International Journal of Soft Computing 9 (6): 364-376, 2014
ISSN: 1816-9503

A Bayesian Belief Network Based Decision Support System for Embedded System Design

[1]V. Prasanna Srinivasan and [2]A.P. Shanthi
[1]Department of Information Technology, R.M.D. Engineering College, Chennai, India
[2]Department of Computer Science and Engineering, College of Engineering, Anna University, Chenni, India

Abstract: This study presents a decision support system that aids the embedded system designers during the synthesis phase to select the optimal system components such as processors, memories, communication interfaces, etc., from the available huge design alternatives. The selection process should consider the configuration options available both at the system level and the micro-architectural level, along with the knowledge about the system parameters that affect the overall objectives of the system in order to satisfy the applications requirements. The focus of the Electronic Design Automation (EDA) community is towards developing an efficient strategy for aiding the system designer during the synthesis to incorporate the domain knowledge of the target architecture and to take early design decisions. The Bayesian Belief Network (BBN) based modeling framework proposed in this study attempts to resolve the existing limitation in imparting domain knowledge and provides a pioneering effort to support the designer during the process of embedded system design. Sensitivity analysis is performed for identifying the most influential parameters for the decision making and to verify the robustness of the proposed model. Case studies in support of the proposed model are presented in order to understand how the BBN can be used in the embedded system design process by propagating the evidence and arriving at inferences in such a way to ease the decision making process.

Key words: Embedded system design, bayesian belief network, Decision Support System, processors, EDA

INTRODUCTION

A Decision Support System (DSS) is an interactive Computer-Based System or subsystem intended to support decision makers in diverse domains such as medical, transportation, risk management, safety control, product development, etc. The DSS helps the decision makers to use modern communications technologies, handling multiple datas, documents and inferring knowledge from existing models or data to identify and solve problems in order to take decisions early. The main focus of developing a DSS for a particular domain is to improve decision quality and productivity and to reduce cost and time. A systematic approach is essential during the development of any DSS to handle quantitative information at hand for building the model and to ensure easy integration with the existing system.

The importance of developing an intelligent DSS has grown to a large extent due to the technological advancements. Intelligent DSS System has the ability to learn or understand from experience, respond quickly and successfully to a new situation, use reasoning to solve problems and to apply knowledge and infer to manipulate the environment. Artificial Neural Networks (ANN), Inductive learning, Case-based reasoning, Genetic algorithms and fuzzy logic are some of the examples of Intelligent algorithms or techniques used during the development of any DSS.

A DSS for the modern Embedded Computer System design has become important due to the rapid growth with respect to technology scaling in terms of including more number of transistors within a single chip. The scope of adding more number of functionalities has been increased because of the further miniaturization possible at the Electronic System Level (ESL) design. The most important challenge for the embedded system designer is to adapt and utilize the underlying technology efficiently so as to incorporate all the functionalities and constraints required for the given application. The modern system design methodology has been shifted to be application specific rather than providing a generalized architectural design solution. This shift in the design paradigm has provided a mechanism to build and implement a complete system designed for a particular application within a single chip.

Corresponding Author: V. Prasanna Srinivasan, Department of Information Technology, R.M.D. Engineering College, Chennai, India

The objective that must be satisfied during the embedded system design depends primarily on the constraints imposed by the application. Generally, multiple objectives must be considered such as power consumption, performance, core size, memory bandwidth and energy. Apart from these fundamental constraints, there are other higher level factors like incorporating flexibility, providing reconfigurable logic, designing scalable or reusable architecture, Intellectual Property (IP) core integration, etc. which further increases the complexity of the design process (Chen and Kung, 2008).

The embedded system design process usually starts with the initial specification of the application and its requirements. The designer analyses the objectives and nature of the application and requires a high level test bench to aid the design process during the synthesis phase. The synthesis phase is the most critical aspect of the embedded system design process in which the designer has to perform the sub tasks such as representation of the application, modeling the target architecture, consideration of the different design alternatives available in terms of the system parameters, mapping of application onto the chosen design configuration and validating the performance. The critical aspects involved during the synthesis phase cannot be performed manually because of the huge design effort and time needed to complete the entire process. This is the exact stage at which an Electronic Design Automation (EDA) tool is highly essential for aiding the designer during the synthesis phase.

The extensive research effort in each of the sub tasks involved in the synthesis phase has been significant in recent years. Since, considerable amount of time is needed for the selection of optimal design, the most promising effort considered by the EDA community is to provide a mechanism to aid the designer for selecting the optimal design configuration from the available alternatives, considering multiple objectives and to take early design decisions. The challenge that arises with respect to the development of the tool is how much of the mapping of available design space onto the objective space could be automated. Further to this challenge, the objective space usually constitutes multiple criteria to be considered during the design thus transforming the tool design toward multi-criteria decision making problem.

A Bayesian belief network is a graphical model that determines the causal relationships among a set of variables through probabilistic reasoning (Heckerman, 1997). The Bayesian belief network has been utilized for decision making in most real-life problems such as forecasting, medical diagnosis (Vila-Frances *et al*., 2013; Yet *et al*., 2013), risk management (Lee *et al*., 2009; Fang and Marle, 2012), product development (Li and Wang, 2011), safety control (Zhang *et al*., 2013) and multi-criteria, multi-attribute decision making (Delcroix *et al*., 2013) and fault inference (Xu, 2012), etc. The structural and mathematical properties of the BBN easily allow the designer to represent the causal interdependencies that exist between the various system attributes and the specified constraints at the micro-architectural level. The ability of the BBN to propagate the evidence throughout the network provides an easier mechanism for understanding the cause and effect relation between the variables of the system being modeled. Since, the Bayesian belief network provides a convenient way for the construction and representation of the cause and effect relation among system attributes easily, it is considered to be a suitable methodology for decision making in the embedded system design process. This study proposes a BBN based DSS for embedded system design.

