AN EMBEDDED SYSTEM FOR THE ELECTROCARDIOGRAM RECOGNITION

Christos Pavlatos*, Alexandros Dimopoulos* and G. Papakonstantinou*

* National Technical University of Athens
Dept. of Electrical and Computer Engineering
Zografou 15773, Athens
Greece

{pavlatos, alexdem, papakon}@cslab.ntua.gr

Abstract: In this paper we present an embedded system for the efficient recognition of the Electrocardiogram (ECG). The aforementioned system is generated by a proposed platform that automatically maps applications on embedded systems using reconfigurable hardware FPGA (Field Programmable Gate Arrays). The ECG recognition system attains a speed up factor of approximately x20 compared to a fully software-based approach. The platform exploits both the procedural and declarative formalism increasing by this way its expressive power. The underlying model used for the implementation is that of Attribute Grammars (AGs).

Introduction

The Electrocardiogram (ECG) is routinely used in clinical practice. Due to the large number of ECGs analyzed in daily basis, it is worthwhile to automate and accelerate the process to the maximum extent possible. Previous attempts have been made using decision-theoretic methods, syntactic methods and hybrid methods.

The great evolution in technology has significantly modified the methods of designing computing systems, leading gradually in more high level formalisms. The combinational use of procedural and declarative models would increase the expressive power of these formalisms for a considerable number of applications.

This formalism can be used for pattern recognition applications and more specifically for the ECG recognition. The automatic mapping of applications expressed in the above formalism on special purpose embedded systems may substantially improve the execution time. Attribute Grammars (AGs) have extensively been utilized for the hybrid approach to pattern recognition. The task of recognition is essentially reduced to that of parsing a linguistic representation of the patterns to be recognized. Due to their descriptive power AGs may be selected [1], [2] and applied as the model for the formulation of a pattern grammar for ECGs.

Attribute grammars (AGs) were introduced by Knuth [3] in 1968. The addition of attributes and semantic rules to Context free grammars (CFG) augmented their expressional capabilities, making them by this way a really useful tool for a considerable number of applications. AGs have been extensively utilized in Artificial Intelligence applications [4], [5], [6], structural pattern recognition [7], [8], compiler construction [9], and even text editing [10]. However, the additional complexity imposed by the added characteristics, along with the need for fast CF parsing by special applications, dictates the parallization of the whole procedure (parsing and attribute evaluation) as an attractive alternative to classical solutions.

Conventional approaches [11], [12] in the evaluation of AGs are based on the construction of the parse tree that is after traversed one or more times, depending on the form of the grammar, in order to evaluate the attributes. The division of the whole procedure into two separate tasks (parsing and attribute evaluation) and the execution of these, one after the other make these approaches inefficient, although the second task (attribute evaluation) may be executed in parallel. However in [13], [14] where the parsing is executed in parallel with the attribute evaluation, the parsing is sequential while these evaluators may only be applied to a restricted form (S-Attributed) of AGs [12], [13].

In [15] we proposed a parallel algorithm that evaluates attributes simultaneously with the parallel parsing. The architecture consists of $n+2$ processing elements, where n is the length of the input string. One element is used for the control of the whole process, one element is used for the attribute evaluation, and n elements are used for the parallel parsing. The algorithm we proposed is based on the parallel version of Earley's [20] algorithm that was introduced by Y. Chiang & K. Fu [16]. The parallel algorithm has time complexity $O(n)$ and it approximately needs the same time to both parse the input string and evaluate the attributes compared to time needed to solely parse the input string. This fact makes our approach essentially more efficient compared to the conventional approaches where the attribute evaluation process follows that of the parsing phase or the approaches where although the semantics evaluation phase and the parsing are executed in concurrently, the parser is a sequential one. In addition, since parsing and attribute evaluation are executed in parallel, we have the ability, by checking some rules

ISSN: 1727-1983

that we can pose, to early detect a syntax error, avoiding the continuation of the parsing or pruning the parse tree, succeeding by this way to semantically drive the parsing process [17].

In [8] a hardware implementation of a shape grammar was proposed using [16] as well. Our architecture [15] overtakes [8] due to aforementioned improvements on [15]. Additionally, in [8] there were two main limitations. The first is that the attributes are standard and only for the proposed attribute grammar and the second is that the scale of the hardware is problem and input string length depended.

