In the proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007), Imperial College, London, UK 23-26 July, 2007

Parallel Type-2 Fuzzy Logic Co-Processors for Engine Management

Christopher Lynch, Hani Hagras *Senior Member, IEEE* and Victor Callaghan

***Abstract*— Marine diesel engines operate in highly dynamic and uncertain environments, hence they require robust and accurate speed controllers that can handle the encountered uncertainties. Type-2 Fuzzy Logic Controllers (FLCs) can handle such uncertainties; however there are a number of computational bottlenecks posing as significant barriers, preventing widespread deployment of type-2 FLCs in commercial embedded control systems. This paper explores the use of parallel hardware implementations of interval type-2 FLC as a means to eradicate these barriers producing bespoke co-processors for a soft core implementation of a FPGA based 32 bit RISC micro-processor. These co-processors will perform functions such as fuzzification and type reduction and are currently utilised as part of a larger embedded Interval Type-2 Fuzzy Engine Management System (T2FEMS). Numerous timing comparisons were undertaken between the co-processors and hard coded type reducers were the type-2 co-processors reduced the required computational cycles by 99.88 percent. Thus the co-processors enable us to fully explore the potential of interval and potentially general type-2 systems in applied commercial embedded applications.**

I. Introduction

Marine diesel engines operate in highly dynamic and uncertain environments subject to significant transient disturbances [1]. Also inherent in such an application is high levels of uncertainty associated with the sensors and actuators, both affected by diverse environmental conditions and calibration issues, whilst the former is mainly affected by high noise levels the latter is additionally influenced by wear and tear.

Fig.1. Viking 25

Current commercial engine management systems attempt to address these uncertainties through the use of averaging of sensor inputs and gain scheduled control algorithms such as the gain scheduled PID controller with numerous non-linear gain functions embedded in the Viking 25 industrial controller illustrated in figure 1. Despite the additional complexity of applying these supplementary functionalities, the Viking 25 is still computationally efficient requiring several hundred micro-seconds to perform all of its speed control functions, and using the remaining clock cycles to perform other engine management features such as signal conditioning, communications, alarm and monitoring etc. On the other hand, a type-2 FLC requires the equivalent number of clock cycles to perform type-reduction alone [1]. Thus despite any performance improvements type-2 FLC may offer, these computational bottlenecks remain as a barrier to the type-2 FLC deployment in commercial embedded control systems. As a result, an alternative solution which exploits the high level of parallelism offered by type-2 FLC was required with the constraint that the solution must also be backwards compatible with the existing Viking 25 core software written in C/C++. Such compatibility allows the portability amongst applications

Currently there are only two other hardware implementations of type-2 FLC available. The first implementation was presented in [2] and they produced a VLSI implementation were the type-2 FLC was designed at the transistor level on a single chip for a dual input single output type-2 FLC supporting up to 64 rules. This approach whilst offering a tailored solution does not offer the flexibility nor re-programmability of a micro-processor based solution. Alternatively Melgarejo et al [3] designed a type-2 FLC for an adaptive filter with a rule base of nine rules using the Wu-Mendel approximation to height type reduction. This implementation was embedded on a Field Programmable Gate Array (FPGA) which is a single chip programmable logic device. This approach is a highly optimised and pipelined solution offering a type reduced set in 9 clock cycles at the expense of being highly memory intensive; making use of memory base fuzzification, reciprocal division and distributed arithmetic each of which require a large amount of on chip memory. Although his approach is applicable to the higher end FPGAs, it is unsuitable for larger rule bases on lower cost FPGAs with less on chip memory.

This paper presents parallel hardware implementations of interval type-2 FLC for the purpose of control which can accommodate much larger rule bases than the previous hardware implementations mentioned above. This will create bespoke co-processors that can perform functions such as fuzzification and type reduction. The proposed system is currently utilised as part of a larger embedded Interval Type-2 Fuzzy Engine Management System (T2FEMS). Numerous timing comparisons were undertaken between the co-processors and equivalent sequential floating implementations were the type-2 co-processors reduced the

This work was supported by Regulateurs Europa Ltd.

Christopher Lynch, Hani Hagras and Victor Callaghan are with the Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, UK (phone: +44 (0) 1206 873333; email: clynch@essex.ac.uk).

required computational cycles by 99.88 percent.