LITERATURE REVIEW

Embedded system computing has now become ubiquitous computing because of the advancements in the embedded system research and the various application domains in which it has been employed. The most important aspect in the embedded system design process is the consideration of the target architectural platform for the given application to be ported after final verification has been completed. Before the actual implementation has been done, the designer needs to evaluate and verify the correctness of the entire design using the test bench platform. Generally, two methods have been employed in describing the test bench platform, analytic modeling (Gries and Keutzer, 2005) of the target platform or using an Instruction Set Simulator (ISS) (Blythe and Walker, 2000; Burger and Austin, 1997). Analytic modeling of the target architecture has the ability to portray the exact behavior of the platform with high degree of precision. The difficulty in analytic modeling is the time required for building the model, considering all the aspects of the target architecture. ISS exactly resembles the target platform specification, providing user friendly environment for identifying the hot spots and facilities to improve the performance of the given application.

Various methodologies have been proposed for embedded system synthesis such as model-based design, Platform based Design (PbD) (Kempf *et al*., 2011) and abstract description of architecture using

programming constructs. The Ptolemy (Lee, 2003) framework enables the modelling, simulation and design of the concurrent, real-time embedded systems for analyzing the interaction between the system components, using heterogeneous mixture of Models of Computation (MoC). SPADE (Lieverse *et al*., 2001), a system level framework, presents an architectural exploration method based on a single MoC using Kahn Process Network (KPN) (Kahn, 1974). This framework follows the Y-chart principle for system level design based on various abstraction levels.

GRACE++ (Mueller *et al*., 2001) is a SystemC based simulation environment for Network on Chip (NoC) centric systems that targets abstraction levels higher than Register Transfer Logic (RTL) to achieve increased simulation speed. ARTEMIS (Pimentel *et al*., 2001) yet another system level framework has extended the research of SPADE by increasing the simulation speed and incorporating automatic heuristics for choosing design alternatives.

StepNP framework (Paulin *et al*., 2002) represents a development environment including the design aspects of architecture, application and tools. The main focus of this framework is to explore different network processor architectures. The fundamental differences that exist between these tools is the way the given application has been represented, the representation of architectural specifications, compiler optimizations possible, support for Field Programmable Gate Array (FPGA) emulation and how much the tool automates the design process.

System level synthesis using evolutionary algorithms has been implemented by Thiele *et al*. (2002) that target the applications and SoC architectural design in the domain of packet processing. Approximate methods (Eisenring *et al*., 2000; Fredriksson *et al*., 2005) have been implemented for optimization of the design through satisfying some predefined condition or specified constraints. Decision making in real-time embedded systems design has been formulated as a framework by Shankaran *et al*. (2006). Most of the tools or frameworks have not exploited the domain knowledge when modeling the architecture. The existing tools utilize heuristic based approaches for aiding the embedded system design process which result in sub optimal solutions for the decision making problem.

The idea proposed by Beltrame *et al*. (2010) moves the design complexity from simulation to probabilistic analysis of parameter transformations by employing the domain knowledge of the target architecture. Exploration process has been modelled as a Markov Decision Process (MDP) and the solution to such MDP relates with the sequence of parameter transformations to be applied in order to maximise or minimise the desired value function. The simulation has been performed only in particular cases of uncertainty thus massively reducing the simulation time needed to perform the exploration of a system while maintaining near optimality of the results.

BBN has been used by Olson *et al*. (2007) for performing the hardware/software partitioning during the embedded system design for categorizing the functional components into hardware and software. This has been done through incorporating a methodology for generating the qualitative structure of BBN and also the quantitative link matrices. A travelling problem scenario, Watthayu and Peng (2004) has been modeled using the BBN for multi criteria decision making. An efficient BBN based learning has been proposed for reliability engineering and optimization processes in industrial systems by Gruber and Ben-Gal (2012). A design guidance scheme using simulation based BBN has been proposed by Parakhine *et al*. (2008) aiding the designer with guidance by describing the causal relationships between various system characteristics and qualities. The BBN has been used to model the availability of different services provided by a company to the customers. Aleti and Meedeniya (2011) have implemented a heuristic based Bayesian learning algorithm for the component deployment optimization problem in the field of control system application.

Though several researchers have looked at different techniques, very little effort has gone into the incorporation of domain knowledge of the target architecture. The need to incorporate the domain knowledge in modeling the architectural parameters of the system is becoming increasingly important. The designer needs to understand the various architectural specifications in detail to gain knowledge about the target platform. The focus of acquiring the knowledge should be concentrated at the instruction set level of the target architecture. Thus the current demand in the EDA industry to automate the embedded system design process and to incorporate the domain knowledge of the system design has motivated to present the proposed framework, considering the architectural alternatives at the micro-architectural level and multiple objectives to be satisfied during the design for the given application. The Tensilica's Xtensa architecture ISS is considered as the target architectural platform for identifying the architectural attributes in this proposed technique. Tensilica's Xtensa ISS technology provides the configurable and extensible processor cores for the SoC designers with fully supported hardware and software generation. Once the knowledge about the behavior of the system parameters have been acquired then the causal

reasoning capability of the BBN can be fully exploited which has not yet been utilized in the embedded system synthesis process so far.

The fundamental advantage of using configurable and extensible processor is that the designer has the ability to include or exclude a functional unit and extend the basic instruction set that suits the application demand, during the design. Most techniques have not considered the modeling aspect of the target architecture at the functional instruction level. Since, the core instruction set of the Tensilica's Xtensa processor can be customized, the proposed framework utilizes the Xtensa Instruction Set Architectural (ISA) options and other general factors during the construction of the BBN for the design problem.

INTRODUCTION TO BBN

Bayesian Belief Networks (BBN's) have been widely used as a modeling technique for representing the domain knowledge and performing reasoning under uncertainty of a complex system. The construction of a BBN Model is divided into two stages, qualitative and quantitative (Nadkarni and Shenoy, 2004). During the first stage, the basic structure of the BBN is represented using DAG which exemplifies the probabilistic relationship among the variables of interest during the system modeling. The nodes in the DAG directly represent the key attributes of interest to be modeled. The directed edge between any two nodes describes the causal or influential relation among them. At the quantitative stage, each node in the BBN locally represents the uncertainty of the interdependence that exist among the attributes using the Conditional Probability Table (CPT). The important objective in constructing a BBN is to perform probabilistic reasoning of the system attributes that has been modeled (Neapolitan, 2003).

