ELIPS: Toward a sensor fusion processor on a chip

Taher Daud, Adrian Stoica, Thomas Tyson, Wei-te Li, and James Fabunmi[a]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109
[a]AEDAR Corporation, P.O. Box 1469, Landover, MD 20785

ABSTRACT

The paper presents the concept and initial tests from the hardware implementation of a low-power, high-speed reconfigurable sensor fusion processor. The Extended Logic Intelligent Processing System (ELIPS) processor is developed to seamlessly combine rule-based systems, fuzzy logic, and neural networks to achieve parallel fusion of sensor in compact low power VLSI. The first demonstration of the ELIPS concept targets interceptor functionality; other applications, mainly in robotics and autonomous systems are considered for the future. The main assumption behind ELIPS is that fuzzy, rule-based and neural forms of computation can serve as the main primitives of an "intelligent" processor. Thus, in the same way classic processors are designed to optimize the hardware implementation of a set of fundamental operations, ELIPS is developed as an efficient implementation of computational intelligence primitives, and relies on a set of fuzzy set, fuzzy inference and neural modules, built in programmable analog hardware. The hardware programmability allows the processor to reconfigure into different machines, taking the most efficient hardware implementation during each phase of information processing. Following software demonstrations on several interceptor data, three important ELIPS building blocks (a fuzzy set preprocessor, a rule-based fuzzy system and a neural network) have been fabricated in analog VLSI hardware and demonstrated microsecond-processing times.

Keywords: sensor fusion hardware, sensor fusion processor, fuzzy expert system, neural networks, reconfigurable hardware

1. INTRODUCTION

1.1 A general need for sensor fusion processors

With the advent of recent increasingly high-performance sensors and processing power a plethora of novel applications are imagined using multiple sensors, many times of various complementary nature. Novel architectures, algorithms and hardware are required to optimally address the sensor fusion challenges of high-bandwidth, often noisy, sometimes contradictory data. The problem of using more sensors with higher data rates is aggravated by the need for faster response time, which demands higher levels of computational power. The traditional approach is to build/use increasingly powerful general-purpose processors. Yet, classical algorithms for fusing data (originating in preponderant Bayesian approaches) face challenges in addressing the sensor-fusion problem and need complemented/overridden by novel approaches, such as the ones coming from the computational intelligence research.

Computational intelligence techniques, such as fuzzy logic and neural networks combined with the more traditional Artificial Intelligence paradigm of expert systems proved efficient in a solving a category of problems for which an accurate mathematical formulation of models was either not feasible or practically impossible to compute in useful time. The most eloquent examples of such problems are in pattern recognition and decision-making applications. These techniques are essentially parallel, and thus it is natural to build dedicated processors efficient for these types of operations, which would function in stand-alone mode or as co-processors to provide high-speed computation on massive amounts of data in parallel mode. While these processors can be built both in digital or analog hardware, the massive amount of interconnection lines of a parallel implementation and the power requirements encountered in certain space, military or commercial applications such as hand-held devices make the idea of an analog ASIC processor preferable. An example of such an application requiring low power fast processing of sensor data is associated with the discrimination performed onboard interceptors.

2.2 Discriminating Interceptor Technology requirements for an on-board sensor fusion processor

The Ballistic Missile Defense Organization (BMDO) is conducting the Discriminating Interceptor Technology Program (DITP) for the development of advanced and enabling fast frame seeker capabilities. The challenge for the technology is to combat more complex future threats facing the National and Theater Missile Defense (NMD/TMD). The objective is to develop miniaturized interceptor components and subsystems to meet serious space, weight, and power constraints [1]. In this regard, part of a major effort is directed towards the development of new sensor data fusion processing technology that will particularly address high speed and on-board autonomy. This capability can achieve earlier target acquisition, thereby extending the time-to-engage and reducing the dependence on the external battle management and off-board surveillance assets [1].

Once the initially required off-board battle management intelligence is provided to the seeker, the primary goal of the DITP is to exploit the multi-phenomenological sensor data obtained from on-board LADAR and infrared detector arrays for threat engagement via development and integration of real-time sensor fusion algorithms and processors. The overriding hypothesis is that sensor data fusion at three levels (i.e., signal, feature, and decision) is necessary to improve its capability and to accommodate a wide variety of missions and targets.

In order to meet the challenge of compact, low power, and high-speed on-board data processing, a novel intelligent sensor data fusion processing architecture, termed the Extended Logic Intelligent Processing System (ELIPS), has been developed. ELIPS integrates the analog hardware technology of neural networks, fuzzy logic, and expert rule processing with the conventional digital processing using a host computer. The individual modules are designed to be reconfigurable and cascadable. In addition, the overall architecture has been developed to be flexible enough for rerouting of signals to any required processing module by having an interconnecting network with switching arrays.

This paper briefly describes the ELIPS concept and architecture, focusing more on the hardware implementation of the individual ELIPS component modules. Experiments with test chips implementing ELIPS modules illustrate the performance of the analog ASIC implementation.

