A FPGA-based Implementation of a Fault-Tolerant Neural Architecture for Photon Identification

M. Alderighi[1], E.L. Gummati[2], V. Piuri[3] and G.R.Sechi[1]

[1] *Istituto di Fisica Cosmica e Tecnologie Relative, Consiglio Nazionale delle Ricerche*
Via Bassini 15, I-20133 Milano, Italy, Tel: +39-2-23699332 Fax:+39-2-2362946 Email: {monica, giacomo}@ifctr.mi.cnr.it
[2] *Dipartimento di Fisica, Universita' degli Studi di Milano*
Via Celoria 21, I-20133 Milano, Italy Email: lau1@ifctr.mi.cnr.it
[3] *Dipartimento di Elettronica e Informazione, Politecnico di Milano*
P.za L. Da Vinci 32, I-20133 Milano, Italy, Tel: +39-2-23993606 Fax: +39-2-2399 3411 Email: piuri@elet.polimi.it

Abstract

Event identification in photon counting ICCD detectors requires a high level image analysis which cannot be easily described algorithmically: neural networks are promising to approach this application. A system capable of identifying these events on board of satellites needs fault tolerant capabilities to certify result correctness. The rapid evolution of the problem specification, due to the increasing knowledge about the physics, makes attractive the availability and modifiability of prototypes: FPGA-based design is effective to realize these systems. The paper presents therefore an FPGA-based implementation of a fault tolerant neural architecture for event identification.

1. Introduction

Photon counting intensified CCDs have been shown a viable choice for imaging and spectroscopy in selected spectral bands from the extreme to the near ultraviolet and, thus, are currently used in several research for space physics [15]. Due to the massive amount of data to be treated and the strict temporal requirements, processing on board of the satellites often becomes mandatory. As a consequence, massively parallel processing paradigms and architectures need to be adopted. On the other hand, due to the incomplete knowledge about the operation to be performed, it is quite difficult to specify completely and exhaustively an algorithm to analyse the CCD images and to extract the desired features. In the literature, the neural networks have been shown an effective and efficient approach to image analysis due to their intrinsic computational parallelism and their configurability through learning by examples [8].

The use of digital systems in critical applications, such as in the space area, requires error tolerance capabilities in order to guarantee the correctness and the reliability of the results and, as a consequence in our specific application case, of the experimental data, analysis and theoretical consequences from the experiment. Neural networks are often claimed to have intrinsic fault tolerance capabilities. Unfortunately, only a limited amount of intrinsic correction capacity is available [6, 9] for some very specific classes of faults, for some neural paradigms, and only if information and computation have been enforced to be widely distributed in the whole neural network via a suited learning procedure. However, this capability does not provide any information both on the correctness of the current network's output and on the location of the faulty component. Intrinsic error masking may fail for some input sets and from a specific time on, but signal is generated to prevent the use of the erroneous data generated by the network in critical applications. On the other hand, no assumption can be a priori defined on the error magnitude with respect to the normal magnitude of the network results, so that any error can be extremely critical for the application. This implies that we need an architectural support to introduce error detection in these computing structures. When throughput is a mandatory requirement in massive computing applications, suited hardware and support techniques must be adopted to achieve this goal concurrently with the nominal operations, without degrading the system performance in a relevant way with respect to the non-fault tolerant system.

The use of computing systems for experiments in physics makes relevant to adopt dedicated architectures for rapid prototyping in order to achieve a fast 'time to market' and to validate the theoretical analysis of the experiment. In many application cases related to the research in advanced physics, the availability of hardware prototypes and their fast development become critical due to possible uncertainty in the solution and the possible incomplete knowledge of the experiments themselves. When experiments give new information about the physical behaviours of the system or phenomenon under study, the computing system needs to be adapted to capture new information or to behave correctly according to the theoretical interpretation of the phenomenon. The use of FPGAs is an effective solution to implement flexible, modifiable and modular prototypes, even though they may be expensive in terms of system complexity; a similar example was presented in [7] for a simple feed-forward network classifier, while we need a more complex and fast structure modeling a dynamic system for our application case. More efficient implementations could be realized by using ASIC technologies, but flexibility and modifiability would be greatly reduced.

In this paper, we present an application of the FPGA technologies for prototyping in the space area: the design and the implementation of an FPGA-based fault tolerant neural architecture for event identification in photon counting ICCD detectors [9, 10, 15]. Section 2 briefly describes the event identification problem; neural network characteristics are sketched in Section 3, while the system architecture and neuron implementation are illustrated in Section 4. The neuron implementation with added concurrent error detection capabilities is given in Section 5.

2. The photon counting intensified CCDs

Photon counting intensified CCDs have been shown a viable choice for imaging and spectroscopy in selected spectral bands from the extreme to the near ultraviolet [15]. The detector system consists of a high gain electron multiplier, based on MicroChannel Plates (MCP), a readout system based on a phosphor screen fibre optically coupled to a fast-scanned CCD camera, and of a signal processing unit for event identification and centroiding, to localize events to sub pixel accuracy.