The core concept of BBN is based on the probability theory. Conditional probability, joint probability, Bayesian theorem, the chain rule and the product rule play a vital role in defining the mathematical aspects of the BBN (Daly *et al*., 2011). The significance of the BBN has been its capability to infer new knowledge from the existing knowledge. Another aspect of the BBN is the ability to represent the domain knowledge as causal relations among the variables. Figure 1 shows the BBN for a simple land purchase problem. The BBN consists of five discrete variables described as follows, Distance to city (D), basic Facilities (F), approved Land (L), Cost (C) and Purchase the land (P). The conditional independence relation between each of these variables is represented by DAG at

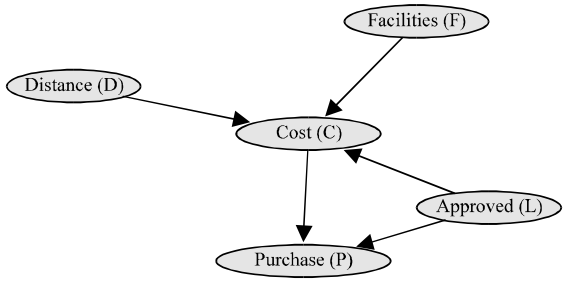

Fig. 1: Example of BBN for land purchase problem

the qualitative level. The variables D, F and L have a direct dependency relation with the variable C that is distance, facilities and whether the land is approved or not will have a direct effect on the cost of the land. Similarly, the variables C and L have a direct dependency relation with the variable P. The dependency relation between the variables has been normally achieved through the domain knowledge of the system being modeled or from the experts. A node X is said to be a parent of a node Y, if and only if there is an arc pointing from the node X to the node Y.

The dependency relations have been expressed in terms of CPT for each variable in the BBN, at the quantitative level. The variables have a set of possible states that represents the intended sample points from the available sample space which must be mutually exclusive and collectively exhaustive. The conditional distributions of each of the variables is denoted as follows: P (D), P (F), P (L), P (C|D, F, L) and P (P|C, L). Using these conditional probability distributions, the joint probability distribution can be computed easily. The fact that there is no arc from D to P and F to P signifies that P is conditionally independent of both D and F, given C and L. In this way, the conditional independence and dependence relations can be easily modeled and analyzed. The first step in the process of inferring from the BBN that has been constructed involves setting of evidence values for some of the variables. After specifying the evidence, the BBN has to be simulated for propagating the evidence throughout the network and to observe the influence of that evidence across all other variables. Thus, the causal relationship among the variables and the probabilities of outcomes has been captured within the network during the process of inference.

The construction of the BBN has been performed using two approaches namely, data based approach and knowledge based approach (Darwiche, 2010). Conditional independence semantics has been used during the construction of the BBN to induce models from the given data in the data based approach. This approach is very

much useful when the BBN has to learn from the existing data that is available as a training set. In the knowledge based approach, the causal knowledge of the domain experts has been used for constructing the BBN. This approach is useful in the modeling situation where the representation of the domain knowledge is of at most importance and the availability of initial set of data is limited. The most effective way of constructing the BBN has been possible by combining the two approaches using the expert's domain knowledge and the initial quantitative data available.

The first step involved in the BBN based modeling is to identify the attributes, influential factors (both external and internal), goals and uncertain criteria associated with the system being modeled. The second step involves building the network from the expert knowledge or available empirical data to specify the qualitative structure of the BBN based on the objectives to be achieved, considering certain factors or constraints. CPTs for each node have to be specified in the next step to reflect the causal interdependencies that exist among the variables quantitatively. Finally, the network has to be compiled to propagate the evidence that have been imposed on some of the variables to observe the behavior of other variables.

The BBN provides a systematic, comprehensive method of representing the causal interdependencies among the variables with the probabilistic information that is available and a convenient way for inferring the network with several inference algorithms. Exact algorithms (Korb and Nicholson, 2004) and approximate algorithms (Lauritzen and Spiegelhalter, 1988) have been used for inferring the BBN. The BBN has been employed in several application areas such as medical, forecasting and control as a solution model for taking decisions with multiple objectives. This has been the key motivation for the implementation of the proposed framework as a DSS for aiding the designer during the embedded system design process.

PROPOSED FRAMEWORK

The main objective of the proposed technique is to develop a modeling framework that has the ability to describe the inter-relations and uncertainties that exist between the key architectural attributes during the embedded system synthesis process. The proposed framework also considers the importance of assisting the system designers to take early design decisions. The structural properties and several advantages of the BBN have greatly influenced its utilization during the system design process.

Identification of architectural variables, factors and criteria: The most important step before building the BBN Model is to classify the identified architectural attributes in terms of their functional and behavioral aspects which influence the overall performance of the system. Instruction option, interface option and memory option are the three fundamental options that have been identified as constituting the core of the architectural specification. Instruction options provide necessary instructions to implement several operations. Interface options provide the facility to enable the communication between processor to processor and processor to memory. The memory options provide various combinations of Instruction Cache (IC), Data Cache (DC), cache associativity and cache line size. Imparting the knowledge gained about the performance of the system in the presence/absence and various combinations of these options is the key issue in the construction of the BBN.

The performance of the system depends not only on the functional units configured but also on some of the factors like total cycles taken for executing the application, cache misses, instruction counts of the application being executed, number of cycles taken for executing an instruction that is Cycles Per Instruction (CPI), etc. The interdependencies and importance of these highly influential factors is considered during the construction of the BBN. After having considered the architectural options and factors, the overall objectives like power consumption, core size, execution time, energy and cost have to be considered which further increases the complexity towards achieving a common goal pertaining to the design of any embedded systems. The nodes in the proposed BBN construction are classified into objectives, factors, constraints and set of actions. The entities associated with the problem under consideration are described in Table 1.

