Implementing Embedded Expert Systems via Programmable Hardware

Mária Pohronská*
Institute of Computer Systems and Networks
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 3, 842 16 Bratislava, Slovakia
pohronska@fiit.stuba.sk

Abstract

The work deals with intelligent embedded systems, particularly with the problem of application of expert systems in embedded architectures. It summarizes the state of art and challenges in areas of embedded systems and rule-based expert systems, and gives motivations for implementing expert systems in embedded architectures.

We design architecture of expert system and hardware architecture of embedded system suitable for implementation of embedded expert systems. We also devise a universal representation for knowledge bases of embedded expert systems. We propose two methods of hardware acceleration of inference in embedded expert systems. One of the devised methods we experimentally evaluate and claim its remarkable contribution to inference process of expert systems and its suitability for utilization in embedded expert systems. Based on the performed experiments and acquired experience we synthesize a set of rules for implementation of expert systems in embedded architectures which contribute to the problem area of intelligent embedded systems development.

The devised method for hardware accelerated inference enables implementation of expert systems even in embedded architectures where it has not been possible with the current state of art, thus facilitating further adoption of intelligent embedded systems.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representations (procedural and rule-based)*; I.2.5 [**Artificial Intelligence**]: Programming Languages and Software—*Expert system tools and techniques*

Keywords

embedded systems, expert systems, hardware acceleration, programmable hardware, real-time systems, intelligent systems

1. Introduction

Embedded computer systems have become exceptionally important in multiple domains that affect our daily life. Statistics show that embedded microprocessors account for more than 98 percent of all produced and sold microprocessors - vastly surpassing computing power in the IT industry [8]. A particular growth of their utilization has developed in the area of smart card computing, production control, automotive industry, network applications, in medical applications and in personal handheld appliances.

Resulting from their designation, embedded systems have tough restrictions to their physical size, power consumption, thermal emission and other attributes. As a result of these restrictions, they usually dispose with limited computational power, program and operating memory and operate at low frequencies. They mostly employ specific processors that are equipped with sophisticated instruction set tailored for their task.

Besides these technical considerations, the characteristic and most challenging features of embedded systems are connected with their strictly stated demands on reliability and response time. As their small computational power and the demand for fast and (particularly) guaranteed response time are quite contradictory, the task of designing reliable embedded real-time systems remains open problem since their introduction.

For the task of advanced diagnosis, monitoring, and control in various areas including production systems [10], vehicle control, power plant control [22], medical applications[6] and many others, it is desirable to perform knowledge-based computation. Knowledge-based systems are also used in decision support applications which are not restricted only to implementation on personal com-

*Recommended by thesis supervisor: Assoc. Prof. Tibor Krajčovič. Defended at Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava on May 22, 2012.

Pohronská, M. Implementing Embedded Expert Systems via Programmable Hardware. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 4, No. 2 (2012) 10-19

puters but are also deployed on embedded and even handheld devices [20]. Deployment of decision support systems on handheld devices has particularly promising perspective; as such systems can be very helpful in numerous problem areas and have wide practical use.

However, implementation of knowledge-based systems in embedded and real-time systems is nontrivial task which has been addressed since the introduction of the embedded computing concept. Angeli [1] states that for the real-time control applications it is more appropriate to use rule-based systems than model-based systems. Implementation of knowledge-based control and reasoning in embedded systems is currently achieved by application of fuzzy control systems and expert systems adjusted for the embedded environment. Both of these approaches need to store and process the decision rules and utilize the data from environment. As searching the rule base and performing the inference process are computationally intensive tasks, there is a necessity to optimize these systems for employment in embedded computing.

In our work, we deal with rule-based expert systems and possibilities of their implementation in embedded architectures and real-time systems. We discuss embedded and expert systems' properties and architectures to address their suitability for building embedded expert systems and propose a representation of knowledge suitable for utilization in these systems. We devise two methods of utilizing programmable hardware for acceleration of expert system inference process and propose a set of rules forming a method for implementation of embedded expert systems.

2. Related work

In embedded systems problem domain, many problems remain open and are addressed by research activities. These include the problem of advanced scheduling, maximizing the computing power of the system, e.g. [3]; minimizing the system's power consumption, ensuring the stability and robustness of the system, e.g. [5] and many other challenges which are connected with the embedded system design and implementation, including the problem domain of hardware-software co-design.

In the domain of real-time systems, the mostly addressed research tasks are connected with scheduling and include the improvement of currently known scheduling methods, both offline and online program analysis and inspection and development of methods for benchmarking and comparison of scheduling algorithms. Determination of the worst-case execution time of the program or its part is the most challenging task in designing real-time systems and must be given high priority in the process of their development [14].

The problem of implementing rule-based systems in embedded and real-time systems has been firstly compiled by Laffey [13]. Afterwards, the works [4] and [15] provided improvement in this area by proposing the methods for knowledge-base pre-compilation and parallelization of the inference process. The recent works focus mostly on development of specialized proprietary systems, e.g. [7]. Another challenging task in the domain of rule-based systems has been the knowledge acquisition and its transformation in the set of rules in the system's representation [12]. As this task requires cooperation with human experts, it is a serious bottleneck in designing each new expert system and has to be given appropriate attention even in small-scale systems.

The problem of accelerating knowledge and data-based systems has been addressed by several authors, however the proposed approaches are highly diverse and no common methodology has been developed. We consider the lack of a common approach to the problem of implementing expert systems in embedded and real-time systems the main limiting factor of the further development in this area.