Incident photons impinge on a photocathode material deposited directly onto the front MCP face causing the emission of photoelectrons into the MCP channels (or back outwards and reflected forwards into the MCP by a biased grid). Each channel constitutes an independent, continuous dynode photomultiplier that yields an electron cloud at the MCP output face. The electron cloud exiting the MCP output face is accelerated, across a proximity gap, onto a phosphor screen deposited onto a fiber optic (FO) faceplate. Photons emitted are channelled out of the screen by the fibers and, through a FO coupler with appropriate reduction ratio to match the matrix format, are directed onto a CCD that provides an image of the signals detected. On the CCD matrix (512x512 8-bit pixels), each photon event is represented by a charge distribution of approximately Gaussian profile, covering a 5x5 pixel area.

The processing unit is to recognize valid photon events by a morphological analysis of the whole CCD frame. Upon identification, the centroids of each and every individual event detected are then determined to sub-pixel accuracy and the resulting image is accumulated at higher resolution. Only those 5x5 CCD areas having the requested energy and Gaussian distribution of pixels are to be recognized as good events. All the others, spurious or noisy events, are to be rejected.

The time required for identification is the key factor in determining the ultimate speed of the system because every pixel is to be examined, while successive steps are to performed only on accepted events. The frame rate of the camera presently used is 60 frame/sec. Acquisition and analysis of each pixel, at 20 MHz readout frequency, takes 50ns.

In order to meet the strict temporal deadlines and possibly cover even higher readout speed and/or larger CCD format, we designed an *ad hoc* computational system to be interfaced with the CCD readout system. A fast and robust morphological analysis of signals cannot be easily described by using an algorithmic approach, while it is possible to create many meaningful examples of the desired system behaviour. Therefore, we designed a neural network based computing system: the SIgnal REcognition Network (SIREN) [3, 4, 5].

3. The SIREN network

SIREN is the feedback neural network designed for the event identification problem. All neurons work at a time and their state (which coincides with the output) is both communicated to the connected neurons and fed back to the neurons themselves. The network is synchronous, meaning that neurons work simultaneously on stable input data.

SIREN operates by analysing a whole CCD frame, detecting and identifying good events. Upon completion of its dynamics, only good events are preserved, while all other bad or noise events are zeroed.

SIREN topology is based on a regular scheme of interconnections of 5x5 neurons, called kernel from now on, corresponding to the event "window". Each neuron is viewed as the central element of a 5x5 neuron area and has 25 connections: 24 to each one of the neighbouring neurons, and one to itself (the output is fed back as one of its input). This choice is due to the nature of the event to be recognized, which is expected to cover an area of 5x5 pixels. This scheme may be repeated to cover any subset of the CCD frame.

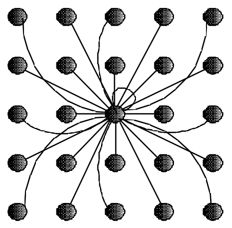

Figure 1: SIREN connections of the central neuron.

Similar connections and the same set of synaptic weights characterize all neurons in the network. This allows to analyse all the possible 5x5 windows in a frame, i.e., to identify good events independently of their position. Let

$$y_j(t+1) = \sigma_T\left[\sum_{j=1}^{25} \omega_i s_i(t) - \vartheta\right] \quad j = 1\ldots 25 \qquad (1)$$

be the output of the j-th neuron at time t+1, where $s_i(t)$ is the state of the i-th neuron at time t, w_i is the i-th input synaptic weight, s_T is the sigmoidal function, and T and ϑ are threshold and temperature respectively. The equation describing the dynamic behaviour of neurons is given by:

$$s_j(t+1) = \begin{cases} s_j(t) & \text{if} \quad \left|y_j - s_j(t)\right| < \alpha \bullet s_j(t) \quad j = 1,\ldots,25 \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

The sigmoidal function s_T is given by [3, 4, 5]:

$$\sigma_T(x_j) = \begin{cases} 0 & \text{if} \quad x_j \le 127 \\ \operatorname{int}\left[128 + x_j / T\right] & \text{if} \quad -128T \le x_j \le 127T \\ 255 & \text{if} \quad x_j \ge 128T \end{cases} \qquad x_j = \sum_{i=1}^{25} \omega_i s_i(t) - \vartheta \quad j = 1,\ldots,25$$

The rule provides a fast convergence of the dynamics and event identification can be accomplished in three cycles only. The value of a has been experimentally determined in 0.25. Higher/lower values cause a greater number of bad/good events to be accepted/rejected. The SIREN computational model is based on an integer arithmetic: four bits represent weights and eight input data are used. The rule (2) as well as the identification capabilities and some performance evaluations of SIREN are fully described in [5].

The synaptic weights are defined by applying a specific supervised learning algorithm [3], based on a gradient descent optimization technique tailored on the specific network structure. Due to the translational invariance of the identification process and the rotational symmetry of the patterns to be identified, the number of independent weights for each neuron is reduced to only 6. The training is accomplished off-line and off-board.