Table 1: Entities considered during the embedded system design problem

Objective	To design a modeling framework that aids the embedded system designer during the HLS phase
Decision problem	To decide the combinations of the functional instructions and other micro-architectural features to be included for the given application, with some constraints to be satisfied during the design
Set of alternatives (design space)	Instructions options, interface options, memory options
Criteria (objective space)	Execution time, cost, energy, area, power
Set of possible actions	Set of functional components from the design space to be mapped on to certain constraints in the objective space
Factors	Instruction count, cycles per instruction, total cycles, cache-misses, instruction fetch width, component density, technology

CONSTRUCTION OF BBN

Once the variables, factors and objectives have been identified then the system attributes can be represented qualitatively by constructing the BBN. Each variable associated with the system design is represented as a node in the BBN. The dependency relation between the variables is represented by an edge from one node to the other. The identification of the causal relationship between the variables is usually achieved through domain experts, fundamental knowledge about the system or by performing simulation studies. The proposed methodology for the construction of the BBN utilizes both the domain knowledge and the results obtained from a simulation study that is performed to analyze the behavior of the basic architectural parameters.

The simulation study is performed considering computationally intensive, memory intensive and communication intensive application workloads. The ALPbench (Li *et al*., 2005) and Mediabench (Lee *et al*., 1997) benchmark suite provides various application workloads for analyzing the performance of the target architecture. Three different applications such as Joint Photographic Experts Group (JPEG) encoder, Motion JPEG encoder, Advanced Encryption Standard (AES) and Secure Hash Algorithm (SHA) have been considered for the performance analysis. Each application has been simulated in thirty different configurations involving various combinations of instruction units in the target Tensilica Xtensa processor architecture. The simulated results exhibit the performance of the system in the presence or absence of the functional instruction units across various configurations. An edge between any pair of nodes in the BBN represents the direct dependency relation that exists between those two variables. Figure 2 shows the BBN that is constructed for the proposed problem with the predefined probabilities of each node. The BBN is constructed using GeNIe 2.0 which is an open source development environment for building graphical decision-theoretic models.

The target architecture considered has a certain basic instruction set for performing several operations. Other than these basic instructions there are some additional instructions that can be configured to suit a particular application's requirement. For example, a computationally intensive application may require a 32-bit multiplication instruction unit or a floating point instruction unit to improve the performance. The proposed model constitutes most of the functional instructions that can be customized which accounts for the combination of available ISA options. The functional instruction units such as multiply and accumulate, multiplication, integer division, floating point, density instructions, synchronize instructions, normalized shift, clamps and zero overhead loop instructions have been identified as the potential instruction units that directly affect the overall performance. The behavior of the system is modeled with respect to the effect of presence or absence of these functional instruction units.

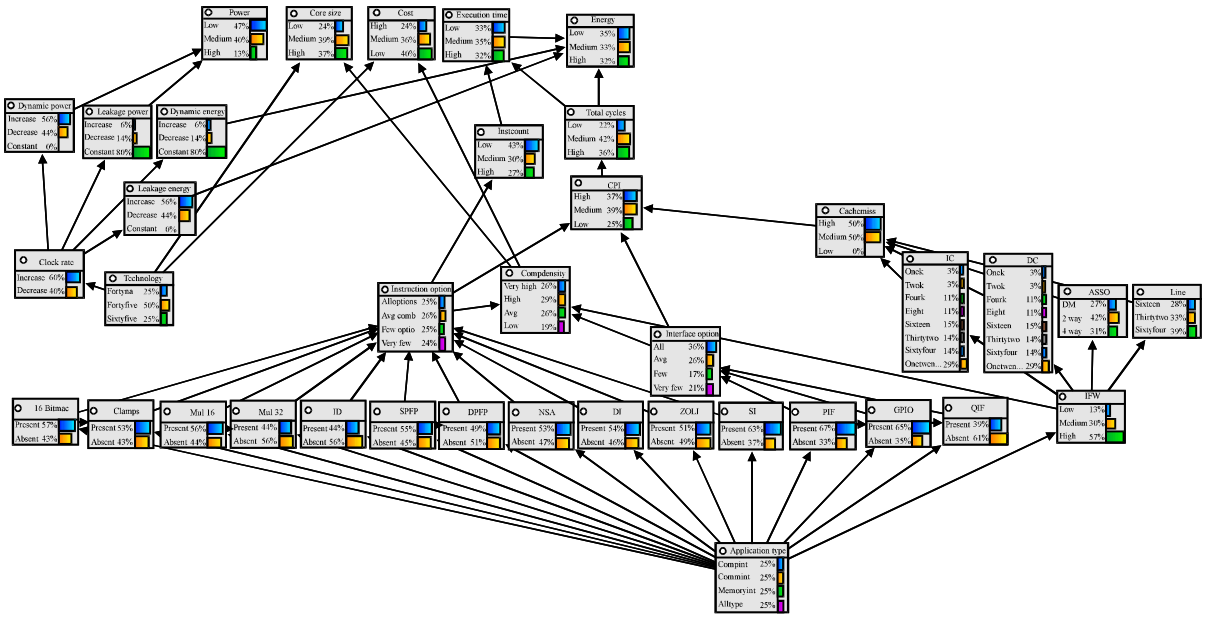

Fig. 2: The predefined probability distributions of the variables in the constructed BBN

Three different interface facilities for enabling the communication between processor to processor and processor to memory have been provided within the Tensilica Xtensa architecture. The combination of the presence or absence of the interface units such as Processor Interface (PIF), General Purpose Input-Output (GPIO) and Queue Interface (QIF) constitutes the available interface options. The Tensilica Xtensa architecture also provides support for the inclusion of the on chip cache memory. Based on the Instruction Fetch Width (IFW), the Instruction Cache (IC) size, Data Cache (DC) size, cache associativity and line size may vary, hence all of these variables in the BBN constitute the available memory options.

Each of the identified instruction units, interface units, memory units and other important influencing factors are represented as individual nodes in the BBN as shown in Fig. 2. The various types of applications considered in the simulation study are represented as states in the node 'application type' which highly influences the selection of the identified instruction options, interface options and IFW. An edge between each of the identified instruction units, interface units, IFW and the node 'application type' signifies that the type of application directly affects the selection of the combination of instruction, interface and memory units for the design.