In this paper *i)* a platform is proposed for the automatic mapping of applications on embedded systems using reconfigurable hardware FPGA (Field Programmable Gate Arrays) , *ii)* a parallel algorithm that evaluates the semantics of a grammar simultaneously with the procedure of syntactic analysis is presented that is an improved version of the one we presented in [15] due to the fact that the system also accept input strings. Our approach is based on the fastest parsing algorithm (Earley's parallel parsing algorithm [16]) extended to handle attribute evaluation computations for L-Attributed grammars. The parallel algorithm has time complexity *O(n)* using *n+2* processing elements where n is the length of the input string (representing the ECG), and *iii)* we have used the platform for the implementation of an embedded system for the efficient normal ECG recognition.

The rest of the paper is organized as follows. Second section analyses the necessary theoretical background; third section presents the relation of ECG and AGs; next materials and methods are stated. In the last two sections we discuss the performance of the proposed system and outline directions for future work.

Introduction to Attribute Grammars

An AG [3] is based upon a Context Free Grammar (CFG). A CFG is a quadruple G = (N, T, R, S), where N is the set of non-terminal symbols, T is the set of terminal symbols, R is the set of grammar rules (a subset of N x(N $\cup$ T)* written in the form A $\longrightarrow\alpha$, where A $\in$ N and $\alpha \in$ (N $\cup$ T)*) and S (S $\in$ N) is the start symbol (the root of the grammar). We use capital letters A, B, C… to denote non terminal symbols, lowercases a, b, c… to denote terminal symbols and Greek lowercases α, β, γ... for (N $\cup$ T)* strings, λ is the null string and V = N $\cup$ T is called vocabulary.A $\overset{*}{\longrightarrow} a$ means that *a* can derive from A after the application of one or more rules.

Let S $\overset{*}{\longrightarrow}$ α, ($\alpha \in T^*$) be a derivation in G. The corresponding *derivation (parsing) tree* is an ordered tree with root S, leaves the terminal symbols in α, and nodes the rules that are used for the derivation process. The process of analyzing a string for syntactic correctness is known as *parsing*. A *parser* is an algorithm that decides whether or not a string $a_1a_2a_3\ldots a_n$ can be generated from a grammar G and simultaneously constructs the derivation (or parse) tree.

An AG is a quadruple AG = {G, A, SR, d} where G is a CFG, A = $\cup$ A(X) where A(X) is a finite set of attributes associated with each symbol X $\in$ V. Each attribute represents a specific context-sensitive property of the corresponding symbol. The notation X.a is used to indicate that attribute *a* is an element of A(X). A(X) is partitioned into two disjoint sets; the set of synthesized attributes $A_S(X)$ and the set of inherited attributes $A_I(X)$. Synthesized attributes X.s are those whose values are defined in terms of attributes at descendant nodes of node *X* of the corresponding semantic tree. Inherited attributes X.i are those whose values are defined in terms of attributes at the parent and (possibly) the sibling nodes of node *X* of the corresponding semantic tree. The start symbol does not have inherited attributes. Each of the productions $p \in P$ ($p: X_0 \rightarrow X_1 \ldots X_n$) of the CFG is augmented by a set of semantic rules SR(*p*) that define attributes in terms of other attributes of terminals and on terminals appearing in the same production. The way attributes will be evaluated depends both on their dependencies to other attributes in the tree and also on the way the tree is traversed. Finally d is a function that gives for each attribute *a* its domain d (*a*).

An *l-attributed grammar* is an attribute grammar where any semantic rule of any production $p: X_0 \rightarrow X_1 \ldots X_n$ satisfies the following conditions:

Inherited attribute values of X_i can be determined using inherited attributes of X_0 and synthesized attribute values of $X_1 \ldots X_{i-1}$

Synthesized attribute values of X_0 can be determined using inherited attribute values of X_0 and synthesized values of $X_1 \ldots X_n$

Attribute Grammars and ECG

In Syntactic Pattern Recognition, the task of recognition is essentially reduced to that of parsing a linguistic representation of the patterns to be recognized with a parser that utilizes a certain grammar, called "pattern grammar" [7]. The pattern grammar describes the patterns to be recognized in a formal way, and the formulation and parsing of the grammar are always the crucial subproblems in a pattern recognition application that is to be tackled by the syntactic approach. In the case of ECGs, where we have a large number of different morphologies of the patterns, where added morphologies can be found due to the noise, and where measurements of the various parameters have to be performed, powerful grammars capable of describing syntax as well as semantics are needed as a model for the formulation of a pattern grammar. Due to their descriptive power attributes grammars are usually [1], [2] selected and used as the model for the formulation of a pattern grammar for ECGs. We used the automated synthesis tool to generate a hardware parser for the syntactic part of the attribute grammar presented in [1].