2. FUZZY, EXPERT AND NEURAL COMPUTATION: FUNDAMENTALS AND PREVIOUS DEDICATED HARDWARE IMPLEMENTATIONS

2.1 Fundamentals of fuzzy, expert and neural computation

Expert systems are considered in the sensor fusion literature to have a variety of utilities. An example detailed in [2] is guiding the user in defining the architecture for the sensor fusion system. Fuzzy logic and neural networks are also becoming widely accepted in the sensor fusion community as techniques which proved powerful in sensor fusion applications [3], [4].

Conditional rule-based systems are using rules of the form "IF a is A AND b is B THEN y is Y' where a, b, and y the input and output variables respectively, and A, B, Y are classes - in particular fuzzy classes/sets. Thus, a rule-base system can be seen as accepting input data from measurements or preprocessing and providing outputs as transformed by the rules. In particular the outputs could be associated with classes to which the inputs cluster and the magnitude of the outputs associated to the degree of membership to these classes. (Another possible interpretation is that the numbers represent the confidence in the classification, e.g. 70% confidence that the object is target1, 20% that it is target2, 10% confidence that it is decoy.

New concepts from fuzzy sets theory have revitalized the use of rule-base system, which can thus better cope with the imprecision in matching antecedent clauses. The main operations of fuzzy reasoning are fuzzification, rule evaluations and defuzzification. Fuzzification transforms a crisp input to a degree of membership to a fuzzy set and certain rules are evaluated depending on which fuzzy sets are matched. For certain problems such as classification, this is the end of fuzzy reasoning - the output results are fuzzy sets and degrees to which they are matched. For example, the output result can be that input signals match the characteristics of target A to 0.8 extent, targets B in degree 0.4 and decoys in degree 0.3; sometimes this can be (improperly) expressed as probabilities, i.e., there is 80% chance/probability/confidence that object is target A, etc. If the desired output is a crisp one, for example an output control signal - the output sets and the associated degrees of memberships are transformed by a defuzzifier into a crisp value. Amongst the most popular methods for defuzzification is the center of gravity method, which requires mainly additions and multiplication and division.

Neural networks are parallel computation structures characterized by somatic operation between inputs and weights and somatic operations aggregating the weighted inputs and usually passing them through a nonlinear function. Different neural architectures were explored, with different ways of interconnecting the neurons in feed-forward only or in recurrent mode as well, and with a variety of learning rules.

Requirements for fast processing, compact or low power implementation lead to efforts for developing various hardware implementations. The nature of computations involved in fuzzy reasoning is essentially parallel (for example, rule evaluations are independent of each other and can be calculated concurrently). A dedicated parallel hardware solution is therefore preferable to a software solution on a general-purpose processor and even to a RISC processor with fuzzy-oriented instructions like VY86C570 (70-microsecond inference speed) [5]. Ideally one would want to preserve high versatility of general-purpose processors while reaching low-power high-speed operation. Analog offers the advantage of lower power consumption. While better precision can be obtained in digital implementations, very precise computations are not required for fuzzy processing; usually 8 bits are considered sufficient for most applications. (This relaxed restriction on precision is due to the fact that membership functions representing fuzzy classes are usually defined by humans, who can do not specify fuzzy set borders with high precision - usually with less than 8 bits) [7-9].
The same parallelism is true for neural processing, and ideally hardware implementations should be parallel for maximum efficiency. In the same way as for fuzzy expert systems, large number of interconnections and low power justify analog VLSI implementations of neural processors. For a detailed justification of analog neural processors see [10].

3. ELIPS CONCEPT AND ARCHITECTURE

The main assumption behind ELIPS is that fuzzy, rule-based and neural forms of computation can serve as the main primitives of an "intelligent" processor. Thus, in the same way classic processors are designed to optimize the hardware implementation of a set of fundamental operations, ELIPS is developed as an efficient implementation of computational intelligence primitives, and relies on a set of fuzzy set, fuzzy inference and neural modules, built in programmable analog hardware. The hardware programmability allows the processor to reconfigure into different machines, taking the most efficient hardware implementation during each phase of information processing.
The ELIPS architecture is designed to accomplish, for the first time, a fully parallel implementation and seamless integration of three artificial/computational intelligence technologies: (1) membership-function-based fuzzy logic; (2) rule-based expert systems; and (3) massively parallel artificial neural network. In its initial demonstration ELIPS will perform various DITP functions of discrimination, recognition, tracking, and homing [2]. It is necessary to develop a design that is hardware-implementable using very large scale integration (VLSI) technology to provide an ultra low power embodiment in a compact package, with an unprecedented signal processing speed (10 to 15 microseconds for each operation), at least three orders of magnitude faster compared to a conventional digital machine (e.g. several milliseconds on a personal computer, PC).

ELIPS is envisaged as a synergistic processor incorporating four processing modules illustrated in Figure 1. FSP is a Fuzzy Set Processor, MERP stands for Multistage Expert Rule Processor, and PFN and PRN refer to Programmable Feed-forward and Recurrent (feedback) Neural networks, respectively. ELIPS modules are destined to work cooperatively in a variety of configuration sequences. For example, to implement fuzzy expert reasoning as a processing sequence of FSP, MERP and PFN modules, fuzzification is performed by FSP, rule evaluation is done by MERP, while defuzzification (when needed) is done using the PFN.