The nodes 'instruction options', 'interface options' and 'memory options' contribute to the number of transistors needed for implementing the functional units and hence there is a direct edge from each of the these variables to the node 'component density'. The combination of instruction units directly contributes to the number of instruction counts needed for the execution of the given application. The edge between the nodes 'instruction options' and 'instruction count' depicts the causal interdependency between these nodes. The simulation study revealed the effect of instruction cache size, data cache size, cache associativity and cache line size on the variable cache misses. The causal relation between different memory options available and cache misses is represented by the edges from the nodes 'IC', 'DC', 'ASSO' and 'LINE' to the node 'cache misses'.

The impact of instruction, interface and cache misses on the number of cycles spent for executing an instruction that is CPI is represented in the BBN by incorporating edges from the nodes 'instruction options', 'interface options' and 'cache misses' to the node 'CPI'. The total cycles needed for executing the given application depend on the CPI which influence an arc between the nodes 'CPI' and 'Total Cycles'. The technology with which the chip is designed imposes the limitation on the choice of clock rate for driving the functional units within the chip. The choice of clock rate in turn relates to the fundamental parameters of the SoC design such as leakage power, dynamic power, leakage energy and dynamic energy. The causal effects of all of these parameters are represented qualitatively using a direct edge between the respective nodes.

Finally, the variables affecting the system's overall objectives such as power, core size, cost, execution time and energy are represented using causal relationships. The leakage power and dynamic power are directly related to the overall power consumed during the execution of the given application. The variables 'component density' and 'technology' influence the effect on the variables 'core size' and 'cost' which is qualitatively represented by the corresponding arcs between these nodes. The causal effect of the variables 'Total Cycles' and 'Instruction Count' on the variable 'execution time' is represented in the BBN. The effect of 'leakage energy', 'dynamic energy', 'total cycles' and 'execution time' contributes to the overall energy dissipation which is also taken into account in the proposed BBN. Table 2 describes in detail about each node and their corresponding state representation used during the BBN construction and modeling.

Specifying Conditional Probability Table (CPT): The CPT for each node in the constructed BBN is to be specified to represent the causal relation of the variables quantitatively. The CPT is constructed by using both the subjective estimate from the available domain knowledge and simulation statistics. The construction of CPT for all the parent nodes is easier because it is the initial prior probability distribution specification. The difficulty lies in the construction of the CPT for the nodes with number of parents. The CPT for a particular node grows exponentially with respect to the number of parents and the number of associated states within each parent node. Generally, two scenarios are possible when specifying the CPT for each node in the BBN which is described in Table 3.

For example, the node 'instruction option' has eleven parent variables each of which has two states associated with them. Thus, there are 2^{11} total combinations of CPTs that should be filled for the node 'instruction options' which exemplifies the complexity involved in the process of specifying the CPTs during the quantitative representation of the BBN. Similarly, the node 'CPI' has three parent variables, namely 'instruction options', interface options' and 'cache misses'. The corresponding states associated with these parent variables are four, four and three, respectively. Hence, the total combinations of

Table 2: BBN nodes and states used in the embedded system synthesis model

Nodes	Description	States
Aplication type	Applications have been categorized according to their computational complexity and the domain to which they belong to	Computational intensive, communication intensive, memory intensive, all type
MAC16	The MAC 16 option adds multiply-accumulate functions that are useful in DSP and other media processing operations	Present, absent
Clamps	Instruction operation used in conjunction with MUL-16 unit to clamp the result to 16-bit before they are stored in the memory	Present, absent
MUL16	Provides two instructions that perform 16 by 16 multiplication, producing a 32-bit result	Present, absent
MUL32	Provides instructions that perform 32-bit integer multiplication. Consumes more area than MAC 16 and MUL 16 units	Present, absent
ID	Integer divider option provides instructions for 32-bit signed and unsigned integer remainder and quotient operations	Present, absent
SFFP	Single precision floating point operation with 32-bit floating point register file	Present, absent
DPFP	Double precision floating point instruction to accelerate the 64-bit floating point operations	Present, absent
NSA	Implements the normalized shift instruction option	Present, absent
DI	Density Instruction options allow more compact code and can reduce the memory foot print of the application software	Present, absent
ZOLI	Zero overhead Loop option adds the ability to determine the number of iterations before entering into a loop	Present, absent
SI	Synchronize Instruction option provides facilities for shared memory communication between multiple processors	Present, absent
PIF	Processor interface	Present, absent
GPIO	General purpose I/O interface	Present, absent
QIF	Queue interface	Present, absent
IC	Instruction cache	1, 2, 4, 8, 16, 32, 64 and 128 K
DC	Data cache	1, 2, 4, 8, 16, 32, 64 and 128 K
ASSO	Cache associativity	Direct mapped, 2 Way, 4 Way
LINE	Cache line size	16 byte, 32 byte, 64 byte
Instruction option	Instruction Set Architectural (ISA) options	All options, average combinations, few options, very few options
Comp density	Density of the ISA options	Very high, high, average, low
Interface option	Combinations of possible interface units	All options, average combinations, few options, very few options
IFW	Instruction fetch width	low, medium, high
Cache miss	Instruction and data cache misses	Low, medium, high
CPI	Cycles per instruction	Low, medium, high
Inst count	Instruction count	Low, medium, high
Total cycles	Total cycles taken for instruction execution	Low, medium, high
Core size	Total area occupied for the corresponding combinations of the configurations	Low, medium, high
Cost	Cost of die, design cost	Low, medium, high
Execution time	Time taken for the execution of the instructions	Low, medium, high
Energy	Core and memory energy	Low, medium, high
Power	Total power consumption	Low, medium, high
Technology	Manufacturing technology used for the chip design	40 nm, 45 nm, 65 nm
Clock rate	Choice of the clock rate for driving the functional units	Increase, decrease
Leakage power	It is the power consumed by unintended leakage that does not contribute to the Integrated Circuit (IC) function	Increase, decrease, constant
Dynamic power	It is the power consumed during the execution of instructions	Increase, decrease, constant
Leakage energy	It is the energy dissipated by unintended leakage that does not contribute to the Integrated Circuit (IC) function	Increase, decrease, constant
Dynamic energy	It is the energy dissipated during the execution of instructions	Increase, decrease, constant

