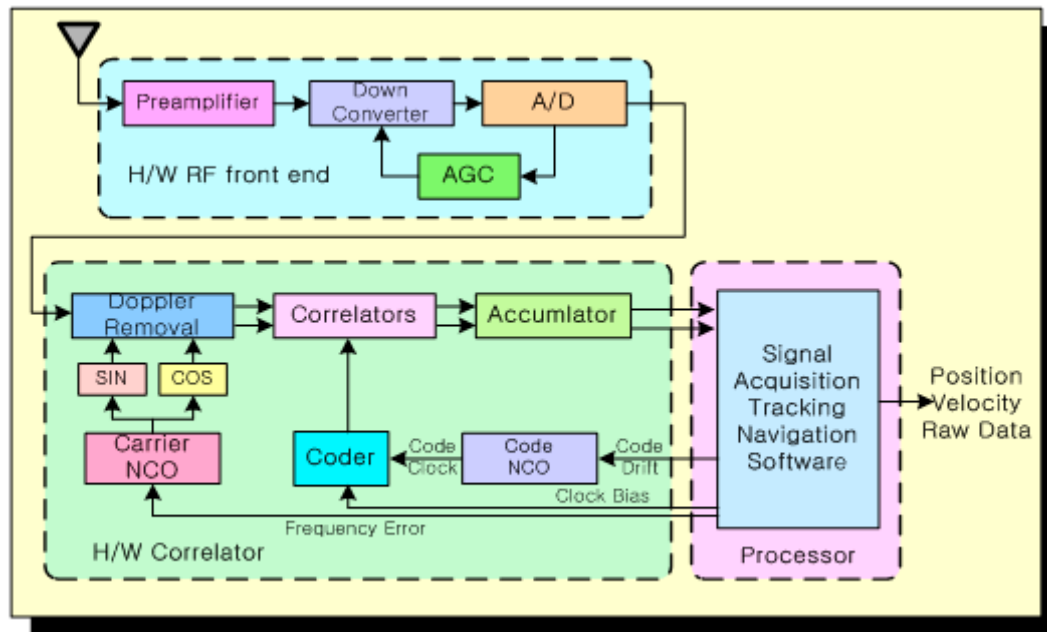




A LOOK AT EXISTING SYSTEMS

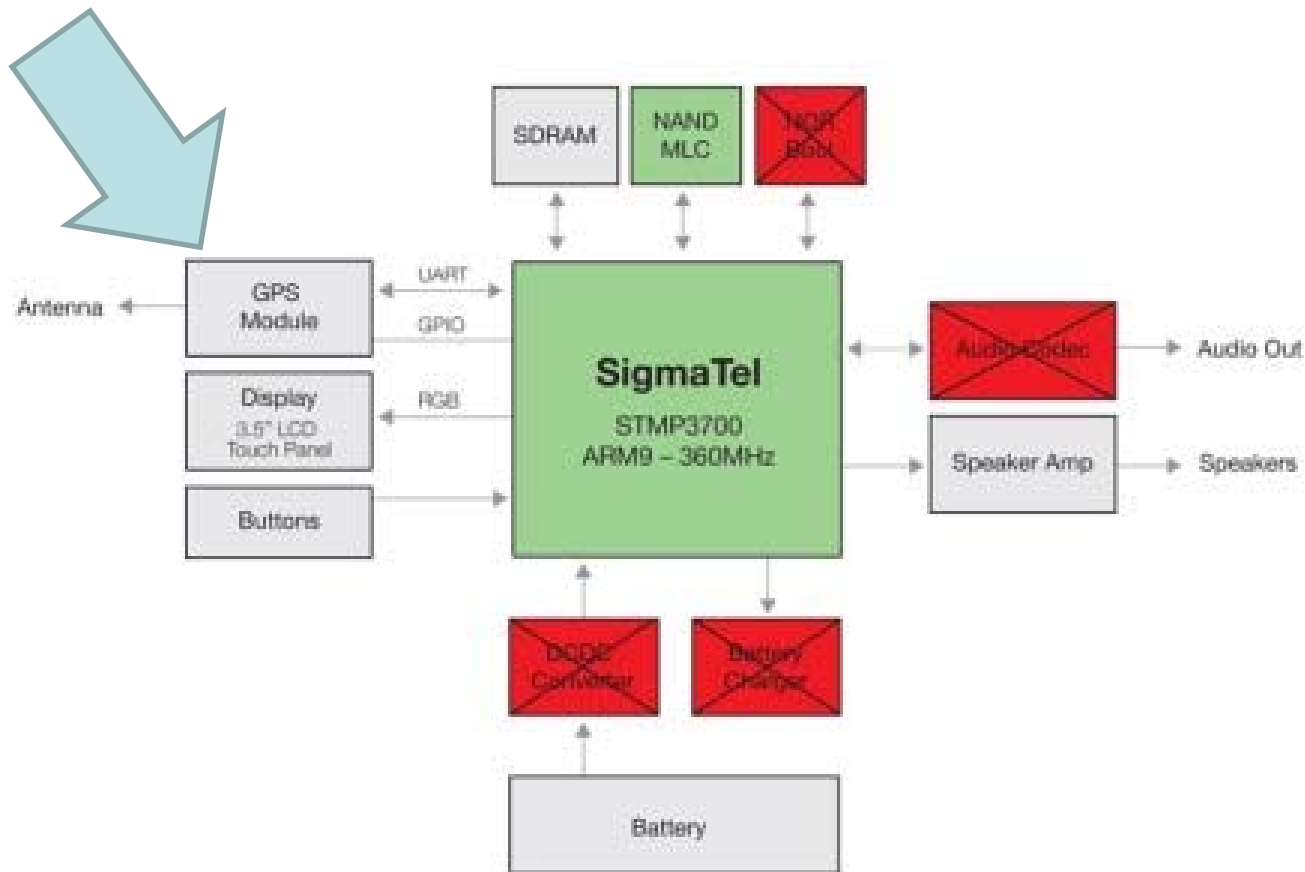
Typical Navigation System

- Existing systems typically uses at least two chips the GPS chip and a SOC for user applications e.g. the Map software.



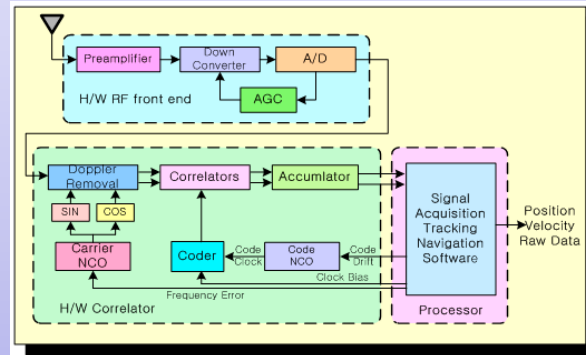
SOC

Typical Navigation System



In contrast: our Proposed System

FPGA



SOC

A NEW GLOBAL DISCOVERY



BASE TECHNOLOGIES





FPGA

- Field Programmable Gate Array is the most advanced programmable logic device
- Called the Poor man's ASIC because of the absence of NRE cost
- Advantages over ASIC includes reconfigurability, i.e. the device hardware can be reconfigured to implement new functionality without requiring any manufacturing process

FPGAs: Hottest new features

- Processor cores inside the chip with computation clocks up to 500 MHz and above, and
- lower core voltages to keep power and heat down
- Over 1000 dedicated hardware multipliers
- Memory densities approaching 15 million bits
- Silicon geometries to 40 nanometres
- Configurable logic and I/O interface standards



- **Software-Defined Radio (SDR)** is a radio communication system where components that have typically been implemented in hardware (e.g. mixers, filters, modulators/demodulators, detectors. etc.) are instead defined using software on a personal computer and then implemented on an FPGA.