The most common approaches to knowledge and data-based systems acceleration are based on software optimization of data storage methods, searching and processing algorithms, e.g. the Rete algorithm by C. Forgy [9] and caching techniques. Authors of [2] describe a technique and architectures for hardware acceleration of database operations using content-addressable memories. However, as their approach is oriented on characteristic database operations, it is unemployable in our area of interest.

Several works have addressed the problem of implementing knowledge-based systems in real-time and embedded systems. In [4] authors present an expert system in which the knowledge base is automatically precompiled, parallelized and optimized for the real-time behaviour. Systems presented in [15] and [19] come with approach of compiling the knowledge base into the *AND/OR* network. Both works focus on careful knowledge base preparation, in order to avoid redundant and problematic parts and optimize the search space. The common drawback of these approaches is that the knowledge base cannot be updated during the system's operation.

The work [11] presents specialized hardware architectures for the realization of a simple inference engine. The architectures are focused on the problem of detecting input events and finding corresponding rules in the system. The work states this being effectively a pattern matching problem. The presented system is simplistic, as it provides only one output bit. However, authors claim that the developed architectures are sufficient for their particular application.

In the work [21], author presents implementation of an embedded expert system that utilizes the *Lernmatrix* data structure for the knowledge base representation. This structure is mentioned as a special neural network structure. In fact, it can be reduced to a simple associative memory structure. The implementation of the knowledge base by the associative data structure leads to very short and particularly, linear time of search and reasoning.

Several commercial knowledge-based system platforms that are aimed for implementation in real-time systems exist, nevertheless, we have not encountered with a commercial platform aimed for embedded implementation.

3. Embedded expert systems architectures

This section deals with the architectures of embedded and expert systems suitable for implementation of embedded expert systems. Firstly, we select architectures of embedded systems that are theoretically capable of performing rule-based reasoning. Secondly, we propose architecture

of expert system aimed for embedded implementation. Selection of these architectures is the essential point in our work, as it forms a base for its further development.

3.1 Embedded architectures

We have dealt with the analysis of embedded architectures suitable for implementation of embedded expert systems in one of our previous works [17]. From the available and commonly used architectures of embedded systems, we have selected two architectures that are eligible for implementing the computation and data-hungry expert inference process.

The first selected is the **architecture with a powerful universal processor**. Main features of this architecture are processor capable of running an operating system and relatively large amount of program and operational memory. Such architecture is being used mostly in user-oriented embedded systems focused on performance, user interface and applications.

The most straightforward solution for implementing expert systems in such architectures is utilization of an existing expert system shell, e.g. CLIPS[1]. Apart from interpreting the expert system commands in real-time, CLIPS offers pre-compilation of the expert system to a self-standing application which performs the expert system functionality much more effectively. Program memory demands of such pre-compiled expert system are in hundreds of kilobytes and more [17]. It is clear that this approach is not suitable for simpler, less powerful or more specialized embedded systems that do not provide mechanisms for standard application deployment and enough hardware resources for the demanding expert system computations.

The second selected architecture suitable for implementing embedded expert systems is **architecture with programmable hardware working as an autonomous unit**. Utilization of programmable hardware for implementation of demanding computations in expert systems appears to be a promising possibility of realization of embedded expert systems. The programmable hardware is intended to serve as an autonomous computation unit, loosely coupled with the system processor. On contrary to tightly coupled configurable architectures which advantage of implementing programmable hardware directly on the processor chip, the loosely coupled configurable architectures employ programmable hardware connected with the processor via external input/output pins. Tightly coupled architectures are usually aimed for computations requiring frequent communication with the processor and generally provide low logic capacity. As we need to perform the demanding computations autonomously and with minimum amount of communication, the loosely coupled architecture suits our needs closely. Another advantage of utilization of programmable hardware in embedded architectures stems from the programmable hardware's versatility that allows the developer to customize the design and makes the architecture more universal.

The connection between the programmable hardware and the processor can be realized in several ways, to choose the most suitable option we have to consider data throughput, latency of communication and overall embedded system specifications. Considerable connection options are:

- Direct connection to the processor in a co-processor mode
- Connection via the parallel system bus
- Direct memory access
- Connection via a serial bus

Connecting programmable hardware directly to the processor provides communication with low latency and (depending on number of reserved connections) high data throughput. Drawbacks of this approach are high demands on hardware resources and service resources.

Connection via parallel bus is simple in terms of service and provides high data throughput. It is also the most universal method in the terms of utilization of programmable hardware for multiple purposes. The drawbacks of this approach are relatively high hardware resource consumption and high communication latency.

The direct memory access option is suitable for implementing computational-intensive algorithms that process large amount of data, without need of the assistance from the processor. However, with this approach it is impossible to guarantee the response time of the system.

Connecting the programmable hardware via serial bus is the most simple and least expensive method that saves hardware resources and has low service demands. However the high latency and low data throughput make it unsuitable for most hardware acceleration tasks.

3.2 Expert system architecture

Implementing expert systems in embedded architecture is connected with various problems that make the spread of embedded expert systems practically impossible. For enabling the further development in this area, it is firstly needed to specify the expert system architecture that is suitable for use in embedded expert systems. Within our work we have devised such specification which not only suits our needs but is also utilizable generally in the process of embedded expert systems' development. We have defined an expert system suitable for embedded deployment to be a software system formed of:

- facts and rules that make a knowledge-base of the system
- mechanisms of computation of the system's outputs according to the knowledge-base rules
- mechanisms for transforming the system's inputs and outputs to expert system facts and vice-versa
- system resources for hardware service
- user interface modules
- mechanisms for providing security and response time guarantees.