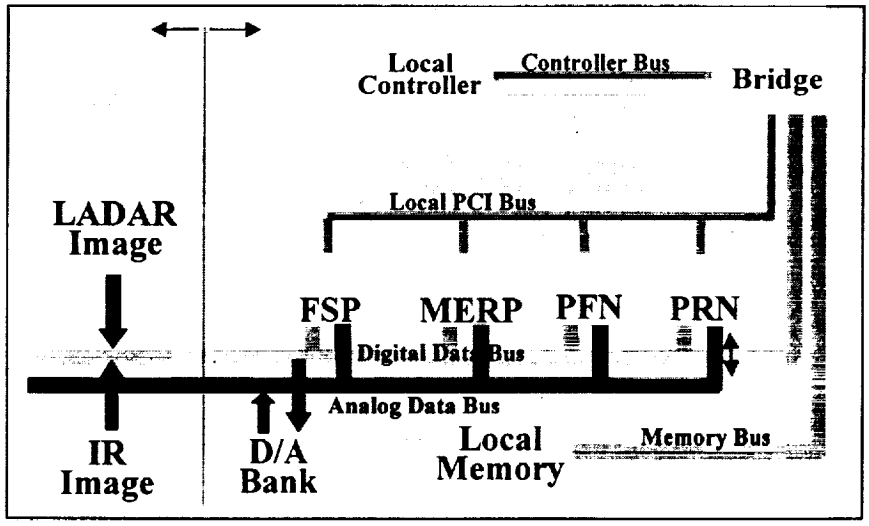

Figure 1. ELIPS architecture and main computational modules

4. ELIPS BUILDING BLOCKS AND THEIR HARDWARE IMPLEMENTATION

4.1 The fuzzy set module: FSP

The main function of a fuzzy set processor is signal transformation, which can be interpreted for example as

- fuzzification - i.e. association between an input crisp signal and a degree of membership to a fuzzy set/class, or
- signal conditioning/ non-linear transformation, coordinate transformation.

The FSP was designed as a processing module with 16 inputs of 5 membership classes each. The architecture of the FSP is presented in Figure 2. The chip has 16 analog voltage inputs and 16x5 outputs, and allows digital programmability of the membership functions for each input variable.

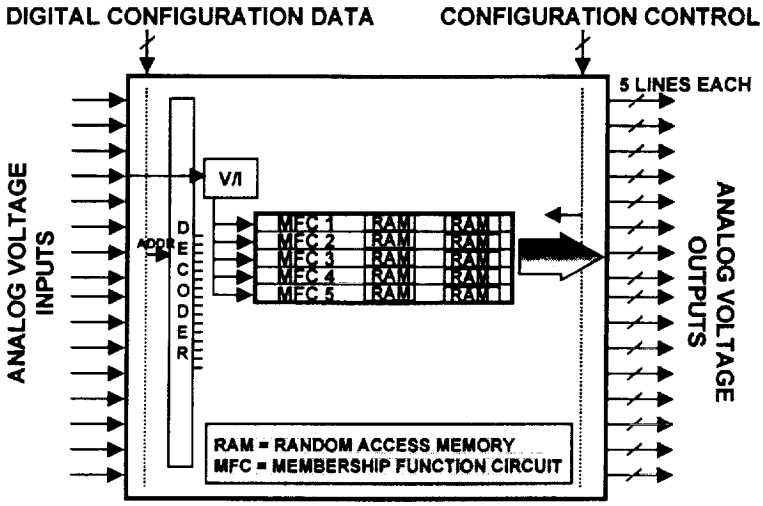

Figure 2. FSP architecture

The membership functions have trapezoidal shape, with programmable parameters for the legs and slopes as illustrated in Figure 3. The position of the legs can be specified with 8-bit resolution and the slope with 5-bit resolution. The equations that describe the output of a trapezoidal membership function are:

If $X <= A$, $Y = Low$

If $A < X <= (CD+AB)/(B+C)$, $Y=MIN[BX-AB + Low), High]$

If $(CD+AB)/(B+C) < X < D$, $Y=MIN[-CX + CD + Low), High]$

If $X >= D$, $Y= Low$

where A is the location of the left leg, B is the unsigned slope of the left leg, C is the unsigned slope of the right leg, and D is the location of the right leg. The chip design currently uses Low = 1 volt and High = 4 volts with Vdd = 5 volts.

The schematic diagram in Figure 4 details the processing path of a single membership function circuit (MFC). While inputs and outputs are in voltage mode for external compatibility, the internal MFC implementation is in current-mode. The input voltage enters the first processing block which is a Voltage to Current (V/I) converter. Currents proportional to the digital values of the legs, A and D, are generated in Multiplying Digital to Analog Converters (MDACs). The current corresponding to the left leg gets subtracted from a copy of the input current, while a different copy of the input current gets subtracted from the right leg current. The resulting currents, which correspond to the left and right sides of the trapezoid, enter their appropriate Dividing Digital to Analog Converter (divDAC) where the signals are divided by 5-bit digital values to scale the slopes. The minimum of the two resulting values is then selected which chooses the side that is along the trapezoid. The top of the trapezoid is achieved by taking the minimum of the resulting current and the full-scale current, and this result is converted to the voltage output of the MFC. A test chip for 2 input variable with 5 membership functions calculating the degree of membership has been implemented and tested. A variety of membership functions generated by the chip is illustrated in Figure 4.