Software Defined GPS receiver

- A GPS receiver can be implemented as a software defined radio
- Students at the National Taipei university of Technology implemented a simplified four-channel GPS Digital Satellite Signal (GPS-DSS) baseband system including design of C/A code generator, navigation message processing and BPSK baseband modulation, utilizing only 10% resource of Xilinx Xc2s50e FPGA.

Software Defined GPS receiver

- GPS receiver development based on SDR has advantages over conventional GPS receiver because the software GPS receiver can be upgraded to process future satellite signals such as Galileo, GPS L5 and Glonass without extra hardware.

The text 'SOC' is displayed in a large, white, sans-serif font, centered over a blue-toned background featuring a world map and a 3D rendering of a satellite or space station.

- With the increase of system resources available on latest generation FPGA, the System-on-Chip paradigm can be borrowed from classical silicon implementations into reconfigurable environments.
- It is now practical to ship volume embedded systems in a single FPGA
- the FPGA implements all of the system logic including a processor core

A NEW GLOBAL DISCOVERY



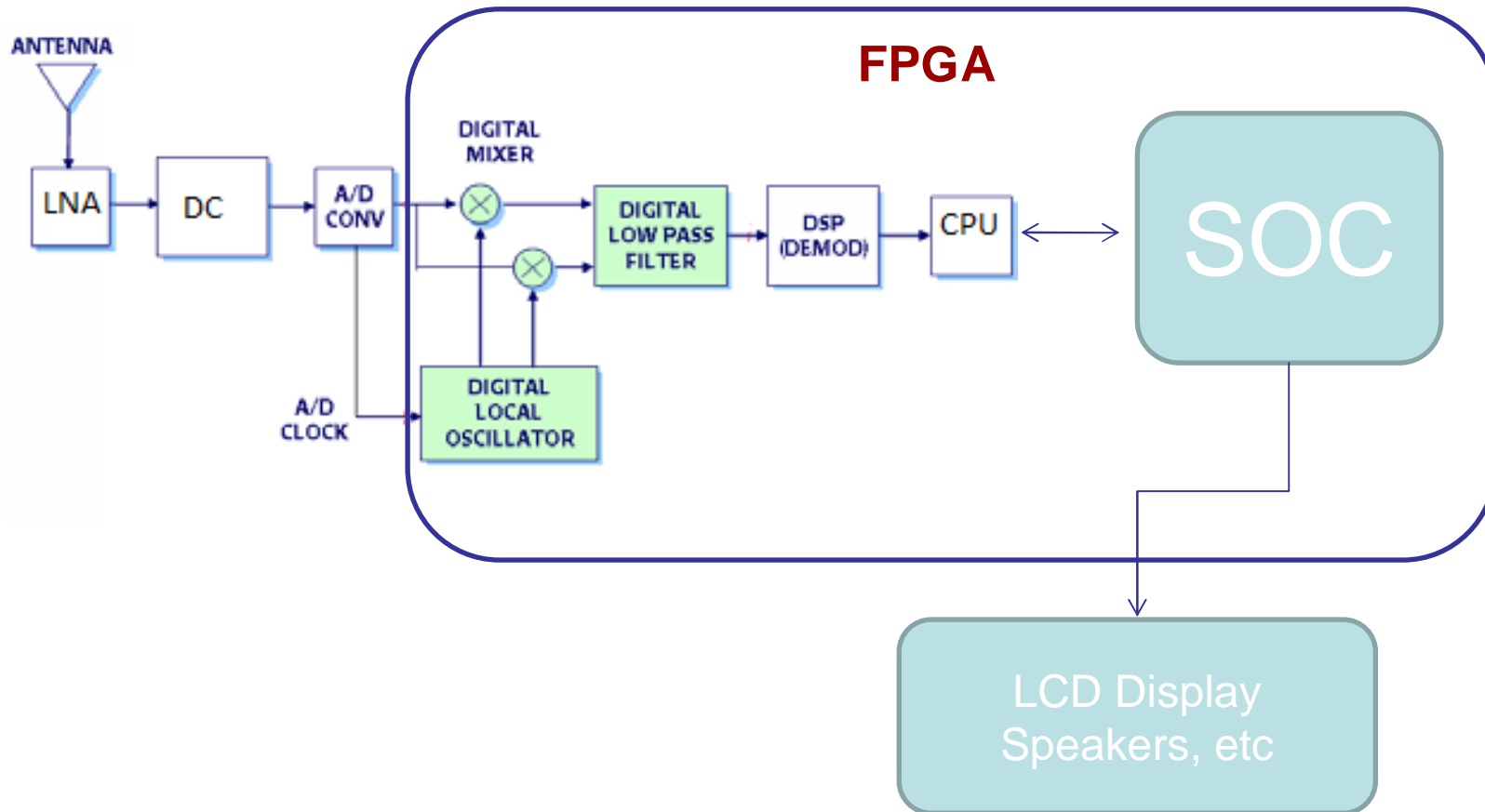
PRELIMINARY DESIGN



Preliminary Requirement

- 12 channel GPS receiver function
- Embedded System (SOC) with LCD Controller IP
- GUI Library including Map/Navigation Functions

Block Diagram



A NEW GLOBAL DISCOVERY



DESIGN CONSIDERATIONS





Challenges

- Because the received GPS signals are at such low levels this presents some challenges. One of the main considerations is to avoid contamination of the incoming signals with interference that can be generated from the digital electronics when using an FPGA.

Fine tuning the FPGA to minimize interference

- Reduction of the slew rate of I/O pins.
- using the floor plan editor to optimise the FPGA routing after design was compiled and fitted into the device.
- For the clock: using a Low Voltage Differential Signalling (LVDS) (sine wave)⁴



PCB Layout

- Copper Flooding
- Component placement to give maximum isolation for the RF components
- RF Shielding
- Bypass Capacitor and FPGA grounding



SUMMARY



Summary

- We believe that current FPGs technology is able to handle the design presented here
- The high-level integration offered by this design is the trend of the future as FPGAs are becoming more widely used in communications and mobile applications
- Because FPGA-based hardware design is reconfigurable, future technologies like L5 or Galileo can be incorporated easily
- Receiving algorithm can be developed for special purposes like military applications



REFERENCES



References

1. Sung-Chun Bu, et al (December 2004), **A FPGA-based software GPS receiver design using Simulink**
2. Trong-Yen Lee, et al, **A Low-Cost GPS Satellite Signal Baseband System Using FPGA Prototyping**
3. Zedong Nie, Kangling Fang, Xu Xin (2006) **A Portable Positioning System Based on SOPC Technology**
4. Parkinson et al. (2006): **FPGA based GPS receiver design considerations**
5. Jun Xu et al., **Implementation of FPGA-Based Acquisition of Weak GPS Signals**

Government OF HIGH EDUCATION

REPUBLIC OF CAMEROON

DSCHANG UNIVERSITY

FACULTY OF SCIENCES

DEPARTEMENT OF PHYSICS



LAPHAPEP
LAPHAPEP

DESIGN AND SIMULATION OF AN ULTRA MODERN
SECURISATION SYSTEM TELLY FOLLOWED OF A PUBLIC
BUILDING :
CASE OF THE DSCHANG UNIVERSITY 'S RECTORATE

BY :

TCHAPGA TCHITO CHRISTIAN

In intention to obtain a MASTER IN PHYSICS

Option : ELECTRONICS

UNDER THE SUPERVISION OF :