[1]CLIPS is probably the currently most popular expert system shell. It is an open source shell written in C. Project homepage: http://clipsrules.sourceforge.net/

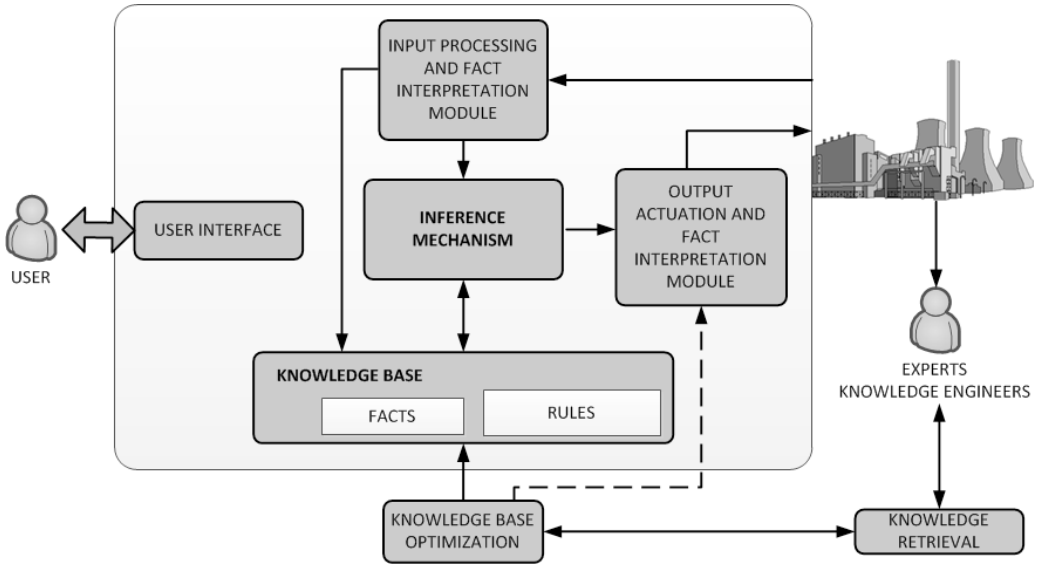

Figure 1: Architecture of an expert system suitable for deployment in embedded expert systems.

We consider the system to be a *rule-based system* , with strictly stated demands on the knowledge-base. The existing approaches in this area generally utilize a tree formed of *AND/OR* rules as their knowledge-base, as the form of tree provides the means to guarantee that the computation will end in finite time. The common utilization of facts that gain *yes/no* values simplifies the implementation of the inference mechanism and divides the problems of input processing and the expert system inference. Regarding these reasons and to sustain the continuity with previous works we have decided also to use this form of knowledge-base. The knowledge-base therefore has to be optimized before its employment in embedded expert systems. The proposed architecture of an expert system suitable for deployment in embedded expert systems is shown in Figure 1.

Standard expert systems are mostly focused on user interaction and presentation of knowledge. On contrary, embedded expert systems are focused on the environment and the controlled process, with the need of periodical operation and strictly stated demands on their response time. The proposed system is directly connected to the controlled process and it collects most from its inputs from the environment. The user interface serves mainly as an information screen or for serving the operator inputs. The proposed architecture also lacks the reasoning explanation module and module of knowledge retrieval from user, which are commonly present in regular expert system architectures.

Typical operation of the proposed expert system is periodical and operates with stated frequency, so it is mandatory to finish all computations and provide corresponding outputs within the repetition interval. It is thus a real-time system (possibly also a hard real-time system) which performs these operations in each cycle:

1. Processing the system inputs and transforming the values to representation of expert system facts.

2. The active facts are evaluated and fire a set of rules which eventually activate other system facts.

3. Evaluation of rules is repeated until the inference process gains stable state (no new facts are activated).

4. 4. The inference outputs are transformed to actual system outputs (e.g. indication, control, alarm).

4. Knowledge base for embedded expert systems

This section describes the proposed specifications of the knowledge bases suitable for utilization in embedded expert systems. Within our work we have also developed a universal XML scheme for representation of knowledge in the form of rules which is described further in this section. This paragraph is followed by definitions of key knowledge-base parts.

A **fact** in the knowledge base represents value of one monitored system state. It consists from natural language description of the monitored value and from the value itself. The allowed values are *yes/no*.

A **rule** in the knowledge base is defined as an *if-then* rule with a logic term consisting of available facts and any combination of logic operators *NOT, AND, OR, XOR, NAND, NOR, XNOR* on the right side; and a set of output facts with assigned values on the right side. Each rule can be expressed by a corresponding Boolean function.

The **knowledge base** of the embedded expert system is comprised of a set of facts and rules and has to satisfy these requirements:

- The form of the knowledge base is a tree, i.e. it cannot contain loops.
- One output fact is allowed to be modified by only one rule.
- The knowledge base must contain all facts, including those potentially emerging during the system's operation.