In [1] the alphabet of symbols T ={K^+, K^-, E, Π} has been adopted for encoding the ECG waveforms, where K^+ denotes positives peaks, K^- negative peaks, E straight line segments and Π parabolic segment. Thus an

ISSN: 1727-1983

ECG waveform is linguistically represented as a sting of symbols from the alphabet T. Each symbol is associated with the values of the corresponding attributes. Therefore a string of the following form may be a linguistic representation of an ECG waveform.

$\Pi\ K^-\ K^+\ \Pi\ K^-\ E\ K^-\ K^+\ \Pi\ K^-\ K^+\ E\ K^-\ E\ K^-\ K^+\ E$

Using the grammar [1], consisting of 34 syntactic rules with maximum length 7 symbols (terminals or non-terminals), strings of the abovementioned form can be parsed. By the simultaneous evaluation of the attributes the ECG may be recognized.

Materials and Methods

Our main intension is to accelerate the performance of applications which are based on syntactic pattern recognition. Since ECG waveform recognition may easily be transformed into an equivalent AG evaluation problem, the underlying model of implementing an embedded system for the aforementioned applications is that of an AG evaluator.

The method used is the separation of the parsing from the semantics evaluation procedure. The parsing is carried out by hardware (FPGA) and the semantic evaluation is carried out by a conventional RISC microprocessor. The cooperation of the FPGA with the RISC microprocessor forms an embedded system for the recognition of the normal ECG.

An additional goal is to automate the procedure of designing so as to present a platform that, for a given AG, automatically generates the proper code to build the embedded system. As far as the parsing is concerned, we embedded the reconfigurable hardware parsers presented in [21] in the parallel architecture presented in [15]. As far as the attribute evaluation is concerned, we used an improved version of the parallel algorithm we presented in [15]. All the above components were merged in a platform, for the automatic mapping of applications on embedded systems, which reconfigures the components according to the input AG. The platform outputs the proper Verilog HDL [18] source code to be downloaded to the FPGA as well as the proper assembly code to be downloaded to the RISC microprocessor.

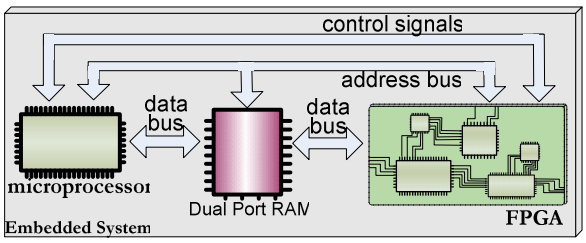

Figure 1: Overview of our approach

The Risc microprocessor handles the semantic evaluation while the parallel algorithm that implements the parsing is handled by a Xilinx FPGA board [19]. The board interfaces with the microprocessor using hardware/software co-design methods (see Figure 1). All data are stored in a Dual Port RAM, shared by both components. Finally this platform has been used for the implementation of an embedded system dedicated to the ECG recognition.

The Proposed Architecture

The procedure of parsing may be reduced to the procedure of filling a two dimension table (parsing table). Chiang & Fu [5] proved that the construction of the parsing table can be parallelized with respect to the length of the input string *n* by computing at step k the cells pt(i, j) for which $j - i = k \geq 1$.Only the elements on or above the diagonal are used. In [15] a parallel architecture (see Figure 2) has been presented that uses *n+2* elements to compute the parse table in *O(n)* time where *n* is the input string length. Every processing element is computing one cell pt(i, j) in each execution time (t_{e1}, $t_{e2,}$ ... t_{en}) and the next execution time is used again to compute the cell that belongs to the same column and is one row higher pt(i-1, j). In addition one processing element is required to control the whole process and one more to handle the attribute evaluation process as shown in Figure 2. The *n* elements that are used for the parallel parsing are following the design presented in [21].