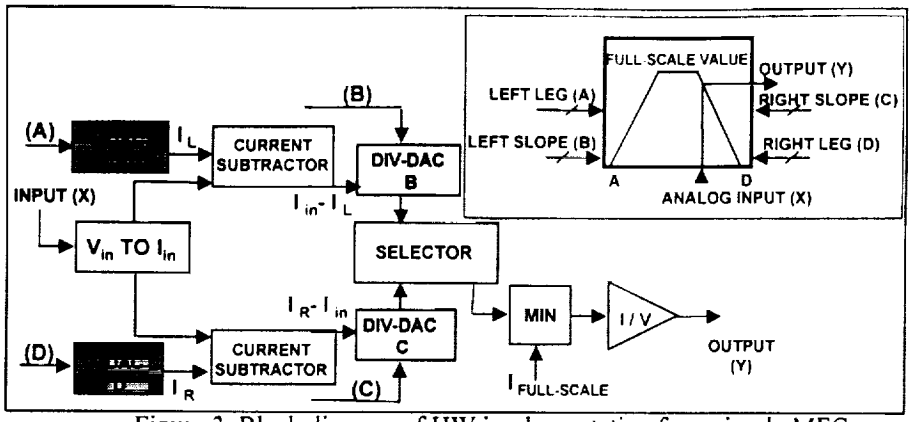

Figure 3. Block diagram of HW implementation for a simple MFC

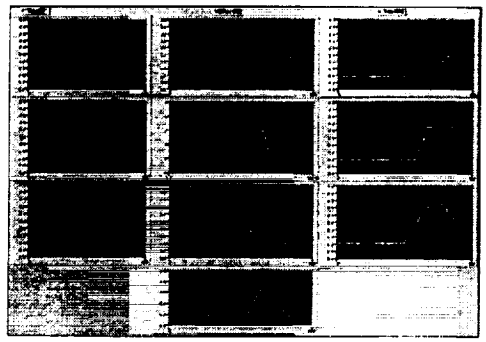

Figure 4. A variety of membership function shapes generated on the MFC test chip

Signals obtained from the chip are also illustrated below in a discrimination task. The results are compared with the software implementation and show accurate reproduction in hardware of the results obtained by simulation. Figure 5 shows how the membership functions are used to separate the spaces containing targets and decoys. The variables are transformations of some measured parameters characterizing targets and decoys signals. Figure 6 shows only discrimination between targets and decoys. Figures 6 shows further discrimination distinguishing individual targets. The hardware tests show that the fuzzification /discrimination of this type takes less than a microsecond.

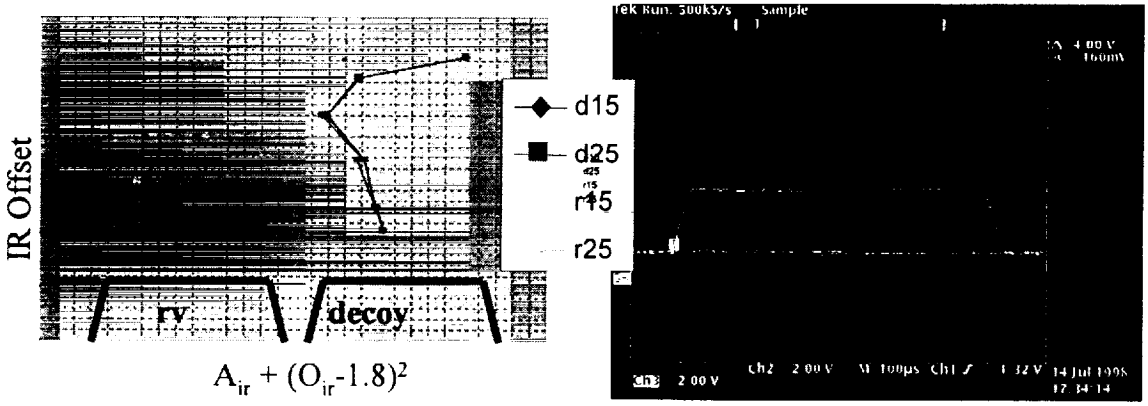

Figure 5. SW and HW generated membership functions in a discrimination task: target vs. decoy