Table 3: Possible scenarios in specifying the CPT for each node in the BBN

Scenario description	Mathematical representation of the number of possible combination
Number of states associated with each parent variable is equal	Where 'S' represents the number of states in each parent variable and 'V' represents the number of parent variables
Number of states associated with each parent variable is not equal	$\prod_{\substack{i=2\text{ton} \\ j=1\text{tom}}} (S_i)V_j$ Where, 'S_i' represents the number of states in each parent variable and 'V_j' represents the number of parent variables with equal number of states

CPTs for the node 'CPI' will be $4^2 \times 3 = 48$. Table 4 describes an example CPT for the node 'execution time' which is filled using the knowledge obtained from the simulation study.

Once the qualitative and quantitative representations of the BBN are completed then the next step is to validate the constructed BBN through sensitivity analysis for verifying its robustness and consistency. The sensitivity analysis provides the most important information of how much a particular factor influences the decision making process based on the fixed objectives to be met. The measure of mutual information represents the difference of the a priori and a posteriori entropies of a random variable (Li and Wang 2011). Since, entropy reduction has

been utilized and proved to be suitable for sensitivity analysis in Bayesian network (Lee *et al*., 2009). In this study, the concept of mutual information and entropy from the information theoretic principles is utilized to deal with the sensitivity analysis, in order to identify some of the key factors that affect the desired objectives.

The relative influence between uncertain variables within the BBN can be easily identified by measuring the entropy. Table's 5 and 6 show the result of entropy reduction calculation from the sensitivity analysis of factors that affect the objectives core size and energy. For example in Table 6, the factor total cycles has a larger entropy reduction value which gives more certain information on the desired objective than the other factors. Similar entropy reduction calculation is carried out for all the remaining nodes in the constructed BBN to identify the consistencies of the initial probability assessment and their influence over the desired objectives.

Table 4: Example CPT for the node 'execution time' specified using the expert's knowledge

Execution time		Knowledge from simulation study		
Instruction count	Total cycles	Low	Medium	High
Low	Low	0.5	0.3	0.2
Low	Medium	0.6	0.3	0.1
Low	High	0.2	0.3	0.5
Medium	Low	0.3	0.5	0.2
Medium	Medium	0.4	0.5	0.1
Medium	High	0.5	0.4	0.1
High	Low	0.3	0.5	0.2
High	Medium	0.1	0.3	0.6
High	High	0.1	0.2	0.7

Table 5: Entropy reduction values in the BBN with respect to core size

Nodes	I (X:Y)
Comp density	0.11565
Instruction options	0.03507
MAC	0.00880
CPI	0.00879
Inst count	0.00506
Total cycles	0.00273
MUL16	0.00211
Application type	0.00118
Energy	0.00079
ID	0.00077
MUL32	0.00053
SPFP	0.00029
DI	0.00017
Execution time	0.00008

Table 6: Entropy reduction values in the BBN with respect to energy

Nodes	I (X:Y)
Total cycles	0.25862
CPI	0.06983
Instruction options	0.01703
Execution time	0.01161
Comp density	0.00569
MAC	0.00433
Inst count	0.00232
MUL16	0.00102
Core size	0.00079
Application type	0.00058
ID	0.00039
MUL32	0.00027
SPFP	0.00015
DI	0.00010

INFERRING FROM THE NETWORK AND UPDATING THE BELIEF HYPOTHESIS

After constructing the BBN, the behavior of the model has to be observed through imposing evidence on some of the variables. Evidence is imposed on a variable by instantiating any one of its states. The change of the state imposed on some variables in the network, causes the corresponding variables to be instantiated and the effect of this evidence is propagated throughout the network to every other node after compiling the network. This causes every node to update their beliefs which is regarded as the posterior probabilities of the variables after observing some evidence. The following case studies describe the mechanism of propagating the evidence imposed on some of the variables throughout the network and the process of inferring the behavior of the other variables within the network.

Case study 1: In this study, evidence is instantiated by assigning the state as 'medium' for both the variables 'cost' and 'power', resulting in the alteration of belief hypothesis to the entire network. The important variables to be inferred after imposing this evidence are 'component density', 'technology', 'dynamic power' and 'leakage power' which have direct influence on the objective variables 'cost' and 'power', respectively. The observation is that the probability for the average component density has increased from 28-43% and the probability for building the system in the 45 nm technology is 50%. Similarly, the probability of dynamic power to be increased is 67% where as the leakage power remains constant. This shows how the states of other variables are affected and guides the designer to choose average combinations of instructions to attain the specified objective. Figure 3 shows the posterior probabilities of the nodes after imposing the evidence on the variables 'cost' and 'power'.

Case study 2: In the second case, evidence is imposed as 'low', 'low' and 'medium' for the variables 'execution time', 'energy' and 'cache misses' respectively. The observation of the posterior probabilities of certain variables indicates that the instruction options to be of an average combination with 30% probability, the probability