An example rule of the knowledge base is shown in the Figure 2. For the sake of facilitating knowledge base re-

usage and sharing and for simplifying its creation, modification and understanding, we have developed a universal XML schema for expressing embedded expert system knowledge. Next paragraph shows an example rule in this representation.

```
<rule>
<and>
 <condition GT="" LT="" EQ="" YN="YES">
  <objective>Building Control System ON
  </objective>
 </condition>
 <condition GT="3" LT="" EQ="" YN="YES">
  <objective>Secure room opened Timer
  </objective>
 </condition>
 <condition GT="" LT="" EQ="" YN="YES">
  <objective>Secure Door opened sensor
  </objective>
 </condition>
</and>
<output YN="Yes">
 Secure Door opened Alarm (Audible alarm)
</output>
Audible alarm:  Secure Door opened Alarm
</rule>
```

The original knowledge base describing the desired problem naturally does not comply with the strict requirements of the embedded expert systems. After the creation of the original knowledge base, the process of its optimization follows, which consists mainly of:

- transformation of original rules into the required form consisting of allowed logical operations,
- removing the loops from the original knowledge base,
- compaction of rules, to ensure that an output fact is modified by only one rule,
- detection of all potentially emerging facts and their insertion into the knowledge base,
- dividing the originally described facts into: 1. Formula for determining the fact value; 2. Name and value of the fact.

Determination of the fact values according to the given formulas is provided by *fact interpretation modules* and can itself be a demanding task for the embedded system. The *inference module* provides rule evaluation and works with the provided fact names and values. As interpretation and inference processes are of different nature and both of them can induce significant load to the system, we consider diving these two tasks into separate modules appropriate.

5. Accelerating embedded expert systems by means of programmable hardware

We have proposed two approaches to providing hardware assistance in embedded expert systems, with the utilization of programmable hardware. The motivation was to:

- accelerate the expert system inference process,
- unload the processor from demanding computations,

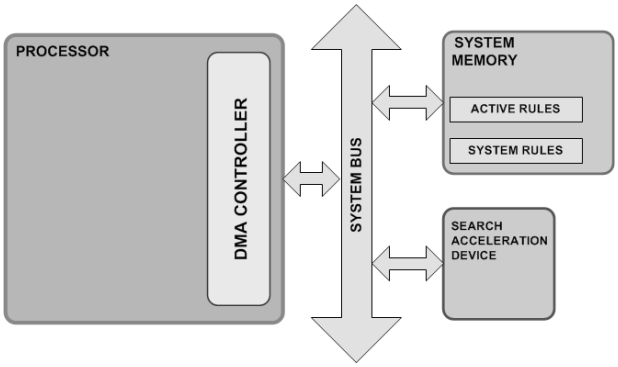

Figure 3: Embedded system with pattern matching hardware.

- extend the facilities of response time estimation,
- enable fulfilling system's response time guarantees.

5.1 Pattern matching assistance

The performed analysis shows that the most demanding task in the expert system inference process is rule-base searching and detection of potentially active rules (rules that use recently updated facts). Acceleration of this process can significantly contribute to the system performance and unload the processor from this demanding task. This approach has been described by [11] where authors developed a simple hardware architecture to perform pattern matching. Application of similar, more complex architectures has potential to contribute to expert system inference process, mainly in systems with extensive knowledge-bases.

We have designed a hardware architecture for performing the pattern matching task, which is fully described in [16]. The architecture exploits DMA access to perform the pattern matching process independently from the processor. The block architecture illustrating device's operation is shown in Figure 3. Typical operation of the system is then:

1. Load knowledge-base data (in specified format) to memory.
2. Evaluate facts and create list of recently updated facts.
3. Initialize the pattern matching device (include information on memory section containing data, memory section reserved for results, the searched pattern).
4. Enable the pattern matching operation.
5. Perform other system tasks. When the pattern matching process is finished (signaled by an interrupt) read the values and optionally return to 3.
6. Optionally return to 2.

The pattern matching acceleration is applicable in embedded systems providing enough memory, a DMA unit and memory management mechanisms. Its main purpose is to unload the processor from the time-consuming task of searching the memory. Following paragraphs summarize the positive and negative aspects of the approach. Positive aspects are:

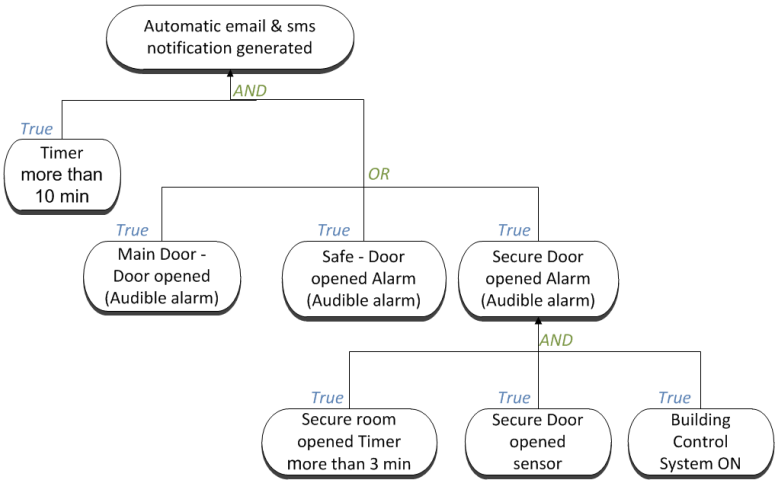

Figure 2: A sample rule of an embedded expert system.

- Unloading the processor of the demanding task of pattern matching.
- The hardware architecture is common to all embedded expert systems, providing that the memory representation of rules will not change.
- Utilization of the acceleration hardware does not put any limits to the knowledge base.
- The developed concept and architecture are universal and usable also in non-embedded applications.

The negative aspects are:

- Need for additional hardware in the embedded system.
- Necessity of the DMA unit and memory management mechanisms.
- Implementation of the DMA access controller requires complex architecture to be implemented on the side of programmable hardware.
- Utilization of the DMA access requires development of more complex software drivers on the side of the master system.