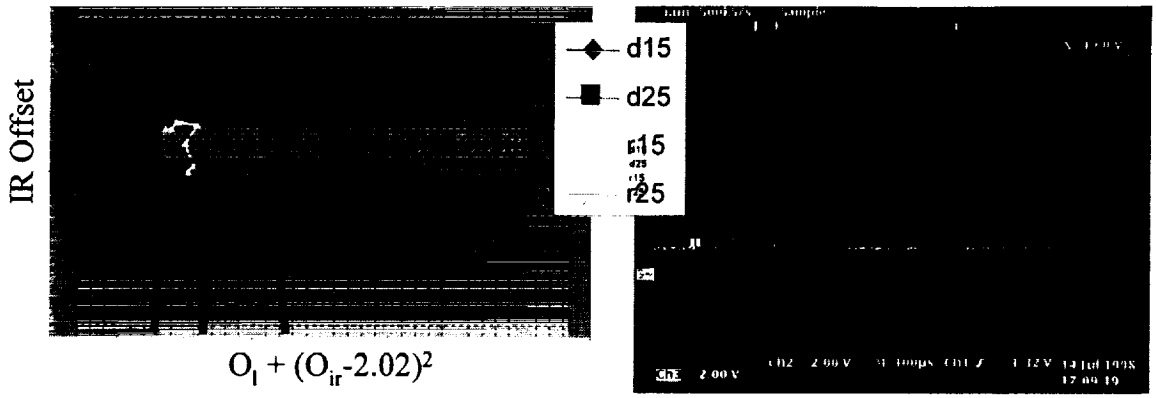

Figure 6. SW and HW generated membership functions in a discrimination task: target1 vs. target2

4.2 The expert-rule module: MERP

The main function of a rule processor is to evaluate matches between input data and classes of knowledge (the satisfaction of certain conditions by the input) and prescribe the implications for such cases.
The general structure of processing in MERP is by inference on a collection of rules of the form:

Rule 1. IF a_1 is A_{11} AND a_2 is A_{12} AND ...a_m is A_{1m} THEN y is Y_1

...

Rule n. IF a_1 is A_{n1} AND a_2 is A_{n2} AND ...a_m is A_{nm} THEN y is Y_n

where A_{ij} are fuzzy sets or their complements, i.e. if A_{lm} is a predetermined trapezoidal membership function/fuzzy set and A_{fk} is its complement then A_{fk} = NOT(A_{lm}). Consider the degree of membership/matching a fuzzy set/class being calculated by the FSP, and thus "a is A" being replaced with u, which is the degree to which "a is A". The complement is commonly calculated either as the difference to unity, i.e. NOT(u) = 1-u, or as the maximum of all other classes except the one to be complemented, i.e. if classes covering input space are u1,u2,u3,u4 then the complement is NOT(u3) = MAX(u1,u2,u4). We built test circuitry to calculate the complement in both ways but only the second version was so far integrated within a rule-system chip. The conjunction AND is treated as the MAX operator. Thus, the antecedent "a_1 is A_{n1} AND a_2 is A_{n2} AND ...a_m is A_{nm}" can be read after fuzzification as (u_{n1} AND u_{n2} AND u_{nm}) and calculated as u_n = MIN(u_{n1}, u_{n2}, ..., u_{nm}). The collection of rules in the rules base can be read as Rule1 OR Rule 2 OR...Rulen; several rules may refer to the same conclusion/class. The logical connective OR is calculated as MAX, thus the degree of supporting an output class is the maximum of all the degrees of supporting that class coming from different rules in the rule-base.

The processing stages calculating complement, conjunction and disjunction are reflected directly in the MERP architecture presented schematically in Figure 7. Stage 1 calculates the complement by MAX operation; Stage 2 calculates the conjunction within the same rule by MIN operator; Stage 3 calculates the disjunction of all rules that refer to the same conclusion by MAX operator. The controls specify which components are selected for MIN and MAX in different rules.