This paper begins by introducing our hardware and the engine testing platform in section II. Section III discusses the hardware co-processor design. Section IV presents the experiments and results followed by the conclusions in Section V.

II. The used Hardware and Engine Testing Platforms

A. The Hardware Platform

The T2FEMS is embedded on a FPGA and exploits a soft core implementation of a 32-bit RISC Harvard architecture (the MicroBlaze) with 32 general purpose registers, Arithmetic and Logic Unit (ALU), and a rich instruction set optimised for embedded applications.

The MicroBlaze is a soft core developed by Xilinx (a manufacturer of FPGAs) implemented using the logic primitives of the FPGA with the key benefits of easy integration with the FPGA fabric while avoiding obsolescence [4]. The soft core based solution has several advantages over a pure hard core based design. Firstly, it supports a purely software based application development, allowing portability between FPGAs. Secondly, the soft core features do not require any silicon area when it is not needed, while the hard processor based approach always consumes the same area [4]. Of particular interest is the prospect of identifying bottlenecks in the embedded algorithms and replacing them with a customised soft core co-processor thus accelerating the performance of certain functional blocks [4]. These customised soft core co-processors communicate with the Micro-Blaze via a number of bus interfaces; two of which are widely exploited in the T2FEMS design which are the On-chip Peripheral Bus (OPB) and the Fast Simplex Link (FSL). The OPB is a Core-Connect IBM standard bus used for less time critical communications. Alternatively the FSL is a dedicated point-to-point data streaming interface providing a low latency interface to the processor pipeline. The OPB and the FSL allow for extending the processor's execution unit with custom hardware accelerators (co-processor) [4].

The MicroBlaze also supports development in C code using the Xilinx Embedded Development Kit (EDK) whilst the co-processors are developed using Xilinx ISE Foundation in VHDL (Very High Speed Hardware Description Language). The functional/timing simulations were performed in Mentor Graphics ModelSim.

All of the aforementioned co-processors and Micro-Blaze will be embedded in a Spartan 3E FPGA (shown in figure 2(a)) which is one of the lower cost per gate devices Xilinx produce and it is also targeted and rated for automotive applications with respect to temperature, packaging etc. Currently the functionally of both the co-processors and MicroBlaze is verified in both simulation with ModelSim and additionally through the use of a hardware development platform with an embedded Spartan 3E (XCS500E) with 64 MByte DDR SDRAM and 16 Mbit SPI Flash. Numerous communications interfaces exist including RS232, Ethernet and JTAG. The platform also has numerous digital inputs and outputs (I/O) and analogue I/O. Currently the T2FEMS is programmed into external flash before being boot-loaded into the FPGA and executed from external DDR SDRAM or alternatively internal block RAM.

(a) (b)

Fig. 2. (a) Spartan 3E FPGA [4]. (b) The testing platform.

B. The Engine Testing Platform

The embedded control algorithms are tested and verified on the engine testing platform shown in figure 2(b) which is designed to realistically reflect the characteristics and operating conditions of the marine diesel engines; with the ability to alter speed, load, inertia and torque. The platform uses the same noisy sensors and actuators used on the engine with the ability to introduce the same uncertainty levels encountered on the real engines.

The core functionality of the MicroBlaze is defined in c code whilst the co-processors are developed in VHDL and communicate with the MicroBlaze via the FSL or OPB interface bus. Unfortunately the restrictive page limit of this paper only allows for the type reduction co-processors to be discussed in detail.

III. Hardware Co-Processors

A. Fuzzification

Numerous fuzzification strategies have been exploited in the design of FPGA and VLSI based fuzzy controllers. The most common method is memory based fuzzification [3], were any arbitrary shaped Membership Function (MF) can be represented in memory by discretising its universe of discourse and storing the resultant degree of membership in memory, providing very fast fuzzification. The disadvantage of this method relates to the required resolution and level of discretisation possibly resulting in very large MF tables.

Mathematical approximations of membership functions [5], [6] are also widely used but any errors produced by crude approximations can be problematic for adaptive systems as a continuous function approximation may not prove to be continuous in all segments.

Analogue fuzzifiers have been developed by [7], [8], despite being very fast and power efficient analogue implementations are prone to temperature related drift and inaccuracies related to component tolerances.

Linear interpolation is applicable for both trapezoidal and

triangular MF fuzzification offering fast fuzzification with minimal resources.