5.2 Hardware implementation of rules

Another proposed approach to inference acceleration is implementing the whole reasoning process in hardware. The acceleration hardware takes values of all system facts, evaluates the system rules and provides the resulting system state - values of all facts after the inference. It is connected to the processor via parallel system bus as a peripheral device; it also utilizes a dedicated hardware connection to one of the processor's pins to provide the interrupt signal. Operation of the accelerated system is illustrated in the Figure 4 and described in the following paragraph.

1. To initialize the inference computation, all fact values have to be written into the device. Depending on number of facts, several bus write cycles are needed to transfer all values.

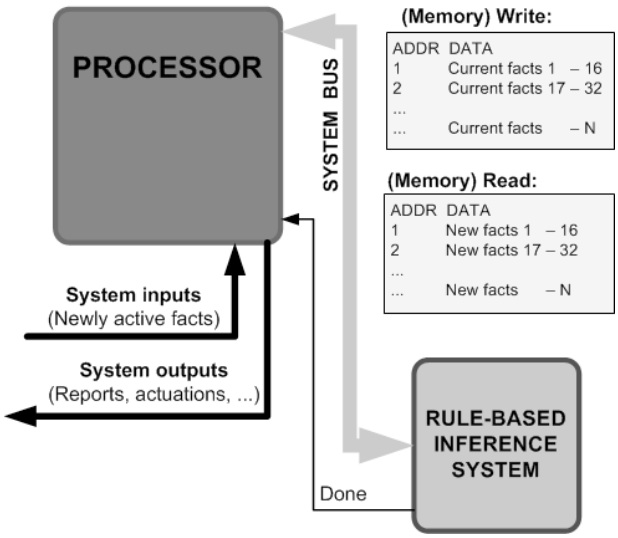

Figure 4: Hardware inference module - connection and communication scheme.

2. After last portion of data has been written, the acceleration device applies the rules and computes next inference state. The inference process continues until there are no new rules fired and values of output facts do not change.
3. After the inference has been completed, the *Done* signal (serving as system interrupt) is activated.
4. Active signal of the interrupt (*Done*) informs the system on the successful completion of the inference. The system can then read the results from the device.
5. The system reads the results from the device. Depending on number of facts, it may take several bus read cycles.
6. The acceleration device is ready to receive new data.

The inference process itself is performed by a combinational logic inside the programmable circuit, which is capable of computing one inference iteration at a time. The

process is controlled by a control unit which decides whether the computation has finished or there are more iterations needed. Depending on the knowledge-base structure and the current state of the system, it can take different number of iterations to finish the inference process. While it is difficult to determine the actual number of iterations needed, it is possible to state maximum number of inference iterations which equals to the maximum depth of the knowledge-base tree. It is thus possible to state maximum computation time of the inference.

The rules are implemented by a simple combinational logic consistent with the knowledge-base. Following paragraph shows an example of rules implemented in VHDL.

```
fact_outputs(2) <= (fact_inputs(27) or
  fact_inputs(28)); --- Battery fault
fact_outputs(1) <= (fact_inputs(2) and
  fact_inputs(29)); --- Battery alarm
```

Implementation of rules via dedicated combinational logic elements enables parallel and particularly, fast rule evaluation. The actual time needed to perform an iteration of the inference process is thus defined by overall latency of the combinational logic and depends on the rules complexity and the programmable hardware technology limitations. The overall inference computation time depends on this limitation, duration of the read/write cycles and the number of inference iterations. The inference computational complexity can thus be supposed linear, dependent on the number of iterations.

Implementation of the whole inference process via programmable hardware can lead to significant acceleration of computations. However, this approach is only applicable with utilization of the restricted knowledge base format (specified in the previous section) and is also restricted by the programmable device's capacity. Positive aspects of the approach are:

- The hardware inference computation is likely to perform much faster than the software approach.
- The main processor is completely unloaded from the inference computations.
- The hardware inference device requires only minimum system overhead.
- Amount of transferred data between processor and the device is much lower than amount of transferred data between the processor and memory when using the software approach.
- Re-configurability of the programmable hardware allows us to change the knowledge base of the system.
- Utilization of programmable hardware facilitates implementation of various dependability solutions, e.g. implementing redundant inference units.

Negative aspects of the approach are:

- Need for additional hardware in the embedded system.
- Connecting the reconfigurable device to the system requires additional hardware and software overhead.
- The knowledge-base cannot be modified during the system's operation.

6. Transforming knowledge-base to hardware

As can be understood from the previous section, the hardware implemented inference acceleration has potential to significantly contribute to the embedded expert system reasoning process. However, the reasoning hardware architecture is specific and differs for each knowledge-base. Thus it is necessary to create new hardware architecture for each knowledge-base and recompile it each time the knowledge-base is changed. As this is rather demanding process that involves knowledge of the hardware architecture and actual work of a hardware designer, it would demotivate the usage of the proposed approach even when the acceleration would be remarkable.

The aim of our work was to provide new means to implement embedded expert systems, even in the architectures and environments where it hasn't been possible before. The proposed hardware implemented inference approach appears to be capable of such contribution; however the demanding process of the architecture creation is ineffective and would discourage developers from exploiting this approach.

With this motivation we have developed an automated translation tool *Knowledge base parser* that transforms the knowledge-base in XML format to the VHDL representation of the corresponding hardware acceleration device. The resulting device description corresponds with the input rule-base and is fully accommodated to the system bus width and facts count, providing appropriate communication functionalities. The tool is described in more detail in one of our previous works [18].