4. The system description

The system implementing the event identification processing consists of two main parts: a Serial Acquisiton Unit (SAU) that concurrently samples event windows and an Event Identification Unit (EIU) to identify events within the image being acquired from the CCD camera. A prototype based on an early neuron model was presented in [2].

SAU operation principle is illustrated in Figure 2. Five 512-byte shift registers, connected in series, are fed by the pixel-by-pixel CCD readout system; at any given clock time (but the first four ones), the contents of their five end registers map a 5x5 pixel window of the CCD frame. As time evolves, the window covers dynamically the whole CCD image being acquired. As a pixel content enters the input 512-byte shift register, five new pixels (a_5 to e_5) enter the window from the left-hand side, while the five pixels (a_0 to e_0) exit from the right-hand side. This allows to analyse the whole CCD frame, avoiding problems of either fixed or dynamic image partitioning. The 5x5 pixel window is used for event identification and centroiding. Details about the SAU implementation can be found in [2].

As far as the EIU is concerned, it consists of a twenty-five neuron unit interfaced to the SAU. This means that one single kernel is used in the actual implementation, sliding on the CCD frame. EIU are implemented by using the Xilinx XC4013 FPGA. EIU consists of five boards for the neural network and one board for the microprogrammable control unit. The neural network is realized by twenty-five XC4013 (five for each board).

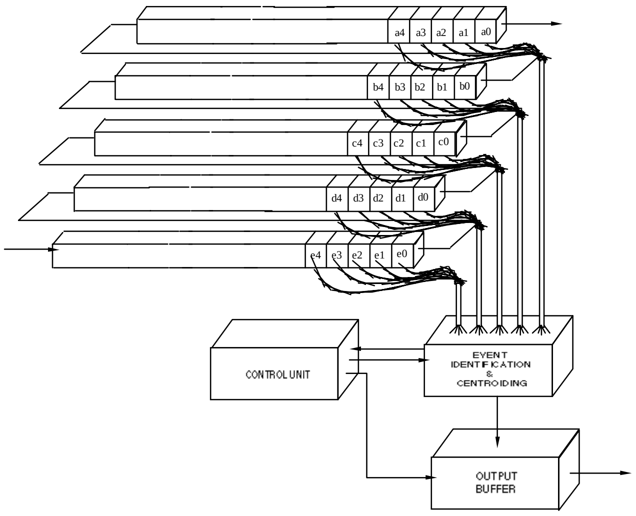

Figure 2: System Block Diagram.

The neuron structure mainly consists of two parts: the Weighted Sum Section (WSS) and the Sigmoidal Function Section (SFS). For the WSS we adopt the same architecture as presented in [1]. The logical scheme is reported in Figure 3.

Briefly**,** the weighted sum is implemented via several levels of pipelined full adders. Data are acquired serially (one bit at a time) to minimize the number of inputs and grouped into six classes according to the given six weight values. Data in each class are summed up and multiplied by their corresponding weight. Results of all classes are finally summed up by a three-level full adder. The operation proceeds as follows: let us assume that the first bits of the inputs are acquired at time *t*, at time *t+1* the pre-summation process is performed on the first bits; at time *t+2* the second level of summation is carried out and the results are stored into the pipeline registers (S1, S2, S3); at time *t+3* the content of the pipeline registers are released and the third level of summation is performed; at time *t+4* the last summation is performed and the result stored into the register S5; at time *t+5* the accumulation loop in the cumulating circuit starts by adding the result S5(0) (SumReg in Figure 4) with the content of a 16-bit accumulation register, initially empty (AccReg in Figure 4). At any time from *t+6* to *t+11*, the accumulation continues working on the subsequent bits respectively. At time *t+11* the output of the accumulation adder yields the result of the weighted sums of the first set of inputs: 25 eight-bits inputs. At this time the clear control of the accumulation register is set to high to restart the process on the next set of inputs. This section can be viewed as a stand-alone section of the neuron because it can work continuously on the input data. Except for the initial transient state, the weighted sum is performed in eight clock cycles.

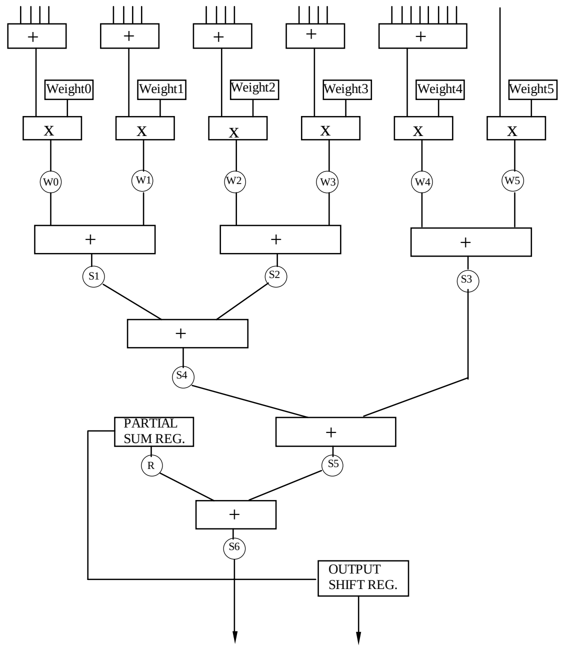

Figure 3: The weighted sum block diagram.