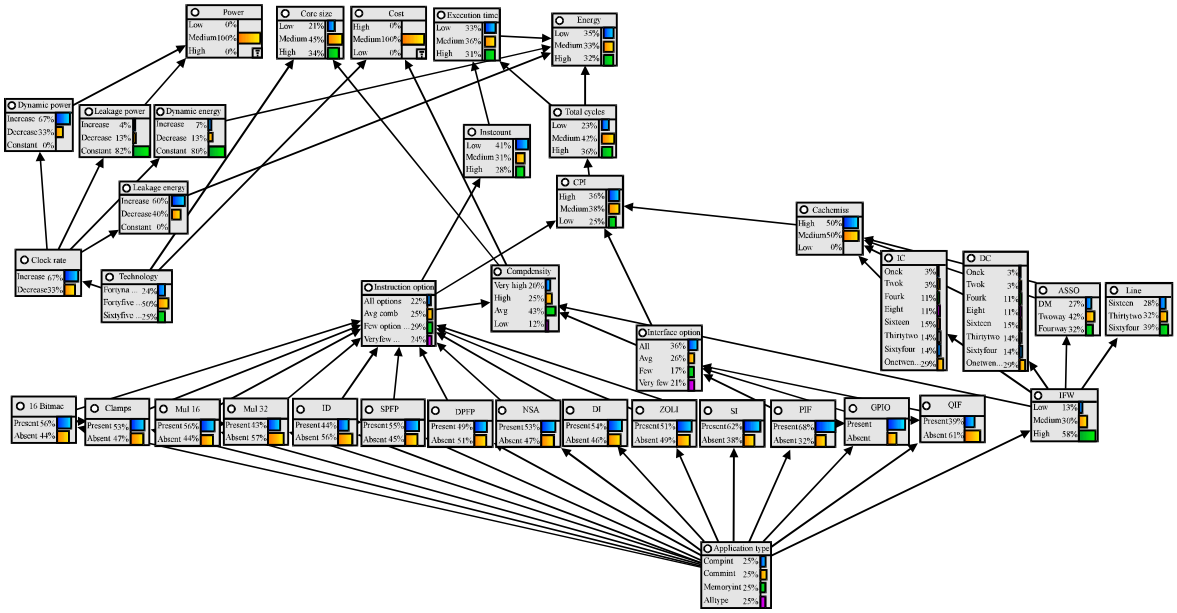

Fig. 3: The posterior probability distributions of the variables for the case study 1

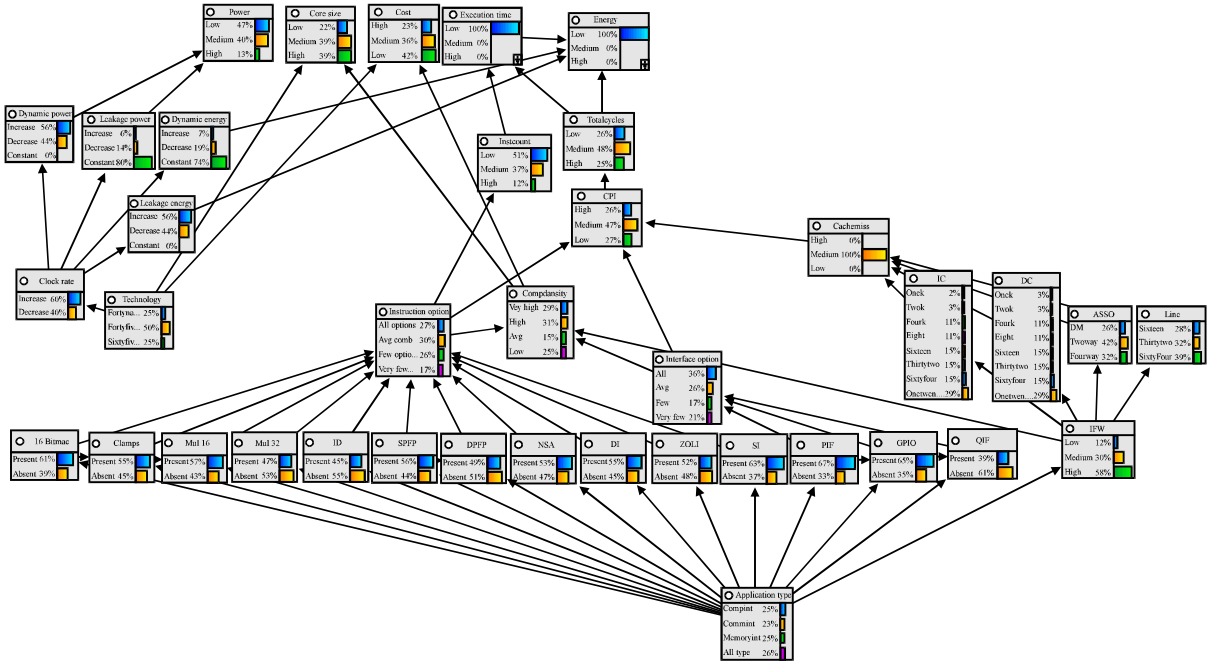

Fig. 4: The posterior probability distributions of the variables for the case study 2

of low instruction count to be 51%, the probability for the total cycles and CPI to be medium at 48 and 47%, respectively. Figure 4 shows the posterior probabilities of the nodes after imposing the evidence on the variables 'execution time' and 'energy'.

Case study 3: This case study reveals the effect of the variables 'instruction options' and 'interface options' on the overall system design. The evidence is imposed by changing the state of the variables 'instruction options' and 'interface options' to 'all options'. The observation of the posterior probabilities of certain variables indicates that the computationally intensive, communication intensive and memory intensive applications requires this combinations with 40% probability. The chance of low power design with respect to this evidence has the