Using the developed tool it is easy to generate a hardware architecture corresponding to any knowledge base fulfilling the stated requirements. The provided architecture is then compiled using standard software tool for the desired programmable hardware and is ready to be programmed into the programmable device. The *knowledge base parser* tool thus contributes to the embedded expert systems development and maintenance process.

7. Evaluation

To evaluate the results of our work, we have performed several experiments with one of the developed architectures for hardware acceleration. We have decided to focus on the hardware implementation of inference approach, as this appeared more promising in the means of acceleration capabilities and versatility. We focused on the attributes of the architecture's resource consumption and acceleration capabilities.

7.1 Experimental setting

For experiments, we have used an embedded development kit equipped with Intel PXA255 XScale processor, providing hardware interface for connecting additional module as asynchronous I/O device utilizing the system's parallel communication bus. For implementation of the acceleration hardware device, we have used a module equipped with the Altera Cyclone EP1C6 - an FPGA programmable device containing 5,980 logic elements and 92,160 RAM

bits. In our design, only the logic elements are utilized. The output *Done* has been connected to the general input/output pin of the processor configured to serve as an external interrupt input. We have used Linux 2.4.19 operating system and own drivers for communication with the acceleration device.

For the experiments we have implemented a knowledge-based system that operates in two modes. In mode one, assistance of the acceleration hardware is used for the computation. In mode two, inference is computed in software, using the naïve reasoning method. In the naïve reasoning method, all rules have to be searched and evaluated for each iteration of inference, what counts for computational complexity of $O\left(F\left(RF\right)^{I}\right)$ where F states for number of facts in the system, R states for number of system's rules and I states for the number of iterations needed to finish the inference.

Several approaches to acceleration of the software computation exist, some of them claiming even linear computational complexity. However, in these approaches the computational complexity is decreased at the expense of memory complexity. As the memory consumption is critical in embedded applications, these approaches are not suitable for our consideration. Furthermore, precise algorithms used in these approaches are very difficult to obtain, some of them being subject of copyright.

For the experiments, we have used three various sources of input data. Firstly, we have manually created several knowledge bases representing real problems of common knowledge-based systems domains. As manual creation of reasonable knowledge bases is time consuming, we have created a random generator of knowledge bases that have logical structure similar to real ones and are suitable for our experiments. These knowledge bases were particularly useful in experiments with architecture resource consumption. For experiments with the computation time, we have created knowledge bases of a structure that maximizes iteration count.

7.2 Architecture resource consumption

Resource consumption of the developed architecture is one of its key properties as it states the requirements for the needed programmable hardware and influences the added cost of the solution. By experiments with the hardware resource consumption we intended to verify feasibility of the proposed solution. Other goals of the experiments were to:

- Determine the hardware resource consumption of an architecture representing a small expert system.
- Determine the limitations for the knowledge-base size and complexity, when assuming utilization of standard programmable hardware.
- Determine the resource consumption dependency on the knowledge-base size.

Each provided knowledge base has been translated into its VHDL hardware representation using the developed translation tools and placed & routed to the Altera Cyclone EP1C6 FPGA. The results are summarized in Table 1.

Table 1: Hardware resource consumption of the inference architecture.

Knowledge base	Fact count	Rule count	LEs	FPGA resources (%)
Car Simple	29	7	119	1.99
Secure System	40	17	16	2.66
Car Complex	105	52	333	5.57
100	99	100	338	5.65
Random 1	462	184	1755	29.35
1000	999	1000	3379	56.51
1500	1499	1500	5106	85.38
Random 2	1514	626	5812	97.19
1600	1599	1600	-	>100

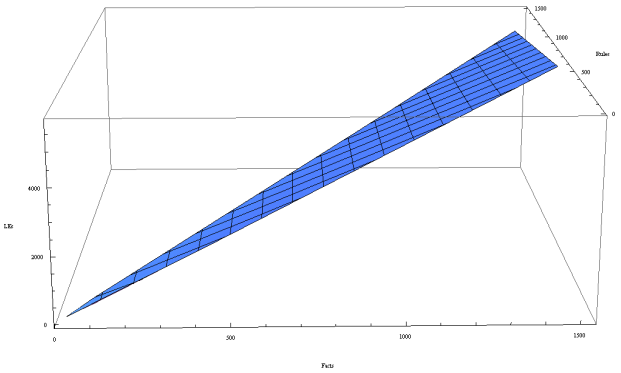

Figure 5: Linear interpolation of the resource consumption values.

We can see that a simple FPGA is suitable for implementation of a relatively extensive knowledge-based system. Furthermore, the obtained results show that dependency of resource demands from system's fact and rule count can be considered linear, as can be seen in Figure 5. Thus it is theoretically possible to implement acceleration of even more complex knowledge-based systems into larger programmable devices.

7.3 Inference acceleration

The acceleration capabilities of the architecture were measured by comparing computation time of software implemented and hardware accelerated rule-based reasoning. For measuring computation duration we have used the tool *time* provided by the operating system. The obtained results correspond with theoretically stated computation complexity values. The software computation time rapidly rises with number of iterations in the inference process, easily reaching values unsuitable for implementation in real-time system. On the other hand, the hardware computation time rises linearly with the inference count, what gives it predispositions for real-time implementation. The count of iterations needed for the inference process is not directly dependent form the number of facts and rules in the knowledge-base, so even large knowledge-bases can lead to short computation times. However, it is hard to predict the actual iteration count, so the embedded system designer always needs to consider the maximum possible number.