The sigmoidal function computation is accomplished by dividing the weighted sum, diminished by the threshold, by the temperature. The division algorithm is implemented by iterate subtractions. Before performing division, the three most significant bits of the dividend are checked: if at least one bit is “1”, then the result will exceed the maximum value allowed for neuron status. In this case the output is set to 255. The dividend sign is also evaluated: if it is “1” (i.e. negative dividend), then the output is set to “0”, since the result will be less than the minimum status value. In all other cases division is performed by eight pipelined steps, each one providing one bit of the result, while the WSS is being processing the next input sets. The division section takes eight steps, one for each bit of the result needed. The neuron is designed in order to allow division to be performed on the inputs at time *t* while the weighted sum section is processing the inputs at time *t+1*, reducing the overall computation time.

The final schematic of the whole neuron is shown in Figure 4. At the top one can see the sets of registers, the central area is filled by the WSS, while the SFS lays in the lower area of the schematic.

An estimate on the overall computation time, after the first transient phase, can be given on this basis: the highest line propagation time is calculated by the simulation program in 100ns for the longest line in the schematic. Therefore one can oversize the clockstep, putting it to 100ns. The global evolution time is 13 clock cycles. Therefore the whole neuron takes 1.3 µs to perform one step of its dynamics. The network is to work in parallel, thus yielding a global recognition time of 1.3 µs per cycle, i.e., 3.9 µs.

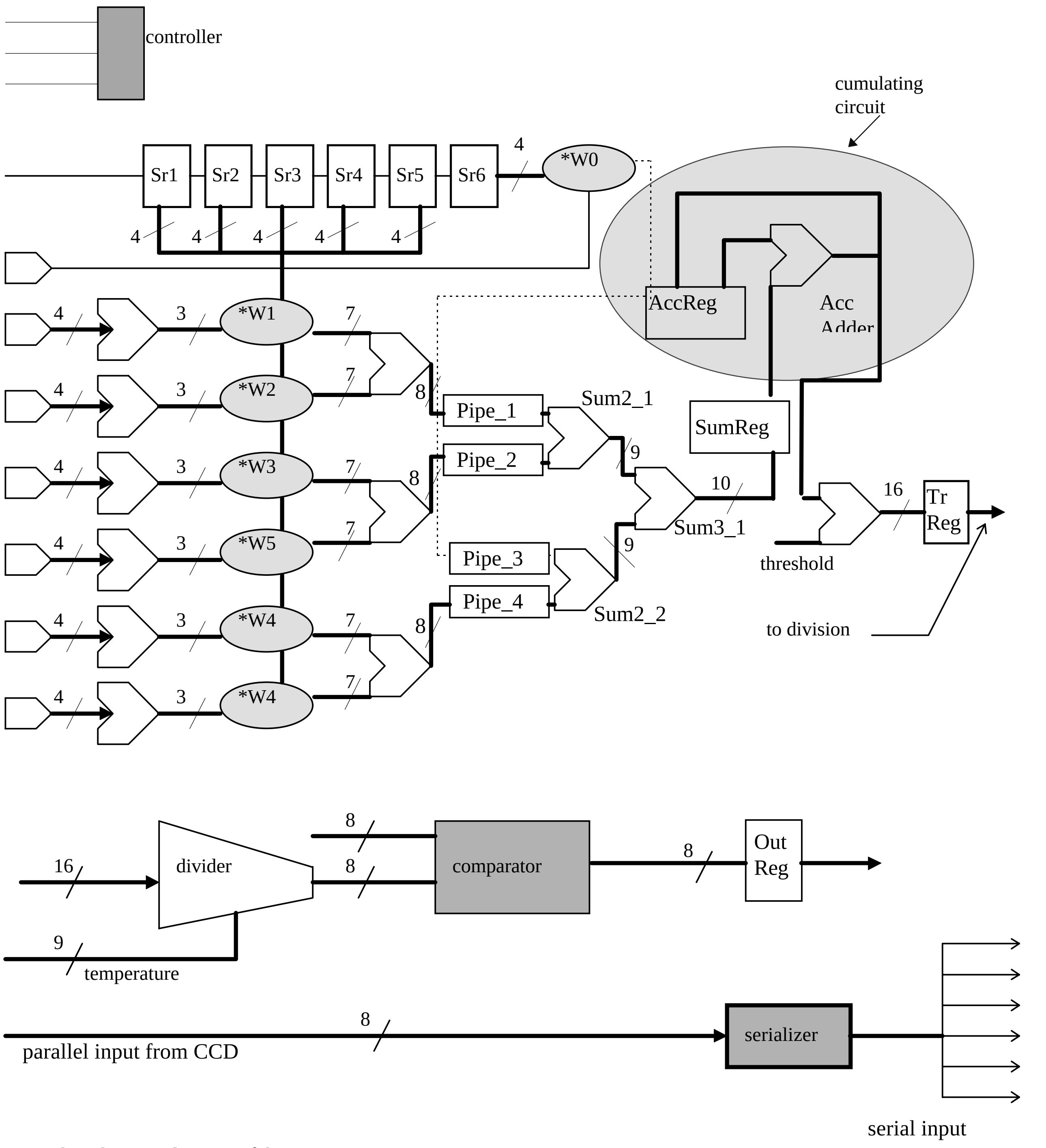

Figure 4: The schematic diagram of the neuron.