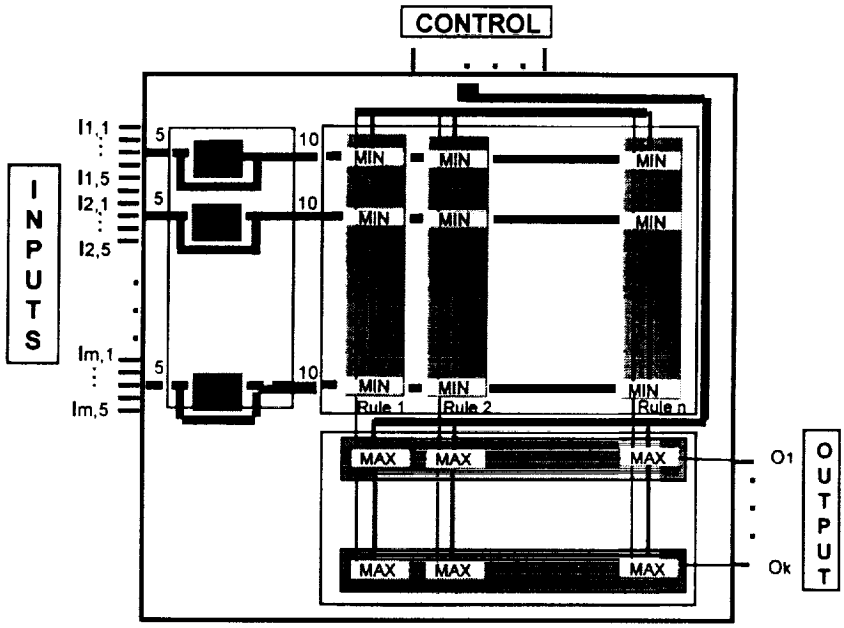

Figure 7. Schematic of MERP architecture

Figure 8 goes into a more detailed implementation design of the MERP module. The MERP module is designed as a processing module with 16 inputs with 5 membership classes each; a complement is calculated for each membership class inside the module. The module supports rules with up to 64 conjunctions; up to 128 rules can be programmed in the module and 32 decisions can be obtained as outputs. The implementation of the MERP module is performed in four development phases allowing testing of various circuits (such as analog MIN and MAX circuits) and system/integration solutions before a full-scale more expensive chip is attempted. Figure 9 shows test results from a fabricated MIN circuit (the upper waveforms are the input and the lower one is the output, which is the minimum of the two).

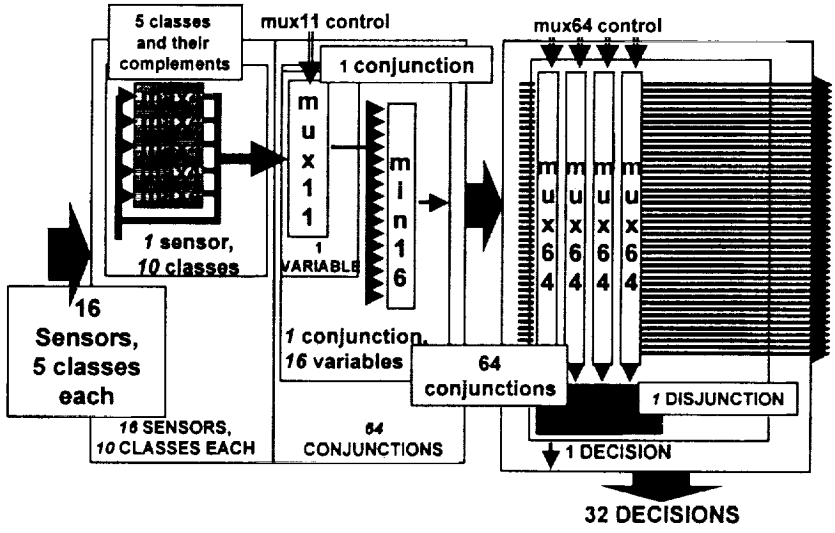

Figure 8. A more detailed design of MERP architecture

In the second a smaller version of MERP (called miniMERP) with 2 inputs and 4 rules was fabricated on a test chip. The block diagram for the chip is illustrated in Figure 10; the chip was fabricated and tested successfully. The propagation time of a signal from inputs to output was around one microsecond. Phase 3 of development consist in integrating 8 analog inputs, 40 membership functions and 9 rules circuits on the same Fuzzy Expert System (FES) chip. The membership functions are digitally programmable trapezoids. The rules are digitally programmed to select from various membership functions for each input variable, including membership function complements. Each rule performs a conjunction amongst selected membership functions and their complements (one per variable). All analog circuitry is current-mode and the rule output currents are available in parallel on nine separate lines. The chip was fabricated and is currently under tests.

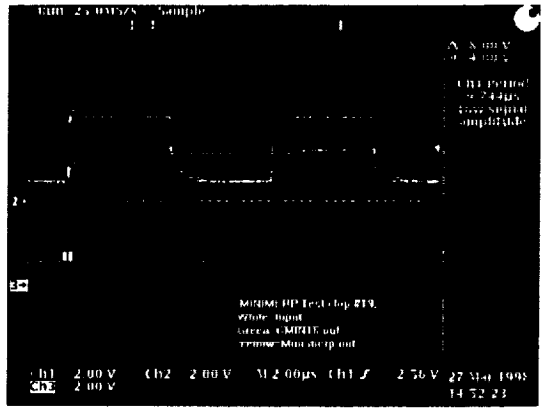

Figure 9. Speed test on a MERP MIN circuit

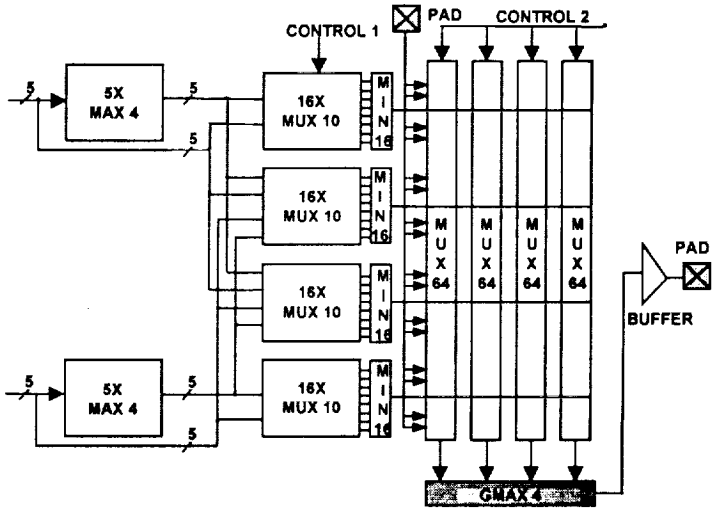

Figure 10. Block diagram of an analog mini-MERP test chip

4.3 The neural modules: PFN and PRN

Neural network modules are implemented around a neural chip architecture developed at JPL [11]. The chip, code named NN64, and schematically illustrated in Figure 11consists of a 64 x 64 array of 8-bit synapses with 8-bit local static memory, 64 neurons, and registers for data and control. The chip is designed to implement a fully interconnected feed-forward neural network with up to 64 inputs and outputs or recurrent neural network topologies.