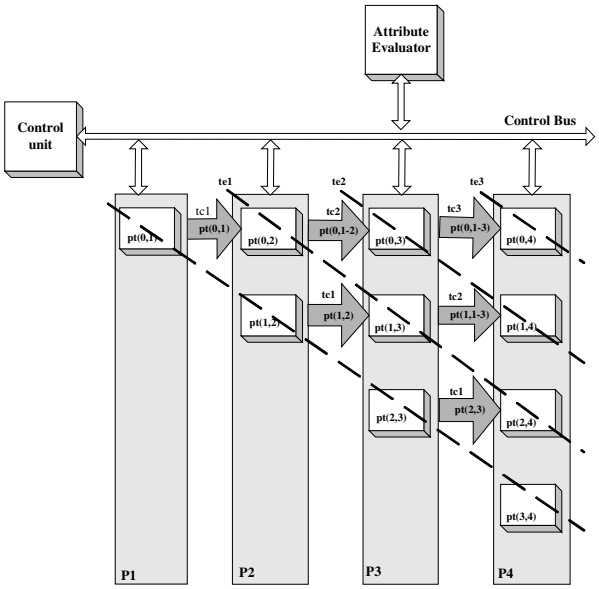

Figure 2: The parallel architecture for the attribute evaluation

After the end of each execution step *k,* the computation of one parsing processing element terminates. At the next execution step this processing element should transmit (communication step: t_{c1}, $t_{c2,}$ ... t_{cn}) the cells that it has computed, to the processing element that handles the attribute evaluation. The cells that are transferred contain the rules that can possibly generate the input string so far. The attributes of these rules are then evaluated, in the next execution step, by the attribute evaluator while the processing elements are calculating the cells of the current step. Each processing element repeatedly calculates a cell, checks if it should send some cells and then if it should receive any.

ISSN: 1727-1983

Finally the processing element sends its cells to the attribute evaluator. During the attribute evaluation process it is crucial to traverse the elements of the parse table in such a way that guaranties the correct computation of the attributes in one pass. An efficient way to achieve this goal that is presented in [15] is also adapted in the presented effort.

Performance Evaluation

In Figure 3 the efficiency (in clock cycles) of seven case studies are shown for various input string lengths. The proposed approach gives better results, attaining a speed up factor of approximately x20 compared to a fully software-based approach. The software implementation uses a conventional Risc microprocessor.

Provided that the technology used for the hardware implementation is the same with the one used for the microprocessor we can safely claim that the clock frequency for both implementations may be the same. Consequently the performance in all implementations is measured in clock cycles. Measurements have been taken for various implementations and number of input string length.

The performance in all implementations is measured in clock cycles. In Figure 3 seven measurements are presented, the clock cycles of the software parser, of the software parser and the attribute evaluator, of a sequential hardware parser, of a sequential hardware parser and the attribute, of the parallel hardware, of the parallel hardware parser and the attribute and finally of our approach that is parallel hardware parser and concurrently attribute evaluator.

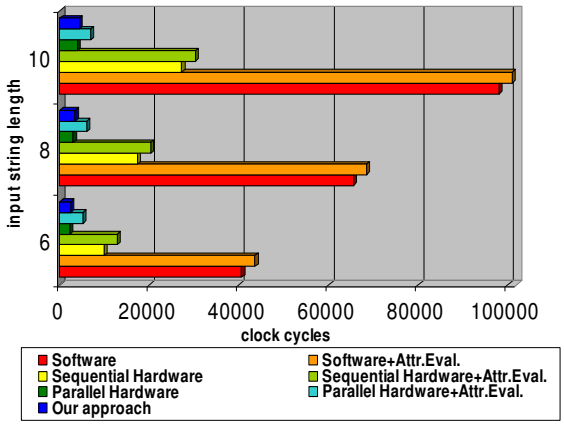

Figure 3: Performance evaluation

Apparently, the number of clock cycles required to firstly construct the parse tree and then by using it to evaluate the attributes –regardless the implementation, software or hardware, sequential or parallel- are greater than the clock cycles required by our approach. Additionally, the fact that our approach appears to be slower compared to time required just to parse the input string by a factor that is a small fraction of the whole parsing process is really an encouraging event.

Conclusion

This paper presents an embedded system for the efficient recognition of the ECG. The embedded system is based on a parallel parsing algorithm that simultaneously evaluate attributes, implemented using a proposed design platform.