The overall recognition time is mostly due to the WSS, because it has to accumulate eight partial results, so it takes eight clocksteps, plus a fixed delay of three clocksteps. The SFS works in parallel with the WSS.

5. The concurrent error detection capabilities

In the serial architecture of the dedicated massive computing system as described in Section 4, the fault model adopted for the FPGA-based implemention is the traditional single gate-level fault model. It is worth noting that other kinds of faults may occur in the FPGAs, but they may be so devastating (leading even to the collapsing of most of the device) that no technique within the individual FPGA is appropriate; on the other hand, these faults are relatively rare so that they can be neglected.

Since we are concerned with protection of the digital components, the fault model allocates the possible faults into the weight storage, the arithmetic devices, and the activation function evaluators; as usual in most fault-tolerance approaches, interconnections are assumed fault-free, unless differently noted.

In the envisioned implementation, it is possible to design the digital structures [14] so that the effect of the adopted fault model onto the computation can be described by the single-error assumption.

Data coding is often considered as a good compromise between the circuit complexity and the computational delay introduced by encoded operations in the neural architecture [12]. The parity code [13] is well suited to protect the memory storage, as it is widely recognized. For the arithmetic devices, we identified AN codes as capable of satisfying all the above requirements for concurrent error detection, given reasonable choices for A and B [11].

AN codes are non-systematic codes in which each nominal datum N is substituted by its coded representation C(N) via the linear transformation C(N)=A*N, where the integer constant A is the code generator. Arithmetic units (namely, adders, subtractors, multipliers) are not modified by the adoption of the coded data; the only difference with respect to the nominal structure is given by the number of bits required, that obviously increases with A. The coding unit is therefore a simple multiplier; since A is a constant, its structure can be optimized to grant maximum compactness and speed. Decoding is performed, for addition and subtraction involving coded operands, by applying the linear antitransformation N=C(N)/A; the same decoding holds for multiplication whenever one operand only is in coded form. If both operands are coded, for a multiplication the antitransformation becomes $N=C(N)/A^2$. Whenever the antitransformation produces a non-null residue, an error is detected. Aliasing is possible whenever the error is a multiple of the code generator (A or A^2); a number of papers have been published on optimum choice of A to minimize aliasing in the various arithmetic units [13]. In particular, it has been proved that choosing A=3 gives satisfactory detection capacity for all arithmetic units in which a single fault induces an error that can be represented as the addition of a power of 2; moreover, both encoder and decoder are quite simple and representation of the coded operands is only two bits longer than the nominal one.

The application of the coding techniques must take into account the specific structure of the neural architecture described in Section 4.

As the shift register subsystem is concerned, it is worth - but effective - adopting the traditional parity coding. Encoding should be performed in the CCD camera to guarantee protection of the interconnection path. However, since we cannot modify the internal structure of the commercially-available CCD camera in the present version, we assume - as it is anyway reasonable - that this interconnection is fault free. Coding is therefore performed at the shift register input. Checking is performed in the FPGA dedicated to extract the pixels from the shift register in the order expected by the neural system. Operation within such an FPGA is duplicated and compared for self checking.

Consider now each neuron implemented by an FPGA. The scheme of the architecture with concurrent error detection capabilities is shown in Figure 5. One input per neuron is presented to the FPGA corresponding to such a neuron in a bit-parallel way and stored in two shift registers. The input block serializes and distributes this input to all neurons. Serialization is duplicated and compared for checking.
Distribution is bit-serially performed: each receiving FPGA, during the input acquisition, computes the 3N representation. The encoder is a serial adder computing C(N)= N+2N; the nominal input is presented twice: one is shifted by one clock cycle with respect to the second one. Each encoded input is stored in a separate shift register. Due to the simplicity of encoding and storing, these structures are duplicated with voting. Also the interconnection weights of the neural network are stored in the weight registers in the 3N code to protect also these registers during the system operations as well as the interconnecting paths among neurons.

The arithmetic operations generating the weight inputs' summation can be directly protected by the use of the 3N code for all operands (namely, inputs and weights). Note that, as a consequence, each weighted input is given in the 9N code as well as the whole weighted summation. The threshold value (in the 9N code) is then added to the weighted summation to generate the excitation signal for the linear activation function. The additional circuit complexity due to the use of all coded operands with respect to the minimum required by single-error detection is acceptable to guarantee protection also of the interconnection paths if the weights are not multiple of 3; on the other hand, the use of FPGA does not require to minimize hardly the number of gates as it is usual in the standard VLSI approach, being sufficient not to exceed the capacity of the FPGA itself.
No intermediate checking or decoding is required in all the above structure to preserve the single-error assumption since no aliasing is induced by possible reconvergent data paths or cycles.
The registers used for pipelining the operations are protected as well as the arithmetic units without introducing any additional technique since a fault in any of them appears as a single error in the subsequent arithmetic unit.