We have implemented type-2 fuzzifiers employing trapezoidal and triangular MFs. The type-2 fuzzifiers were implemented as single FSL co-processor. In this implementation, the triangular MFs were considered a special case of the trapezoidal MFs. This co-processor makes use of linear interpolation producing both upper and lower fuzzified values in 12 clock cycles. Also developed are type-2 Gaussian fuzzifiers in c code for the MicroBlaze.

B. Rule Base

FPGA based rules bases are typically a binary pattern used to reference antecedents, consequents and the connective operator of each rule. The hardware realisation of the rules involves a number of multiplexers and memory elements as in [3]. The T2FEMS rule base will be defined in C code within the Micro-Blaze processor. Hard-coded VHDL implementation's were previously designed but prove a less flexible and maintainable solution. Also the rule base will form part of an adaptive type-2 FLC planned for future development making further use of the Micro-Blaze processor core.

C. Inference

A number of t-norm and t-conorm inference operators exist. One of the most common t-norms for embedded systems is the minimum t-norm due to its ease of implementation and lesser resources required compared to the product t-norm. Were the product t-norm is defined as a multiplier while the minimum t-norm needs only a simple magnitude comparator.

In retrospect, application of the minimum operator is not necessarily the correct choice for embedded control systems with more than two inputs [9], [10], as it creates non-linearities in the control surface. Additionally the MicroBlaze ALU is already defined in the FPGA resources and can perform a product operation in a single clock cycle thus product t-norm will be used throughout this paper.

D. Type-Reduction

The approach taken in this paper is to minimise the logic used by the co-processor sacrificing speed in favour of a reduced gate count and lower cost FPGA's. As with the fuzzifiers the type-reduction co-processor will interface directly to the Micro-Blaze via the FSL bus. The co-processor will support up to 32 fired rules with a resolution of 16 bits for both the firing intervals $[\overline{f}^i, \underline{f}^i]$ and consequent centroids $[w_l^i, w_r^i]$. Were possible single adders will be used for large summations (i.e. were $\sum_{i=1}^{M} \overline{f}^i$ s a single adder) thus requiring *M* clock cycles for the summation but utilising less FPGA resources than a completely parallel implementation requiring *M* adders and a single clock cycle.

Currently both the iterative Karnick-Mendel (KM) procedure for type-reduction [11] (employing the centre of sets type-reduction) and Wu-Mendel (WM) Uncertainty Bounds [12] method (approximating type-reduction) are supported and designed using VHDL as co-processors to the Micro-Blaze, denoted KM-CP and WM-CP respectively. Both approaches will now be defined in a manner more applicable to a FPGA based implementation.

1) KM Co-Processor

The KM procedure is typically defined as a four step iterative procedure [11]. This procedure will now be redefined with the removal of step 2 thus not requiring the *L* and *R* index variables used in [11]. The modified procedure is shown below for y_l.

Without loss of generality assume the w_l^i are arranged in ascending order: $w_l^1 \le w_l^2 .....w_l^M$

1. Compute y_l in Equation (1) by initially setting $f^i = (\underline{f}^i + \overline{f}^i)/2$ for *i=1…M* and set $y'_l \equiv y_l$
2. Compute y_l in Equation (1) with $f^i = \overline{f}^i$ for $w_l^i \le y'_l$ and $f^i = \underline{f}^i$ for $y'_l < w_l^i$ and set $y''_l = y_l$
3. If $y''_l = y'_l$ then set $y_l = y''_l$ and stop, otherwise let y'_l equal to y''_l and return to step 2

$$y_l = \frac{\sum_{i=1}^{M} f^i w_l^i}{\sum_{i=1}^{M} f^i} \qquad (1)$$

The procedure for y_r can also modified in a similar manner. This 3 step procedure for both y_l and y_r was analysed and segmented into a number of parallel processes and memory elements. Figure 3 depicts a graphical representation of the final VHDL implementation of KM-CP in the FPGA.