This work is a part of a project[1] for developing a platform (based on AGs) in order to automatically generate special purpose embedded systems. The application area will be that of Artificial Intelligence (AI), of Syntactic Pattern Recognition for Electrocardiogram (ECG) analysis and of Signal Processing using software hardware co-design techniques. Alternatively, the whole implementation both microprocessor and parser may be mapped on a single FPGA board.

References

[1] TRAHANIAS P., SKORDALAKIS E. (1990): 'Syntactic Pattern Recognition of the ECG', *IEEE Transactions on PAMI*, **12**

[2] PAPAKONSTANTINOU G. (1986): 'An Attribute Grammar for QRS detection', *Pattern recognition*, **19**, pp.297-303

[3] KNUTH D. (1971): 'Semantics of context free languages', *Math. Syst.Theory*, **2**, pp.127-145

[4] PAPAKONSTANTINOU G., KONTOS J. (1986) 'Knowledge Representation with Attribute Grammars', *The Computer Journal*, **29**

[5] PAPAKONSTANTINOU G., MORAITIS C.and PANAYIOTOPOULOS T.(1986) :'An attribute grammar interpreter as a knowledge engineering tool', *Applied Informatics,* **9,** pp. 382-388

[6] PANAGOPOULOS I., PAVLATOS C. and PAPAKONSTANTINOU G. (2004) :'An Embedded System for Artificial Intelligence Applications', *International Journal of Computational Intelligence*

[7] FU K. (1982): 'Syntactic Pattern recognition and Applications', PRENTICE-HALL

[8] CHEN H. AND CHEN X. (1993):'Shape recognition using VLSI Architecture', *The International Journal of Pattern Recognition and Artificial Intelligence*

[9] AHO A., SETHI R. AND ULLMAN J. (1986): 'Compilers – Principles, Techniques and Tools.' Reading, MA:ADDISON-WESLEY,, pp. 293-296

[10] DEMERS A., REPS T., AND TEITELBAUM T. (1981):'Incremental evaluation for attribute grammars with application to syntax-directed editors', in Conf. Rec. 8th Annu. ACM symp. Principles Programming Languages, (1981), pp.415-418

[11] BOEHM H.J. AND ZWAENEPOEL W.: 'Parallel Attribute Grammar Evaluation', *In. R. Popescu-Zeletin, G. Le Lam, and K.H. Kim*, editors,

[1] This work is co - funded by the European Social Fund and particularly the Program "Pened 2003".

Proceedings of the 7th International Conference on Distributed Systems, Berlin,Germany,1987, pp.347-354

[12] KLAIBER A. and GOKHALE M (1992): 'Parallel Evaluation of Attribute Grammars' , IEEE Trans. on Par. and Distr. Sys., **3**, pp.206-220

[13] PAVLATOS C., KOULOURIS A. and PAPAKONSTANTINOU G.(2003):'Hardware Implementation of Syntactic Pattern Recognition Algorithms', IASTED International Conference on Signal Processing and Pattern Analysis (SPPRA), pp. 360-365

[14] TOKUDA T. and WATANABE Y. (1994) :'An attribute evaluation of context-free languages', *Information Processing Letters,* **57**, pp. 91-98

[15] PAVLATOS C., DIMOPOULOS A. and PAPAKONSTANTINOU G.: (2005) 'An Intelligent Embedded System for Control Applications', Workshop on Modeling and Control of Complex Systems, Cyprus ,2005

[16] CHIANG Y. and FU K. (1984) : 'Parallel parsing algorithms and VLSI implementation for syntactic pattern recognition', I*EEE Trans. Pattern Anal. and Mach. Intell. PAMI*,**7**

[17] SIDERI M., EFREMIDIS S. and PAPAKONSTANTINOU G. (1994): 'Semantically driven Parsing of CFG' T*he Computer Journal*, **32**, pp91-98

[18] PALNITKAR, S. : 'Verilog HDL, A guide to digital design and synthesis', PRENTICE HALL, Second Edition

[19] XILINX, Internet site address: http://www.xilinx.com

[20] EARLEY J. (1970):'An efficient context–free parsing algorithm', *Com. of ACM*, **13**, pp. 94-102

[21] PAVLATOS C., PANAGOPOULOS I. and PAPAKONSTANTINOU G: 'A programmable Pipelined Coprocessor for Parsing Applications' Workshop on Application Specific Processors (WASP) CODES, Stockholm, 2004

ISSN: 1727-1983