The excitation signal is then scaled to support the subsequent direct evaluation of the energy associated to the photon activating the pixels. This operation is performed by dividing the weighted summation by a suited factor computed during the neural learning procedure. Being division implemented by iterated subtractions, checking must be introduced at the accumulator output in order to preserve the single-error assumption.
The final result of the division is the coded output of the neuron: it is transferred to the other neurons during the subsequent iteration of the neural operations, while recoding in the 3N code is performed.

Checking of the neural operation is performed locally within each individual neuron concurrently with the nominal computation. The datum Y=3X belongs to the code if it is divisible by 3. Checking consists of dividing Y by 3 by means of the subtraction X=Y-2X, and by verifying that the most significant bits of the result are zeros [11]. A similar technique can be adopted also for checking the 9N code. The presence of a fault in the individual neuron is pointed out by the neuron's error signal. Possible errors are propagated to the other neurons, but the first error signal becoming active identifies implicitly the faulty neuron, without any additional overhead in circuit complexity and latency.

Due to the limited number of neurons in the envisioned application, localization of the faulty component can be obtained by observing the value of each neuron's error signal at each neural iteration. This could be exploited in the future for dynamic reconfiguration of the system.

The prototype implementation of the fault tolerant neural network for the envisioned application gave attractive results with respect to the structure without concurrent detection capabilities. Each neuron has been implemented by using one Xilinx XC4013 FPGA. The version of the neuron without fault-