Figure 3 embeds a number of parallel processes '*P*' and a number of memory elements '*MEM' (32* bits wide each). Memory element *MEM1* stores the consequents $w_l^1 \le w_l^2 .....w_l^M$ and $w_r^1 \le w_r^2 .....w_r^M$ of the fired rules, whilst *MEM2* stores the firing interval $\underline{f}^i$ and $\overline{f}^i$. The firing interval and consequents of the fired rules will be passed to the type-reducer from the Micro-Blaze processor via the FSL bus as fixed point integers. Each firing strength is represented as a 16 bit unsigned value and similarly the consequent centroids are each 16 bits signed. As each FSL bus transfer is 32 bit it was deemed more efficient to reduce the number of FSL bus transfers by combining the centroid interval into a single 32 bit value, likewise with the firing interval. Therefore reducing the number of writes via the FSL bus to M (number of fired rules) writes for each memory element, thus it requires 2M FSL writes to transfer all the required data for type-reduction. An additional FSL write is also required to initiate the co-processor and define status information such as the number of fired rules etc. As

each FSL bus transfer requires two clock cycles it will take a total of (2M+1)*2 clock cycles to complete all FSL writes to the KM-CP.

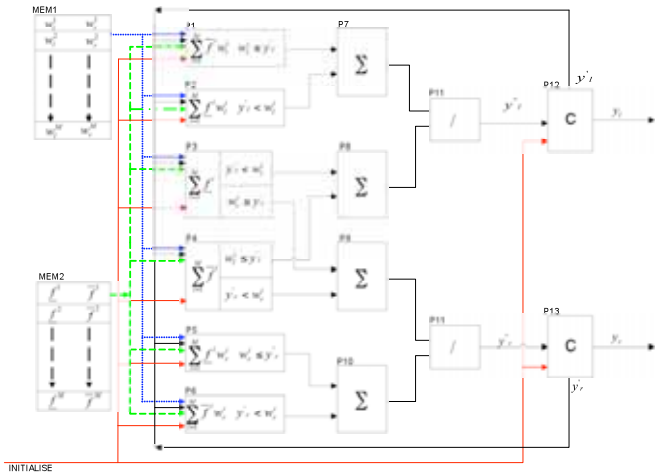


Fig.3. Graphical representation of KM-CP

The processes of figure 3 are represented as a number of summations (*P3* and *P4*), multiply and summations (*P1*, *P2*, *P5* and *P6*), additions *P7*, *P8*, *P9* and *P10*, divisions *P11* and finally comparisons *P12* and *P13*.

Each column of processes operates in parallel i.e. *P1* to *P6* (requires M clock cycles) function in parallel and their outputs are used as inputs to another column of parallel processes *P7* to *P10* (requires a single clock cycle).

The process *P11* (division) is included twice and represents a shared process i.e. the same hardware is used for both divisions. *P11* is a VHDL implementation of a pipelined radix-2 non-restoring signed integer divider requiring an initial 20 clock cycles for the first division and 1 additional clock cycle thereafter thus both *P11* processes will require a combined total of 21 clock cycles to complete.

The final column contains comparative processes *P12* and *P13* representing step three of the modified KM procedure, comparing $y''_l \equiv y'_l$ or $y''_r \equiv y'r$ respectively, if either comparison is false then y' is set equal to y'' and is fed back to the inputs of *P1-P6*, requiring only a single clock cycle to perform. However when both *P12* and *P13* are true then type reduction is complete and y_l, y_r are passed back to the Micro-Blaze processor via the FSL bus.

The additional signal "initialise" relates to the first iteration through the modified KM procedure. Therefore when the initialise signal is set, processes *P2*, *P3* and *P5* are disabled (thus outputting 0), additionally all f^i are set equal to $\underline{(f^i + \overline{f}^i)/2}$, also y'_l is initially set to a predefined maximum value whilst $y'r$ is set to a predefined minimal value. Finally the comparative processes *P12* and *P13* are also disabled simply setting y''_l equal to y'_l and returning the value to the inputs of *P1-P6*. After this first iteration the "initialise" signal changes state and the processes operate as normal. Finally when both comparisons are true the type reduced set is combined into a single 32 bit value and returned via the FSL bus to the Micro-Blaze requiring a further two clock cycles (FSL Read).

Table 1 defines the complete number of clock cycles required for each parallel column of figure 3.

TABLE I
KM-CP TYPE REDUCTION CLOCK CYCLES

	M1, M2	P1 to P6	P7 to P10	P11	P12 to P13	FSL Read
Clock Cycles	4M+2	M	1	21	1	2

The total number of clock cycles required by the KM-CP is greatly influenced by the number of fired rules and the number of iterations required to complete type-reduction and can be represented by the following formula:

$$4M + 27 + (M + 23) * N \qquad (2)$$

Were N is the number of iterations required by the KM iterative procedure (not including the initialisation iteration). In the following section a similar analysis and is carried out for the WM-CP.