Functional description of analog processing in NN64. The 64 analog voltage inputs first get converted to currents by a row of V-I converters at the top of the 64 x 64 synaptic array. Each V-I circuit actually produces two currents: I and 16 x I. These signals are then broadcast down each column for each of the 64 inputs so that all the synapses in a column receive the same input.

The building block for the NN64 array is a current-mode multiplying analog to digital converter (MDAC) which forms the basis of the synapse. A byte, which controls switches to scaled current copies of the input, is stored in a local static memory (SRAM) for each synapse. By switching in different multiples of the input current and adding them together, the input current is effectively multiplied by the digital weight stored in the local SRAM. The most significant bit (MSB) of the digital weight controls the sign of the product by steering the synapse output current so that it is either sunk or sourced through the output node. Synapses on the same row have their outputs summed by attaching them all to the same wire. These 64 signals, one for each row of the array, are then sent to 64 separate neurons where they are either processed through the neuron or sent directly out, depending on how the neurons are programmed. If the neuron is on, the current is converted to a voltage through a small resistor and the applied to a small differential amplifier that outputs a voltage. Should the neuron be off, the output current is routed directly out off the chip as a current.

Digital programming of NN64. The synapses are loaded a single row at a time. The data for a given row is clocked into a 64 long 8-bit wide shift register, one byte at a time. After 64 clock cycles, the data for an entire row of synapses is ready to be loaded into the local memory of each MDAC. A 6-bit row address is supplied and an active-low load signal is asserted, which dumps the data into the synapses on the row specified. Alternatively, a synchronous loading scheme may be used. This method employs a single bit shift register to act as a token ring and specify consecutive rows for loading. When reset is asserted, the top of the token ring corresponding to row 1 is set while the rest of the shift register is reset. As data is clocked in, a 6-bit counter keeps track of how many bytes have been loaded. When the carry-out of the counter indicates the data has been entirely loaded, a load signal is automatically generated that activates the row on its rising edge and passes the token to the next row on its falling edge. In this way the entire array of synapses can be loaded from the top row down by simply clocking in 4096 bytes of data. Neurons are also programmed with a single bit shift register. If a control signal is asserted, all neurons are automatically bypassed since the entire register is reset. Otherwise, a single bit is clocked by a special clock 64 times. The register loads from the bottom up so that the first data loaded corresponds to the first row neuron. More details on the NN64, including its configuration as a recurrent neural network can be found in [11]. The chip was tested in a variety of applications where neural networks proved efficient. A particular application was interpretation of visual input data for automatic tracking of a path by a mobile robot. The photo of the chip, the experimental setup, and the robot tracking the path under the control of the neural chip, are illustrated in Figure 12.

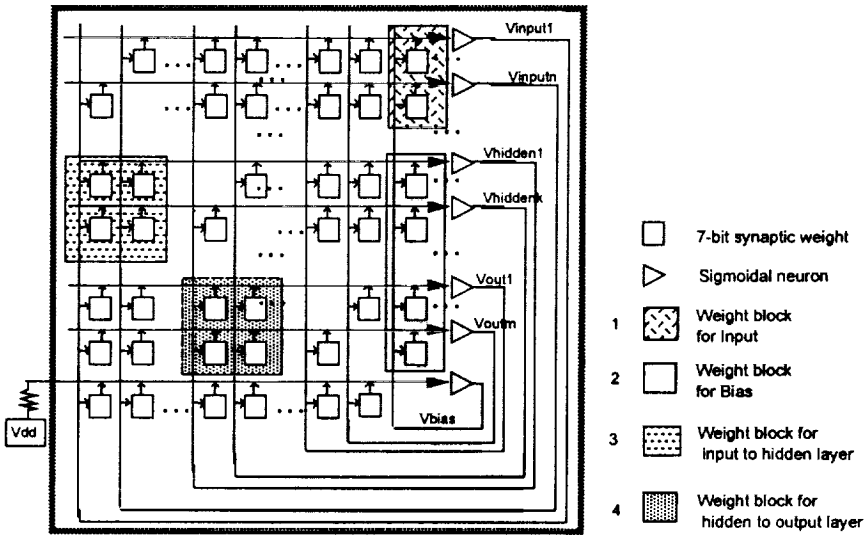

Figure 11. Diagram of a neuron-synapse composite chip configured as a feed-forward neural network.