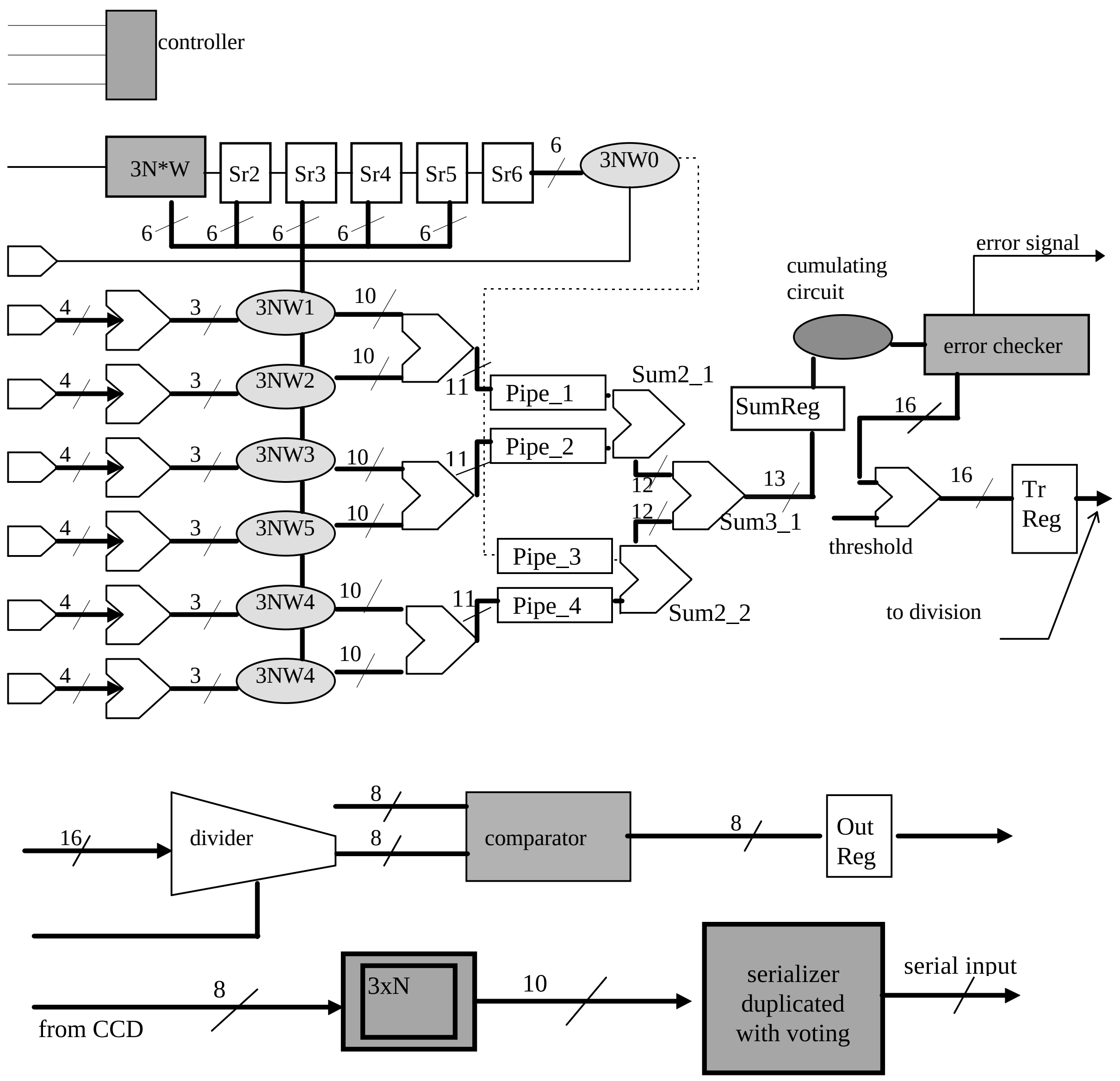

Figure 5: The schematic diagram of the fault-tolerant neuron.

tolerance features occupies 327 CLB's (56% of the total CLB’s of the XC4013) and 110 I/O pins (57% of the total pins of the XC4013). The fault-tolerant version of the neuron (including both the nominal computational features and the concurrent error detection circuits) uses 415 CLB's (72% of the total CLB's of the XC4013) and 118 I/O pins (61% of the total pins of the XC4013). The additional circuit complexity introduced for fault tolerance in each neuron (measured as number of CLB’s) is about 35% with respect to the basic structure without fault tolerance features, while the increase of the interconnection complexity (measured as number of pins) is less than 8%.

The time diagram of the neuron behaviour is given in Figure 6. In particular, it points out the time behaviour and the stability of the pipeline stages within the neuron structure. The design was performed to maximize the system throughput by accurately balancing the operations in each pipeline stage, as it is shown by the input and the output signals of the pipeline registers. With reference to the schematic diagram of Figure 5: PIPE_IN_1 and PIPE_IN_3 are the input signals to Pipe_1 and Pipe_3 registers; PIPE_M_1 and PIPE_M_3 are the output signals from Pipe_1 and Pipe_3 masters; PIPE_S_1 and PIPE_S_3 are the output signals from Pipe_1 and Pipe_3 slaves; SUM2_1, SUM_2_2 and SUM3_1 are the output signals from corresponding sum devices; SUM_3_M is the output signal from SumReg register master and finally SUMREG is the output signal from SumReg slave. PIPE_IN_2 and PIPE_IN_4, PIPE_M_2 and PIPE_M_4, and PIPE_S_2 and PIPE_S_4 are not shown as they have the same behaviour of PIPE_IN_1, PIPE_M_1 and PIPE_S_1 signals respectively. It can be noticed that SUM_2_1 and SUM_2_2 correctly become stable later than PIPE_S_3 and the same applies to SUM3_1 with respect to SUM2_1 and SUM2_2.

The system needs 13 clock cycles in the non fault-tolerant version to generate each neuron output, being 150ns the duration of the clock cycle. In the architecture with concurrent detection, 17 cycles are necessary to complete the operation, i.e., about 30% more than the basic approach.

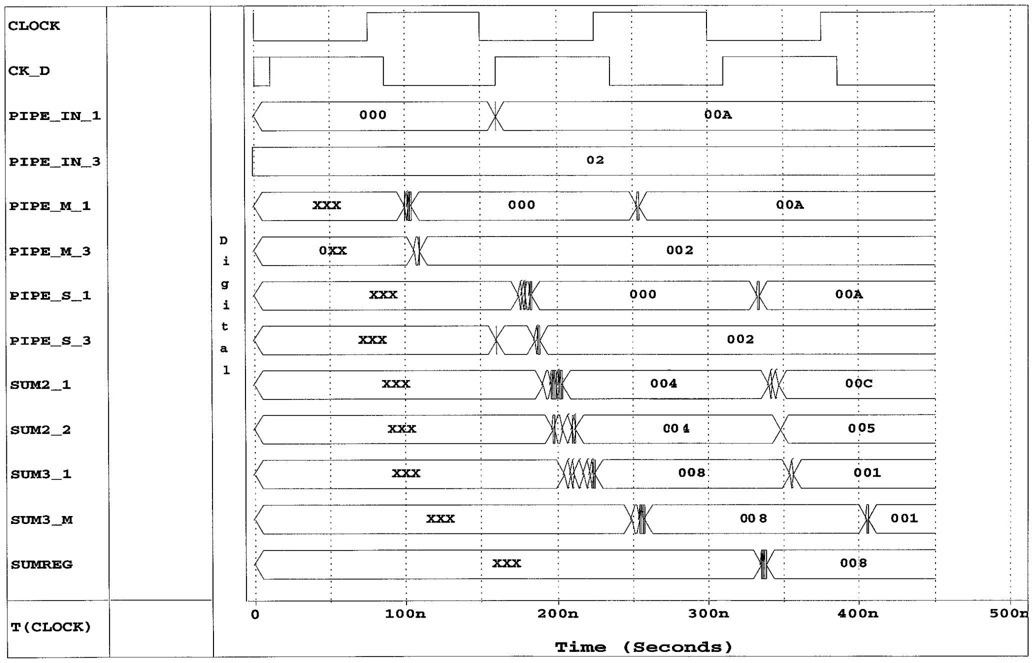

Figure 6: The time diagram.