PR. ANACLET FOMETHE

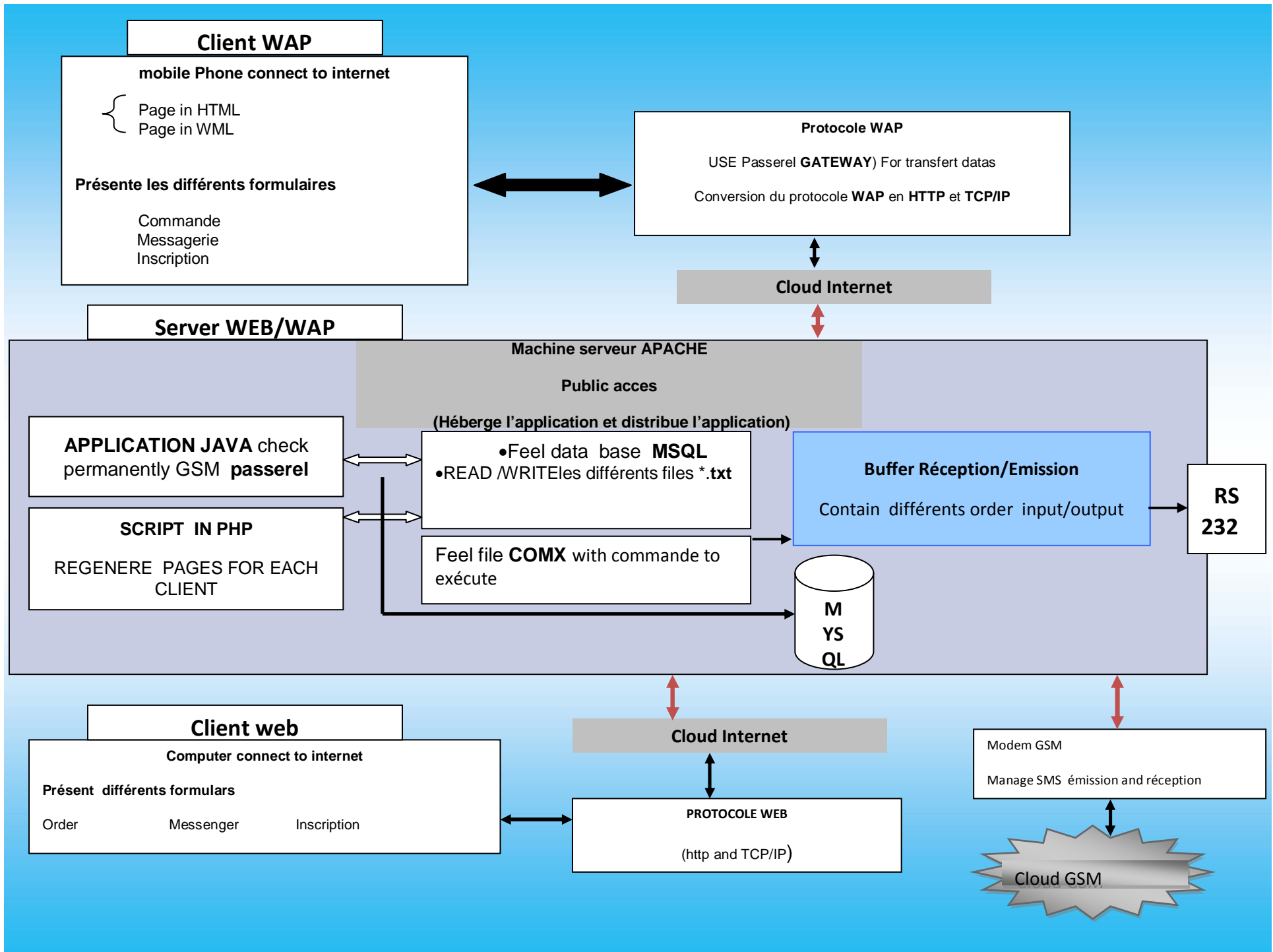
UNDER THE SUPERVISION OF :

DR. ROBERT TCHITNGA

ACADEMIC YEAR : 2008/2009

ABSTRACT

- ***We have designed and implemented for our master thesis in physics, a high-performance security system manager for a public building. Case of the of Dschang University 's Rectorate.***
- ***This system will enable control of electrical equipments , from a position of mobile phone via SMS, from a position compatible mobile phone via the internet, a computer connected to the Internet, from a any position in the local network, from the machine on which it is connected, the status of all connected devices that form the safety and comfort of the cockpit..***



PART 1 : USED LANGUAGES AND LOGIC CONSTITUTION

CONCEPT OF CLIENT-SERVER TECHNOLOGY

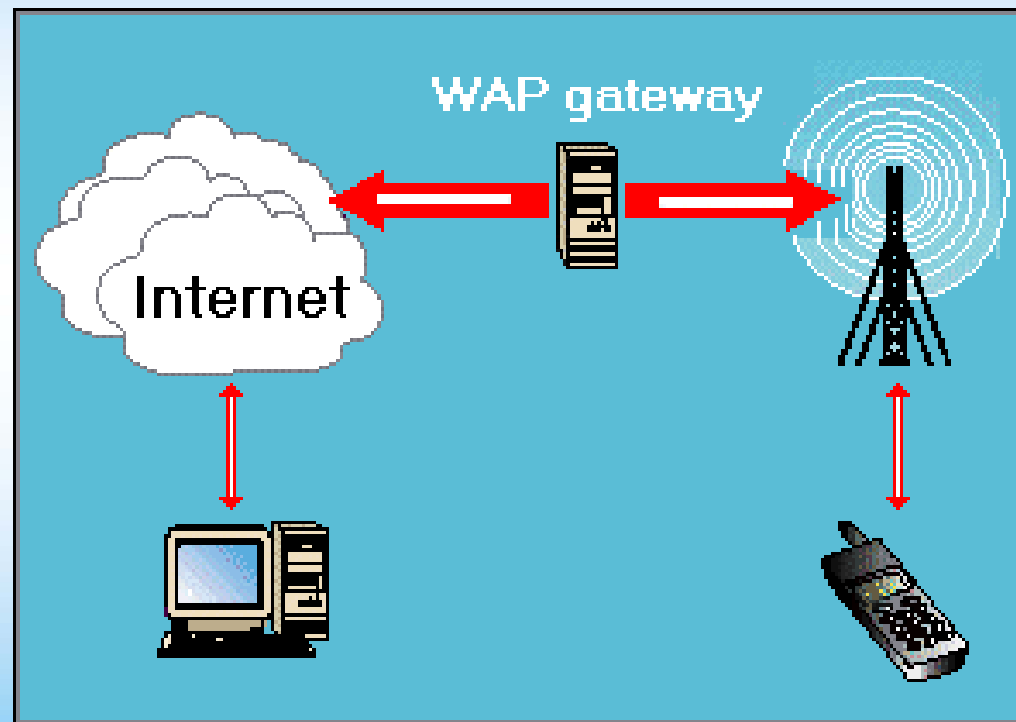
- II.1 client
- II.2 server
- III The PHP language