Figure 6 shows dependency of computation time from iteration count up to 1599 iterations, for both software and

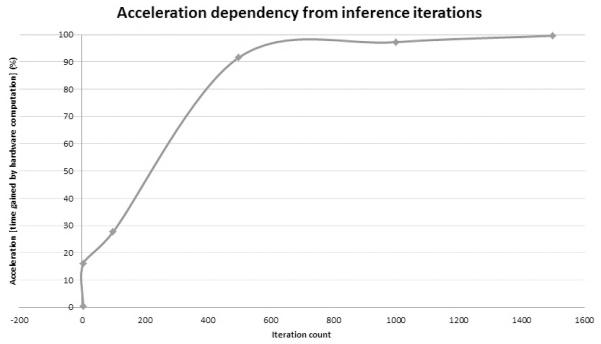

Figure 7: Dependency of inference acceleration (in %) from iteration count.

hardware implemented reasoning. The results are shown in logarithmic scale to make data readable. Figure 7 shows dependency of acceleration (hardware vs. software inference in %) from inference iterations.

8. From knowledge to hardware accelerated embedded expert system

An embed expert systems is a specific instance of embedded system that is specialized for a particular task and environment. Its development thus includes standard methods applied in development of embedded systems, e.g. hardware-software co-design. The development of an embedded expert system also requires extraction of knowledge and creation of knowledge-base. In our work we have described a process of optimizing standard knowledge-bases for implementation in embedded expert systems. In the work, we consider four suitable options of implementing embedded expert systems:

- Software expert system implemented on a universal, powerful processor.
- A simple expert system executed directly in program - the rules are embedded in program code.
- Expert system with hardware inference computation.
- Expert system with hardware mechanism for pattern matching.

Based on performed experiments and experience gained through the work we have proposed a set of heuristic rules on implementing embedded expert system, focused on choosing the implementation approach from the mentioned available options. The purpose of these rules is to provide support in the decision process and indirectly inform about the available options. Following inputs figure in the decision process:

- Average number of inference iterations.
- Size of system's operating memory and availability of memory management mechanisms.
- Processor computing power - relative to the needs of the particular method.
- Size of the system's program memory.
- Designation of the embedded system.
- Availability of programmable hardware.

9. Conclusions and future work

Expert systems are one of the approaches to implementation of intelligent systems; in embedded applications they are particularly suitable for control and monitoring systems and decision support systems. They are currently not used widely, because of limiting factors in embedded systems, such as their computational power, available memories or timing requirements. Development of embedded expert systems stays an open problem area with many challenges as development of appropriate hardware architectures, mechanisms of response time guarantee or methods of input/output processing. Another serious issue is the absence of common approach to developing embedded expert systems.

The main goal of our work was to facilitate implementation of embedded expert systems even in applications where it hasn't been possible before. We dealt with methods of inference acceleration and also touched the problem of common approach to developing these systems. The particular goals of our work were:

- development of method for expert systems acceleration applicable in embedded architectures,
- contribution to the problem area of response time estimation,
- contribution in the problem area of embedded expert systems development.

We have addressed the issues by developing two hardware acceleration schemes for embedded expert systems, by specifying the form of knowledge-base suitable for embedded expert systems, by developing a tool for automated creation of hardware architecture from the knowledge-base and by stating heuristic rules supporting the decision on the appropriate embedded expert systems implementation method selection.

We see the possible extension of our work in supplementing the designed embedded expert systems development toolchain by adding new tools for knowledge-base creation and optimization, as this process stays a significant bottleneck in expert systems' development process. The results of our work can be further extended to provide foundation for implementation of highly reliable embedded expert systems, reconfigurable embedded expert systems or hardware accelerated desktop systems.

Acknowledgements. This work has been partially supported by the Grants No. 1/1105/11 and 1/1008/12 of the Slovak VEGA Grant Agency. This work has been materially supported by the project ITMS 26240120005 supported by the Research 7 Development Operational Programme founded by the ERDF.

References

[1] C. Angeli. Online expert systems for fault diagnosis in technical processes. *Expert Systems*, 25(2):115–132, 2008.

[2] N. Bandi, S. Schneider, D. Agrawal, and A. El. Hardware Acceleration of Database Operations Using Content-Addressable Memories. In *Proceedings of the First International Workshop on Data Management on New Hardware*, 2005.

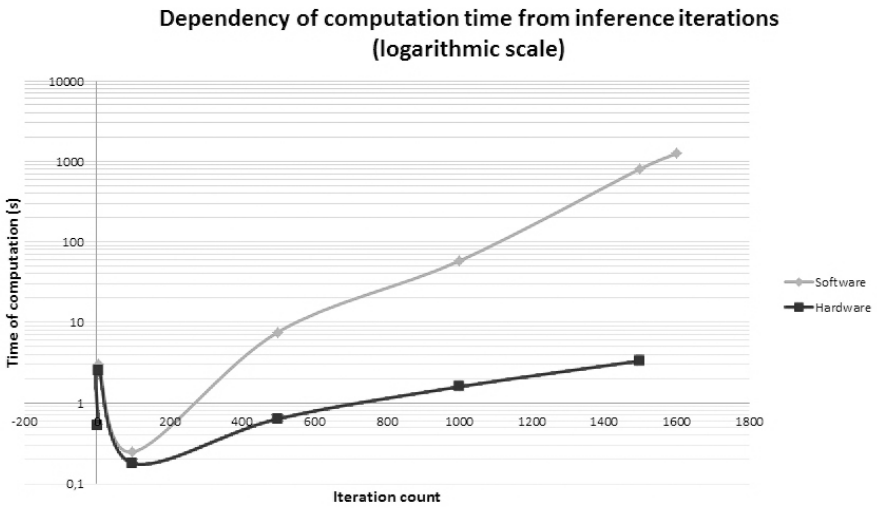

Figure 6: Software vs. hardware accelerated inference computation time (log. scale).