6. Conclusions

A fault tolerant neural network implementation based on FPGA was presented. The resulting architecture has a good level of flexibility and modularity, which are desirable properties for rapid prototyping. Current performance matches CCD camera throughput. Further research will be directed to improve the characteristics of the camera and of the neuron, both from the point of view of speed and fault tolerance. As a consequence, the internal structure of the neuron will be revised to reduce clock cycle time. In particular, we are studying a faster control signals distribution scheme to reduce the control propagation delay. In fact, the arithmetic part is able to operate at a 50 MHz clock rate (20 ns. clock cycle), but the system clock must be delayed to 150 ns. to guarantee a correct control propagation through the whole FPGA device since the FPGA synthesis tool introduces a poor routing for such signals.

References

[1] P. Achdjian, C. Baroncelli, S. D'Angelo, M. Dapri, and G.R. Sechi, "A Hardware Prototype of a Neuron for Signal Processing", *Proc. IASTED Int. Symp. on Applied Informatics*, Innsbruck, Austria, Feb. 21-23, 1995, pp. 79-82.

[2] P. Achdjian, M. Alderighi, S. D'Angelo, G.R. Sechi, E.G. Tanzi, and M. Uslenghi, "Neural Network Based Event Identification in a Photon Counting Intensified CCD: Performance Evaluation", *Proc. Int. Conf. on Massively Parallel Computing Systems MPCS'96*, Ischia, Italy, May 6-9, 1996.

[3] M. Alderighi, F. d'Ovidio, E. Gummati, and G.R. Sechi, "Analysis and Evaluation of a Neural Network Performing Digital Filtering", *Proc. IASTED Int. Conf. on Artificial Intelligence, Expert Systems and Neural Networks*, Zurich, Switzerland, July 4-6, 1994, pp. 106 - 110.

[4] M. Alderighi, D. Crosetto, F. d'Ovidio, E. Gummati, and G.R. Sechi, "A Feedback Neural Network for Signal Processing and Event Recognition", *Proc. IEEE Int. Conf. on Algorithms and Architectures for Parallel Processing*, Brisbane, Australia, April 19-22, 1995, pp. 788-791.

[5] M. Alderighi, S. D'Angelo, F. d'Ovidio, E. Gummati, and G.R. Sechi, "An Advanced Neuron Model for Optimizing the SIREN Network Architecture", *Proc. IEEE Int. Conf. on Algorithms and Architectures for Parallel Processing*, Singapore, June 11-13, 1996, pp. 194 - 200.

[6] C. Alippi, V. Piuri, and M. Sami, "Sensitivity to Errors in Artificial Neural Networks: a Behavioral Approach", *IEEE Trans. on Circuits and Systems - 1: Fundamental theory and applications*, vol. 42, no. 6, June 1995, pp. 358-361

[7] C.E. Cox, W.E. Blanz, "GANGLION - A Fast Field-Programmable Gate Array Implementation of a Connectionist Classifier", *IEEE Jour. of Solid State Circuits*, vol. 27, n. 3, March 1992, pp. 288-299.

[8] R. Eckmiller, and C.v.d. Malsburg, *Neural Computers*, NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 41, Springer Verlag, Germany, 1988.

[9] V. Piuri, M. Sami, and R. Stefanelli, "Fault Tolerance in Neural Networks: Theoretical Analysis and Simulation Results", *Proc. Compeuro 1991*, Bologna, Italy, May 1991.

[10] V. Piuri, M. Sami, and R. Stefanelli, "Neural Networks on Silicon: the Mapping of Hardware Faults onto Behavioral Errors", *Proc. Int'l Workshop on Defect and Fault Tolerance 1991*, Hidden Valley, USA, Nov. 1991.

[11] V. Piuri, and R. Stefanelli, "Efficient Use of the 3N Code for Concurrent Error Detection in Parallel Multipliers", *Computer Arithmetic, Scientific Computation and Mathematical Modelling*, E. Kaucher, S.M. Markov, G. Myer editors, J.C. Baltzer AG, Scientific Publishing Co., 1991

[12] V. Piuri, M. Sami, and R. Stefanelli, "Arithmetic Codes for Concurrent Error Detection in Artificial Neural Networks: the Case of AN+B Codes", *Proc. Int'l. Workshop on Defect and Fault Tolerance in VLSI Systems*, Dallas, TX, 1992.

[13] T.R.N. Rao, *Error Coding for Arithmetic Processors*, Academic Press, NY, 1974.

[14] R. Stefanelli, and M. Annaratone, "A Multiplier with Multiple Error Correction Capability", *Proc. ARITH-6*, 1983.

[15] E.G. Tanzi, "Photon Counting and Analog Intensified Imagers for UV and X-Ray Radiation", *IFCTR-CNR Internal Report*, January 1995.