2) WM Co-Processor

The Wu Mendel Boundary equations provide mathematical formulas for the inner and outer bound sets which can be found in [12]. Analysis of these equations will reveal the WM approach makes use of a larger number of arithmetic operators than the KM-CP thus requiring more FPGA resources, but has the added advantage of not being an iterative process thus does not require any large local memory elements as in the KM-CP.

Figure 4 depicts a graphical representation of the final VHDL implementation of WM-CP in the FPGA. Again each column of processes operates in parallel, were *P1* to *P4* are multiply and summation, *P5* and *P6* summations, *P7* summation and a subtraction, finally *P8* to *P11* are multiply and summation with a subtraction.

Processes *P1* to *P11* operate during the FSL writes to the co-processor requiring 2M FSL bus transfers to complete. As with the KM-CP there is additional FSL transfer containing status information, thus *P1* to *P11* require a total of 4M+2 clock cycles. Also the first few FSL writes are reserved for the centroid values w_l^1, w_l^M, w_r^1 and w_r^M required by processes *P8* to *P11* and thus stored as registered values.

The remaining processes are clearly marked as divisions, product, summation or the min and max comparators required by the WM equations. Were each parallel column of processes *P12-P16*, *P18* to *P21* and *P22* to *P23* require a single clock cycle each to complete.

As in the KM implementation the divider (*P17*) is shared amongst all the processes, making used of the same radix-2 non-restoring signed integer divider that the KM-CP used requiring a combined total of 26 clock cycles to complete all

divisions. Table 2 defines the complete number of clock cycles required for each parallel column of figure 4.

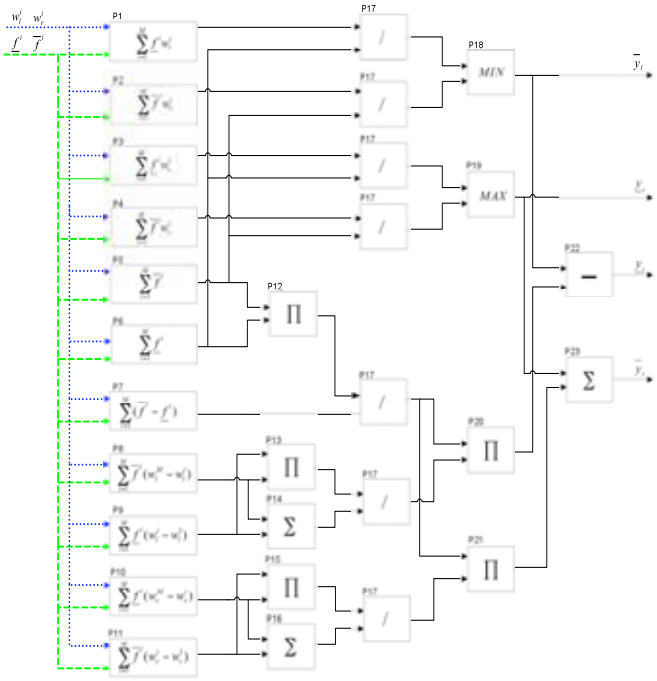


Fig.4. Graphical representation of WM-CP

TABLE II
WM-CP TYPE REDUCTION CLOCK CYCLES

	P1 to P11	P12 to P16	P17	P18 to P21	P22, P23	FSL Read
Clock Cycles	4M+2	1	26	1	1	2

The total number of clock cycles required by the WM type-reduction block is only influenced by the number of fired rules, thus a basic formula representing the total number of clock cycles required for the WM-CP is:

$$4M + 33 \quad (3)$$

Thus if 20 rules fired the WM method of type reduction would require 113 clock cycles to compute the values required by the MicroBlaze for defuzzification whilst the KM-CP would require $107 + (43) * N$, thus if the KM procedure required one iteration (not including the first iteration) it would be 37 clock cycles slower than the WM implementation, otherwise for two iterations it would be 80 clock cycles slower and for 3 iterations it would be 123 clock cycles slower.