THE SOFTWARE EasyPHP

THE LANGUAGE XHTML AND CSS



THE WML LANGUAGE

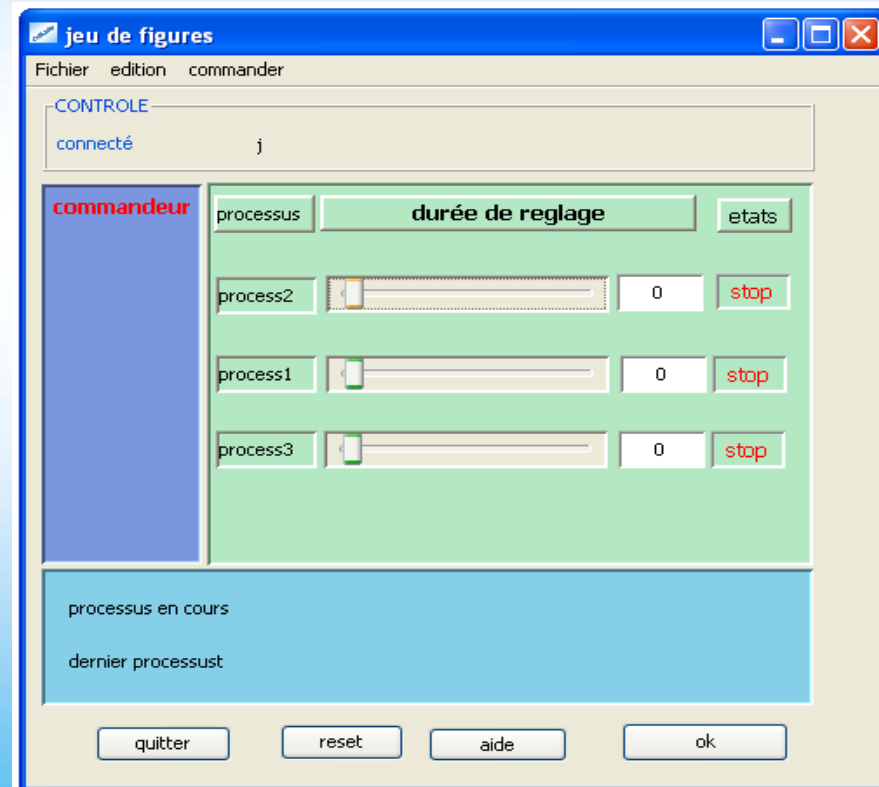


THE GSM



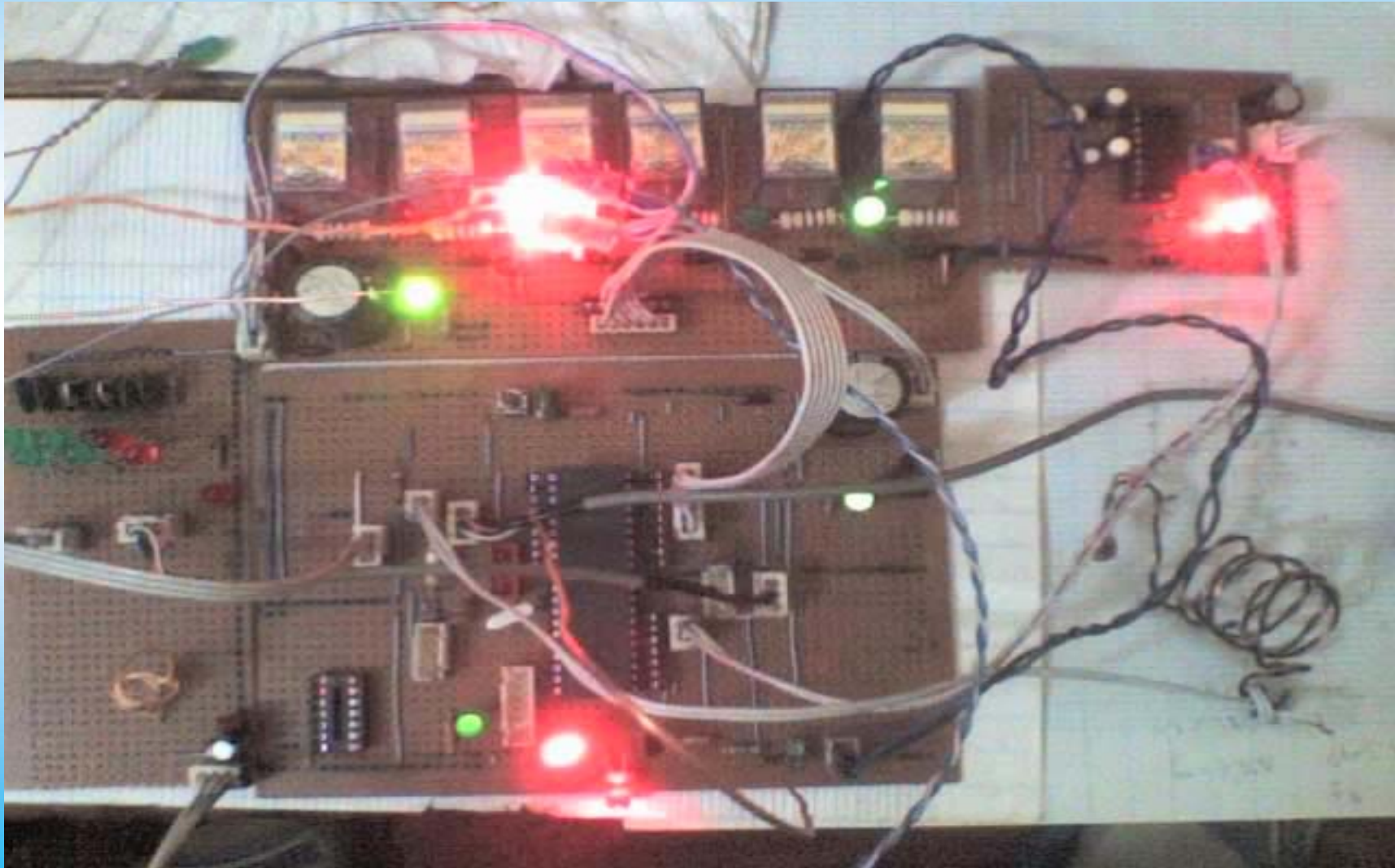
Présenté par TCHAPGA TCHITO
CHRISTIAN

THE JAVA LANGUAGE