Figure 12. Photo of the NN64 on test board, experimental set-up, and robot visually tracking a path under NN64 control

4.4 Integration of ELIPS components

Efforts are ongoing for testing the synergistic operation of ELIPS components before the final cut-off design. In this sense a test board is prepared to test a Hybrid Neuro Fuzzy Expert System (NFES). The board will allow 4 FES chips to be mounted on it, such that up to 36 rules can be programmed. In addition, the board will incorporate the design for the test of the neural network chips, with 2 NN64 chips and a group of 16 quad - A/D chips. The board aims to play multiple roles, allowing

- the test of the FES and NN64 chips individually,
- the test of the chips in tandem configuration, e.g. FES followed by NN64, etc.
- the test of the predictive algorithm in hardware, using the neural chips.

Ultimately, the ELIPS modules will be integrated to provide an intelligent processor on a chip. A sketch visualization of the final integrated product is illustrated in Figure 13.

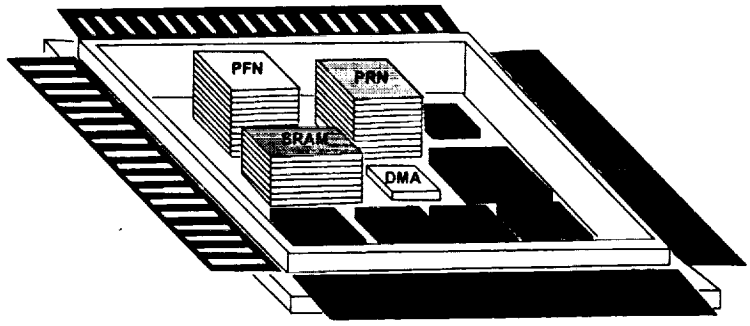

Figure 13. Visualization of a sensor fusion processor on a chip

CONCLUSIONS

Current technology allows the realization of a sensor fusion processor on a chip. A trade-off is to be made between the performance and cost of such a processor. Computational intelligence elements such as fuzzy reasoning and neural networks technology are considered fundamental for a sensor fusion chip. Several test chips implementing components of the ELIPS sensor fusion architecture have been fabricated in analog VLSI hardware and demonstrated processing times of the order of microsecond for a variety of tasks, such as target classification from preprocessed data.

ACKNOWLEDGEMENT

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Ballistic Missile Defense Organization through an agreement with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

REFERENCES

1. B. Figie, R. Linderman, Y. Kinashi, B. Johnson, and J. Fabunmi, "Discriminating Interceptor Technology Program (DITP): Sensor Fusion for Improved Interceptor Seekers," Presented at 1996 AIAA/BMDO Missile Sciences Conference, Session 8: Ballistic Missile Defense Interceptor Technology
2. B. V. Dasarathy and S. D. Townsend, "GIFTS – A Guide to Intelligent Technology Selection", Proc. Of the International Conference on Multisource-Multisensor Information Fusion, H. Arabnia and D. Zhu (Eds) Las Vegas NV, July 6-9, 1998, CSREA Press, 65-72.
3. Y. Xia and J. Wang, "Recurrent Neural Networks for Shortest-Path Routing" International Conference on Multisource-Multisensor Information Fusion, H. Arabnia and D. Zhu (Eds) Las Vegas NV, July 6-9, 1998, CSREA Press, 237-244
4. D. Zhu abd B. Zhang, "Fuzzy Sensor data Fusion in GPS Vehicle Positioning", International Conference on Multisource-Multisensor Information Fusion, H. Arabnia and D. Zhu (Eds) Las Vegas NV, July 6.-9, 1998, CSREA Press, 259-266.
5. http://www.ortech-engr.com/fuzzy/fcachip.html, FCA Chip
6. S. Guo, and L. Peters, "A High-Speed, Reconfigurable Fuzzy Logic Controller," IEEE Micro, December 1995
7. H. Huertas, S. Sanchez-Solano, I. Baturone, and A. Barriga, "Integrated Circuit Implementation of Fuzzy Controllers," IEEE Journal of Solid-State Circuits, Vol. 31, No. 7, July 1996
8. J. Fattaruso, S. S. Mahant-Shetti, J. B. Barton, "A Fuzzy Logic Inference Processor," IEICE Trans. Electron., Vol.E77-C, No. 5, May 1994
9. M. Sasaki, N. Ishikawa, F. Ueno, and T. Inoue, "Current-Mode Analog Fuzzy Hardware with Voltage Input Interface and Normalization Locked Loop," IEICE Trans. Fundamentals, Vol.E75-A, No. 6, June 1992
10. Eberhardt, S. et al, "Analog VLSI Neural Networks: Implementation Issues and Examples in Optimization and Supervised Learning," IEEE Trans. Indust. Electron. v39 (6):p. 552-564, Dec. 1992.
11. T. Duong, S. Eberhardt, T. Daud and A. Thakoor, "Learning in neural networks: VLSI implementation strategies", In Fuzzy Logic and Neural Networks Handbook. Fuzzy Logic and Neural Network Handbook, Ed: C.H. Chen, McGraw-Hill, 1995