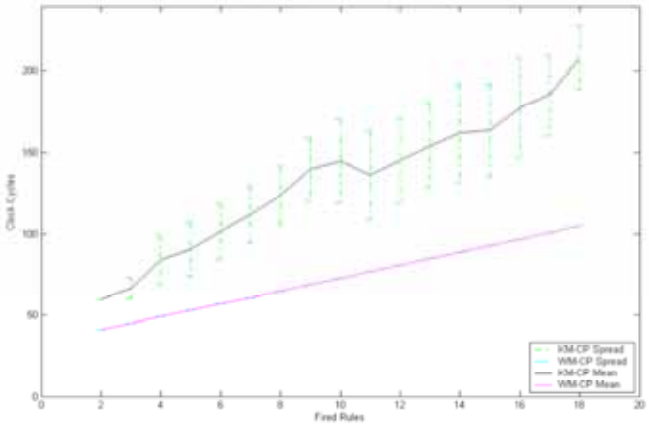
E. Defuzzification

Whilst defuzzification could also be performed in the type reduction co-processors this function is currently performed in the MicroBlaze processor were defuzzification for both the KM and WM methods is easily achieved by multiple logical shift operations in the MicroBlaze requiring a minimal computational effort.

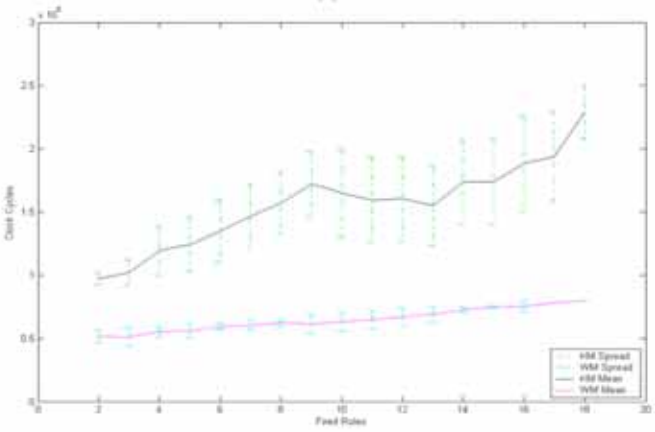
IV. EXPERIMENTS AND RESULTS

A. Computational Comparison

In this subsection, we will introduce a comparison between the computational times of the KM-CP and WM-CP. A type-2 FLC was coded in the MicroBlaze processor in C, including Gaussian type-2 fuzzification and the rule base previously used in a similar timing analysis in [1]. The firing strengths and centroids of the rule consequents were calculated in the MicroBlaze before being passed to the co-processors via the FSL bus.



(a)



(b)

Fig. 5. (a) KM-CP and WM-CP (b) MicroBlaze KM and WM

Figure 5(a) illustrates the final results of this timing analysis and shows the linear relationship between the WM-CP to the number of fired rules, a very advantageous trait as its consistency and predictability allow the MicoBlaze to transfer all data to the WM-CP and then continue executing other code, rather than waiting for the WM-CP to return the type reduced. Conversely the KM-CP is dominated by the number of fired rules and the number of iterations required to complete type reduction, such that it is difficult to predict in advance the total clock cycles the KM-CP will require, as the number of required iterations is unknown. Although the minimum number of clock cycles (at least 1 iteration) can be known allowing the MicroBlaze a minimal window within

which to perform other functions. Figure 5(a) also reveals that the WM-CP required 44.72 percent less clock cycles relative to the KM-CP in the instance of 14 fired rules, a trait clearly reflected across the range of fired rules.

Figure 5(b) illustrates a sequential floating point implementation of the KM and WM type reducers implemented in C on the MicroBlaze (executed from external memory), subjected to the same tests data as the co-processors.

In the instance of 14 fired rules the KM-CP achieved a 99.88 percent reduction in the number of clock cycles required for type reduction compared to the equivalent floating point implementation. Whilst the WM-CP for the same number of fired rules achieved a 99.91 percent decrease compared to its equivalent floating point implementation. Thus the hardware acceleration offered by the co-processors removes any significant bottlenecks from type-2 FLC and consequently identifies a future platform for general type-2 FLC to achieve similar performance advantages.