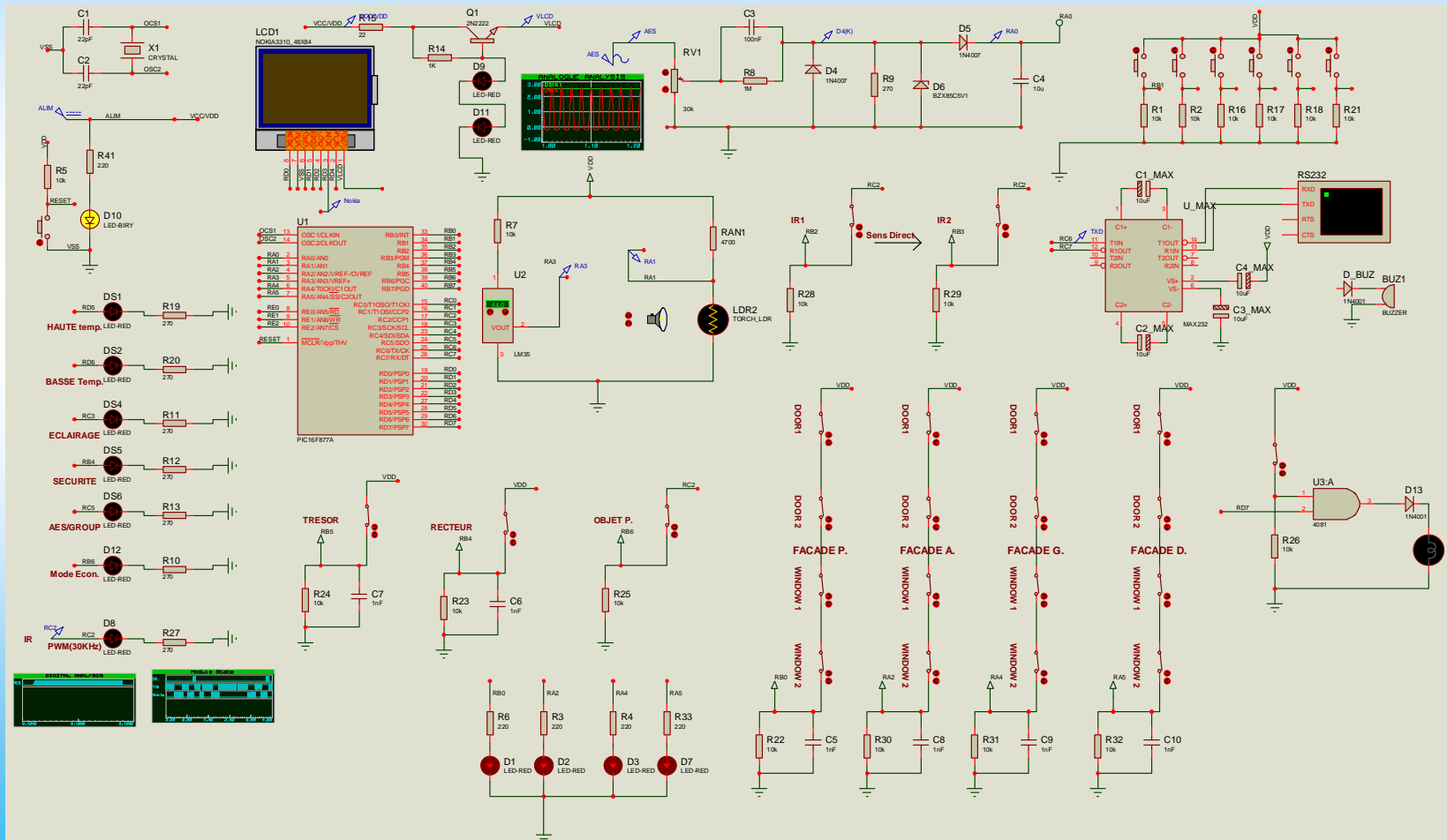


Présenté par TCHAPGA TCHITO
CHRISTIAN

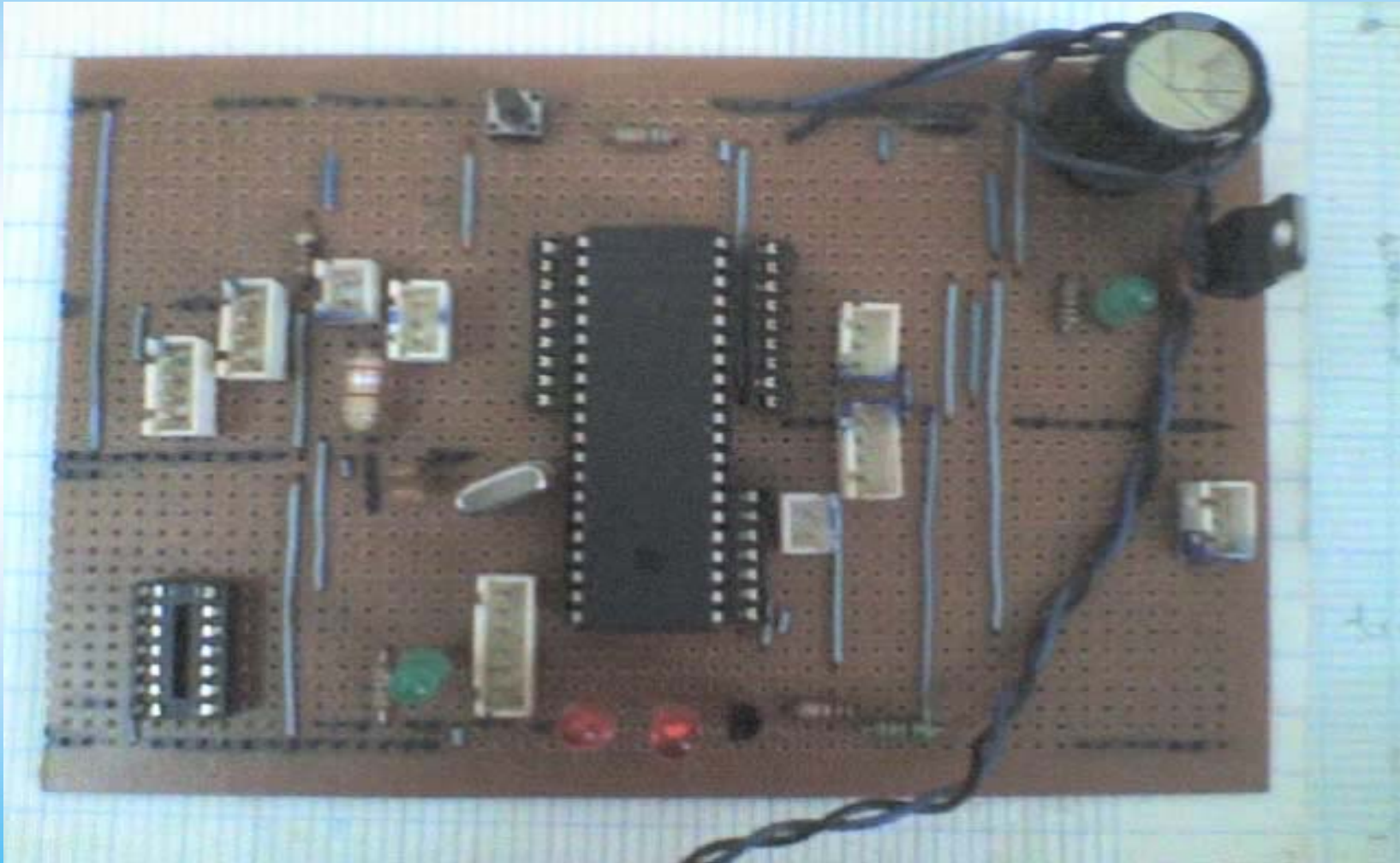
PART 2 : ELECTRONIC CONSTITUTION



ELECTRONIC CONTROL SYSTEM IMPLEMENTATION



WHY MICROCONTROLLER ?