[3] J.-J. Chen and C.-F. Kuo. Energy-Efficient Scheduling for Real-Time Systems on Dynamic Voltage Scaling (DVS) Platforms. *Real-Time Computing Systems and Applications, International Workshop on*, 0:28–38, 2007.

[4] R. K. Chun. Software integration of real-time expert systems. *Control Engineering Practice*, 4(1):83–88, 1996.

[5] E. A. Coyle, L. P. Maguire, and T. M. McGinnity. Self-repair of embedded systems. *Engineering Applications of Artificial Intelligence*, 17(1):1–9, 2004.

[6] Devraj and R. Jain. PulsExpert: An expert system for the diagnosis and control of diseases in pulse crops. *Expert Systems with Applications*, 38(9):11463–11471, 2011.

[7] N. Dlodlo, L. Hunter, C. Cele, A. F. Botha, and R. Metelerkamp. A decision support system for wool classification. *Autex Research Journal*, 9(2):42–46, 2009.

[8] C. Ebert and C. Jones. Embedded Software: Facts, Figures, and Future. *Computer*, 42(4):42–52, 2009.

[9] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[10] R. Garcia and J. L. Calvo Rolle. Supervised rule based thermodynamic cycles design technique. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6679 LNAI(PART 2):327–335, 2011.

[11] M. R. Gómez, J. E. Ventosa, and G. A. Aramendía. Expert System Hardware for Fault Detection. *Applied Intelligence*, 9(3):245–262, 1998.

[12] P. Jackson. *Introduction to Expert Systems*. Addison-Wesley, 1999.

[13] T. J. Laffey, P. A. Cox, J. L. Schmidt, S. M. Kao, and J. Y. Read. Real-time knowledge-based systems. *AI Mag.*, 9(1):27–45, 1988.

[14] I. Lee, J. Y.-T. Leung, and S. H. Son. *Handbook of Real-Time and Embedded Systems*. Chapman & Hall/CRC, 2007.

[15] P. Morizet-Mahoudeaux. On-Board and Real-Time Expert Control. *IEEE Expert: Intelligent Systems and Their Applications*, 11(4):71–81, 1996.

[16] M. Pohronská. *Využitie programovateľného hardvéru na realizáciu expertných vnorených systémov*. Disseration thesis, Slovenská technická univerzita, 2012.

[17] M. Pohronská and T. Krajčovič. Implementation of Expert Systems in Embedded and Real-Time Systems. In V. N. Ján Kollár, editor, *Proceedings of the Tenth International Conference Informatics 2009*, volume 10, pages 259–264. Department of Computers and Informatics FEEI TU of Košice, 2009.

[18] M. Pohronská and T. Krajčovič. Hardware-accelerated Rule-based Systems for Embedded Platforms. In *Proceedings of the International Conference CYBERNETICS AND INFORMATICS 201*, pages 69–70. Vydavateľstvo STU, 2012.

[19] X. L. L. X. L. L. K. A. Qian Y. LUBRES: An expert system development and implementation for real-time fault diagnosis of a lubricating oil refining process. *Expert Systems with Applications*, 35(3):1252–1266, 2008.

[20] V. Stanchev. Consulting expert system for coreless induction furnaces control. In *Proceedings of 2006 IFAC Workshop on Energy Saving Control in Plants and Buildings*, 2006.

[21] A. Wichert. Associative diagnosis. *Expert Systems*, 22(1):26–39, 2005.

[22] W. Zhao. A fault diagnosis and operation advising cooperative expert system based on multi-agent technology. In *Power Plants and Power Systems Control 2006*, pages 195–200, 2007.

Selected Papers by the Author

M. Pohronská, T. Krajčovič. Implementation of the Handheld Decision Support System for Agriculture and Home Gardening. In *Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering*, in press, 2012. Springer.

M. Pohronská, T. Krajčovič. Embedded System Architecture for Real-time Rule-based Reasoning. *2nd Eastern European Regional Conference on the Engineering of Computer Based Systems Proceedings*, pages 85–91, Bratislava, Slovakia, 2011. CS IEEE Press.

M. Pohronská, T. Krajčovič. Implmentation of Multiple Hardware Watchdog Timers for Enhancing Real-Time Systems Security. In *Eurocon 2011 Proceedings*, Lisboa, Portugal, 2011. IEEE.

M. Pohronská, T. Krajčovič. Using Multiple FPGA Implemented Watchdogs for Improving of Embedded Systems Reliability. *Journal of Cybernetics and Informatics*, Vol. 11, pages 41–48, 2010.

M. Pohronská, T. Krajčovič. Embedded Systems with Increased Reliability Using the Multiple Watchdog Timers Approach. In Jiří Pinker, ed. *2010 International Conference of Applied Electronics Proceedings*, pages 273–276, Pilsen, Czech Republic, 2010. IEEE.

M. Pohronská, T. Krajčovič. Embedded Systems with Increased Reliability Using the Multiple Watchdog Timers Approach. In Jiří Pinker, ed. *2009 International Conference of Applied Electronics Proceedings*, pages 207–210, Pilsen, Czech Republic, 2009. IEEE.

M. Pohronská, P. Malík, M. Baláž. FPGA implementation of fully parallel fast MDCT algorithm. *Eurocon 2009 Proceedings*, pages 161–166, Saint Peterburg, Russia, 2009. CS IEEE Press.