B. *Co-Processors Comparison*

TABLE III
FPGA RESOURCES AND COMPUTATIONAL DEMANDS OF CO-PROCESSORS

	MICROBLAZE	FSL KM	FSL WM	DIVIDER
SLICES	1728	562	600	559
FREQUENCY	52MHZ	76MHZ	65MHZ	122MHZ

Table 1. defines the number of FPGA resources expressed in slices (an FPGA is defined by an array of configurable logic blocks each having a predefined number of logic slices) that each component uses and also the maximum frequency of operation expressed in Hertz. The number of slices required by the type reduction co-processors has been defined without the divider as this is currently a highly resource intensive component. Alternative methods of division such as a reciprocal divider or a slower implementation of the current solution may prove more efficient. Both the KM-CP and WM-CP achieve similar maximum frequencies and total slices used with the WM-CP having a slightly lesser maximum frequency due to slower combinational paths e.g. *P8* is a multiplication and a subtraction. Overall the co-processors and MicroBlaze use little of the 4,656 slices available in the Spartan 3E (XCS500E).

V. CONCLUSIONS

In this paper, we presented a parallel implementation of both Wu-Mendel and Karnick Mendel approaches to the centre of sets type reduction. Both implementations were defined in VHDL and operate as co-processors to a 32 bit soft core micro-processor. The co-processors communicated over the FSL bus to the MicroBlaze performing type-reduction in parallel realising reductions in clock cycles of 99.88 and 99.91 percent for the KM-CP and WM-CP respectively (compared to an equivalent sequential floating point implementation).

Timing analysis also compared the WM-CP and KM-CP for the same number of fired rules, were the WM-CP required 44.72 percent less clock cycles than the KM-CP. Also the WM-CP offers predictable timing enabling the MicroBlaze to predict a fixed window within which it can execute other tasks.

The complete T2FEMS implementation of the Type-2 FLC with FSL fuzzification and type reduction co-processors would certainly now be comparable to a sequential implementation of a type-1 FLC. The performance advantages of this type of implementation also reveal new prospects for the commercial application of general type-2 FLC and present an exciting future for applied embedded type-2 systems. We are currently working towards increasing the maximum operational frequency and reducing the number of required slices for all co-processors.

REFERENCES

[1] C. Lynch, H. Hagras and V. Callaghan, “Using Uncertainty Bounds in the Design of an Embedded Real-Time Type-2 Neuro-Fuzzy Speed Controller for Marine Diesel Engines” in *Proc. of the WCCI 2006*, Vancouver, Canada, 2006

[2] Shih-Hsu Huang and Yi-Rung Chen, “VLSI implementation of type-2 fuzzy inference processor”, Proceedings of tthe IEEE International Symposium on Circuits and Systems, 2005, pp. 3307- 3310, Vol. 4, 23-26 May 2005

[3] M.Melgarejo, A. Garcia, C.A. Pena-Reyes, “Pro-Two: A hardware based platform for real time type-2 fuzzy inference”, Proceedings IEEE International Conference on Fuzzy Systems, Vol. 2, pp. 977-982, 25-29 July 2004

[4] Xilinx (2006, June 01). “MicroBlaze Processor Reference Guide” [Online]. Available: http://www.xilinx.com

[5] J.J. Blake, L.P. Maguire, T.M. McGinnity, B. Roche, L.J. McDaid, “The implementation of fuzzy systems, neural networks and fuzzy neural networks using FPGAs”, Information Sciences, Volume 112, Number 1, December 1998, pp. 151-168

[6] D.J. Myers, and G. Storti-Gajani, "Efficient Implementation of Piecewise Linear Activation Function for Digital VLSI Neural Networks," *Electronics Letter*, vol. 25, no. 24, pp. 1,662-1,663, Nov., 1989.

[7] McDaid L.J.; McGinnity T.M.; Maguire L.P “Hardware Implementation of a Membership Function Generator for Fuzzy Reasoning”, Information Sciences, Volume 96, Number 1, January 1997, pp. 93-105(13)

[8] Djuro G.Zrilic, Jaime Ramirez-Angulo, Bo Yuan, “Hardware implementations of fuzzy membership functions, operations and inference”, Computers and Electrical Engineering, March 1998

[9] R. Jager, Fuzzy Logic in Control, PhD thesis, Technische Universiteit Delft,June 1995.

[10] Pauli Viljamaa, “Fuzzy Gain Scheduling and Tuning of Multivariable Fuzzy Control—Methods of Fuzzy Computing in Control Systems”, PhD thesis, Tampere Univ. of Technology, Publications 293, 2002

[11] J.Mendel, "Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New directions," Upper Saddle River, NJ: Prentice-Hall, 2001.

[12] H. Wu and J. Mendel, “Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems,” *IEEE Trans. on Fuzzy Systems*, vol.10, pp. 622-639, October 2002.