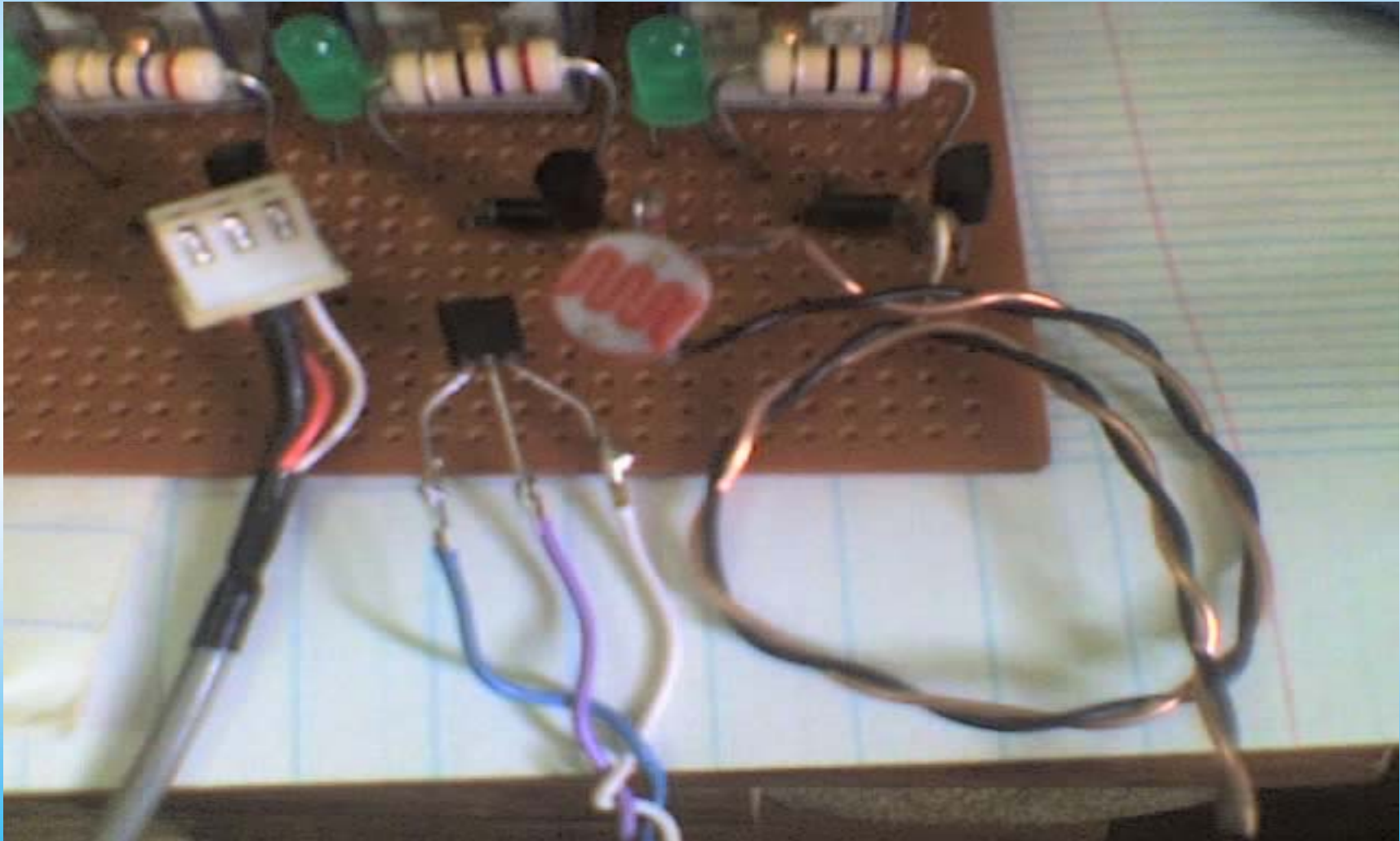
CCS

```
135 // #define PIN_E1
136 #define lampEco PIN_E2
137
138
139 //***** Déclaration des fonctions *****/
140 void initialisation();
141 void controlDesTensions();
142 float valeurADC(int channel);
143 void lireLesEntreeAnal();
144 void compteurPersonnes();
145 void commandIR();
146 void lectureSecteur();
147 void controlTemperature();
148 void messageNokia(char *message);
149 void controlEclairage();
150 void controlLumSolaire();
151 void affichage();
152 void controlIR();
153 void distantCommand();
154 void controlDoorWindows();
155 void alarmControl(int1 alarmOff); //gestion des proessus sécurisés
156 void delay_S(int16 sec); //temporisation en secondes
157 void moveToBank(int1 RP1, int1 RPO); //aller en banque numéro
158 int16 nombreDePersonnes(); //compte le nombre de personnes au rectorat
159 void emissionIR(); //émission des IR et mémorisation des états
160 void recapitulativ(); //renvoi toutes les information par RS232
161
162
163
164
165
166 void initialisation();
```

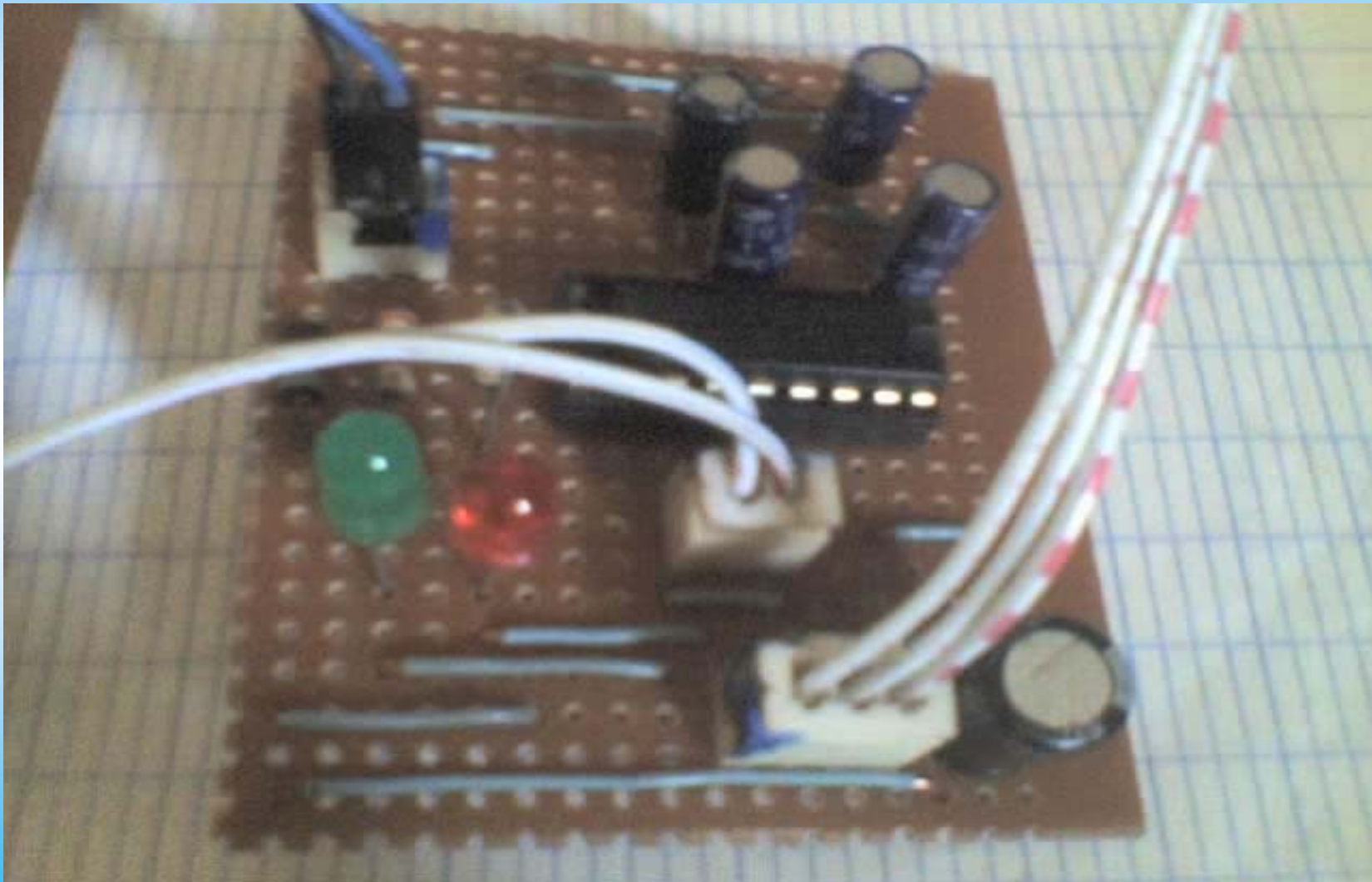
PIC PROGRAMMATOR



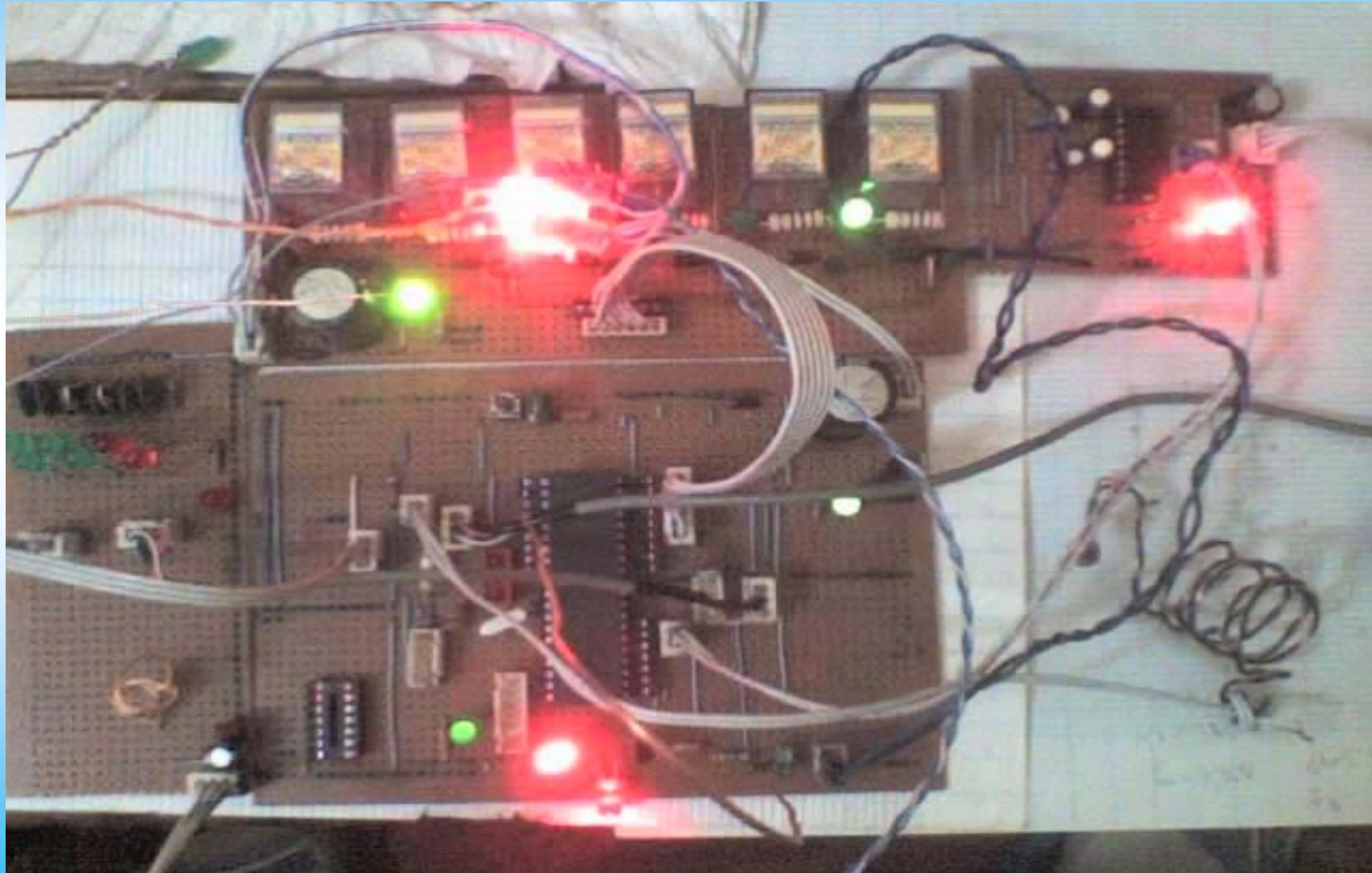
ANALOGIC DIGITAL CONVERTER



RS232 COMMUNICATION



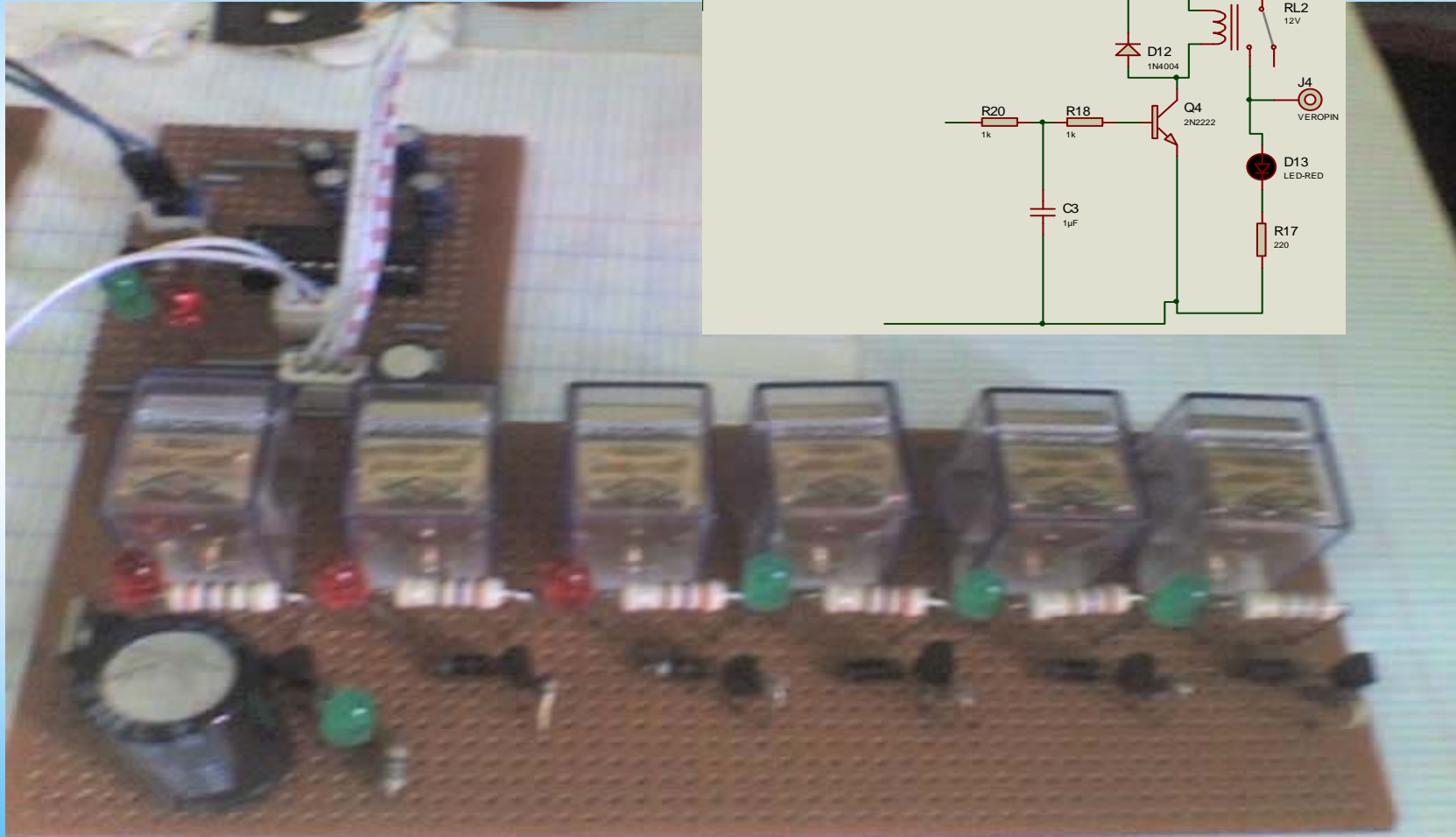
INFRA-RED MODULE



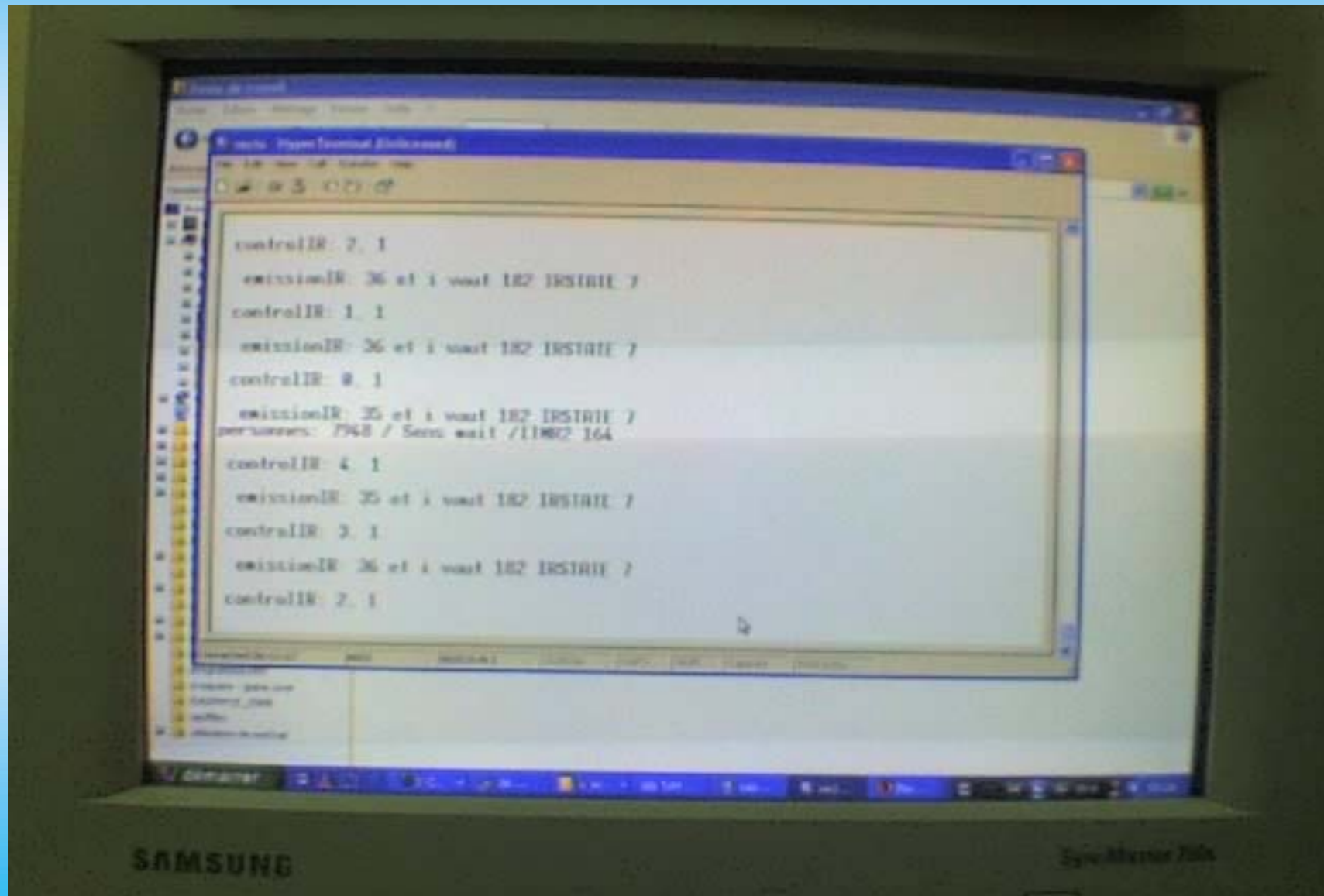
DOORS AND WINDOWS CONTROL

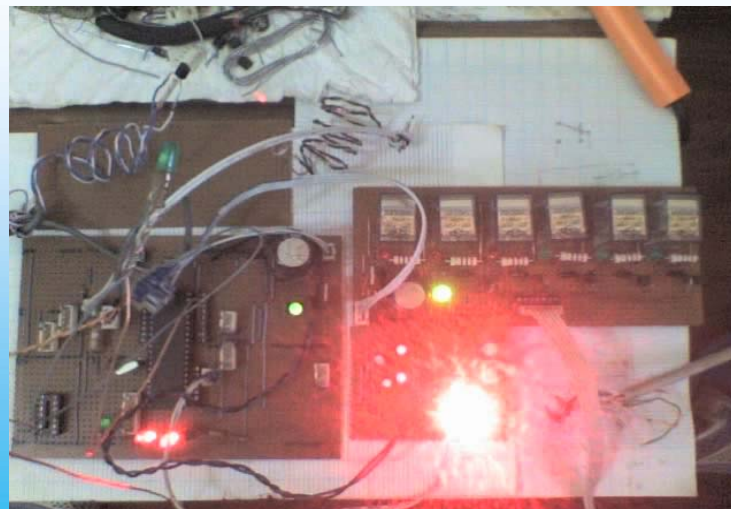
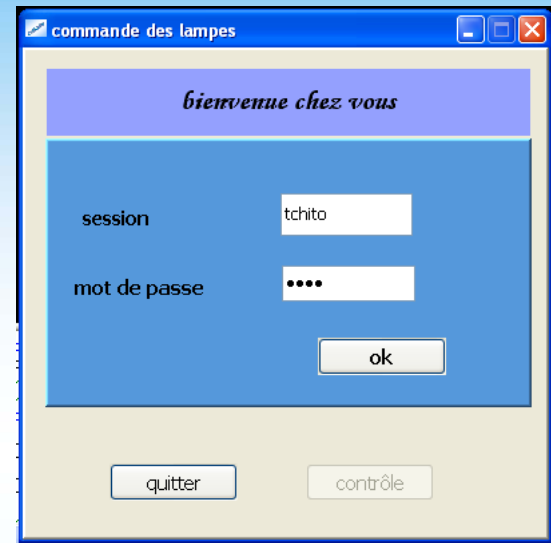
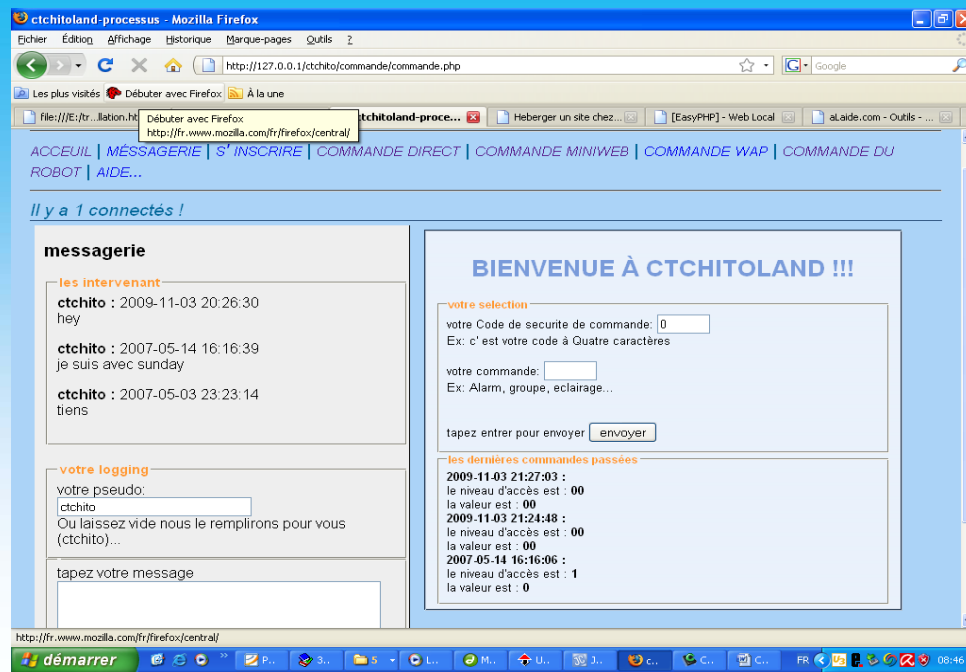


POWER INTERFACE



SOME RESULTS





COST ESTIMATE

- **Electrical equipement : 90 250 XAF**
- **Computer devices : 1 050 000 XAF**

Total: 1 140 250 XAF

1Euro = 655.94 XAF

SCOPE OF PROJECT AND PROSPECTS FOR IMPROVEMENT

- ***Change sequential execution to data flow organisation***
- ***Find a way to reduce components around command unit***
- ***Increase I/O number (digital;analog...)***

END

***TANK YOU FOR YOUR KINDLY
ATTENTION***