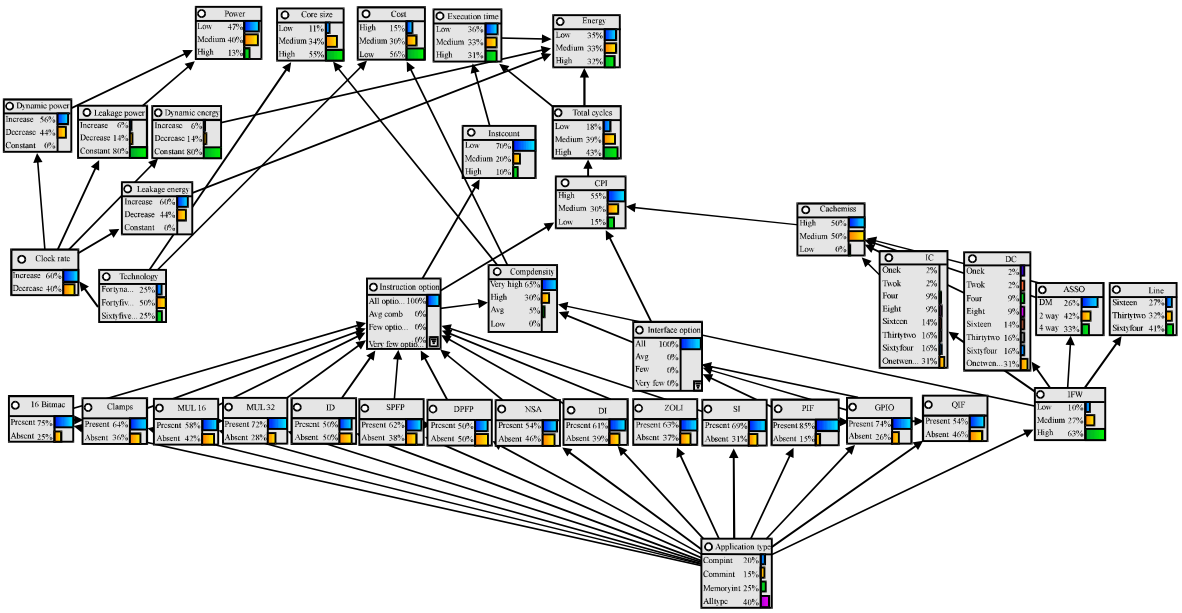

Fig. 5: The posterior probability distributions of the variables for the case study 3

highest probabilistic value. Figure 5 shows the posterior probabilities of the nodes after imposing the evidence on the variables instruction options and interface options.

The three case studies presented above clearly show the mechanism of updating the belief hypothesis and observing the effect of the instantiated evidence on certain variables by inferring the entire network. Thus, the reasoning capability of the BBN can be exploited during the process of embedded system design. The primary finding of this modeling framework is the ability of the BBN to reflect the interdependencies of the key architectural attributes in the embedded system design process. The ability of the BBN to inculcate the probabilistic reasoning with high degree of precision on the uncertain factors of the system being modeled is one of the important aspects in utilizing them for decision making.

CONCLUSION

Efforts to develop complete, well structured and easily adaptable techniques for automating the embedded system design process will benefit the EDA community to take early design decisions. The BBN based DSS framework proposed in this study utilizes both the domain knowledge of the target architecture and empirical results obtained through a set of simulation studies to build the model. The probabilistic reasoning capability of the BBN under uncertain situations is considered to be the most important aspect in utilizing the BBN for the embedded system synthesis. Representation of the causal interdependencies among the system variables and the inference mechanism provided for easy analysis of the performance of the system being modeled under various constraints, moves the embedded system design a step forward towards completely automating the design process.

RECOMMENDATIONS

The future enhancements that are possible with this proposed framework will be in the construction of more realistic models by considering all possible system parameters and constraints to be satisfied. Different inference algorithms can be used for enabling the network to learn and to predict the performance of the system. Different techniques can be exploited for specifying the CPTs automatically for each of the variables used in the design.

REFERENCES

Aleti, A. and I. Meedeniya, 2011. Component deployment optimisation with bayesian learning. Proceedings of the 14th international ACM Sigsoft symposium on Component based software engineering, June 21-23, 2011, Boulder, Colorado, USA., pp: 11-20.

Beltrame, G., L. Fossati and D. Sciuto, 2010. Decision-theoretic design space exploration of multiprocessor platforms. Trans. Comput. Aided Des., 29: 1083-1095.

Blythe, S.A. and R. Walker, 2000. Efficient optimal design space characterization methodologies. ACM Trans. Des. Automation Elect. Syst., 5: 322-336.

Burger, D. and T.M. Austin, 1997. The simplescalar tool set, version 2.0. University of Wisconsin-Madison Computer Sciences Department Technical Report No. 1342, June 1997, pp: 1-21. http://www.eecg.toronto.edu/~moshovos/ACA05/doc/ simplescalar.pdf.

Chen, Y.K. and S.Y. Kung, 2008. Trend and challenge on system-on-a-chip designs. J. Signal Process. Syst., 53: 217-229.

Daly, R., Q. Shen and S. Aitken, 2011. Learning bayesian networks: Approaches and issues. Know. Eng. Rev., 26: 99-157.

Darwiche, A., 2010. Bayesian networks. Commun. ACM., 53: 80-90.

Delcroix, V., K. Sedki and L. Francois-Xavier, 2013. A bayesian network for recurrent multi-criteria and multi-attribute decision problems: Choosing a manual wheelchair. Exp. Syst. Appli., 40: 2541-2551.

Eisenring, M., L. Thiele and E. Zitzler, 2000. Conflicting criteria in embedded system design. Des. Test Comput., 17: 51-59.

Fang, C. and F. Marle, 2012. A simulation-based risk network model for decision support in project risk management. Decis. Support Syst., 52: 635-644.

Fredriksson, J., K. Sandstrom and M. Akerholm, 2005. Optimizing resource usage in component-based real-time systems. In: Proceedings 8th International Symposium on Component-Based Software Engineering, May 14-15, 2005, St. Louis, MO., USA., Heineman, G.T., I. Crnkovic, H.W. Schmidt, J.A. Stafford, C. Szyperski and K. Wallnau (Eds.). Volume, 3489, Springer Berlin Heidelberg, USA., pp: 49-65.

Gries, M. and K. Keutzer, 2005. Building ASIP's: The MESCAL Methodology. Springer, New York, USA.

Gruber, A. and I. Ben-Gal, 2012. Efficient bayesian network learning for system optimization in reliability engineering. Qual. Technol. Quant. Manage., 9: 97-114.

Heckerman, D., 1997. Bayesian networks for data mining. Data Mining Knowl. Discovery, 1: 79-119.

Kahn, G., 1974. The semantics of a simple language for parallel programming. Proceedings of IFIP Congress, August 5-10, 1974, Stockholm, Sweden, pp: 471-475.

Kempf, T., G. Ascheid and R. Leupers, 2011. Multiprocessor System on Chip: Design Space Exploration. 1st Edn., Springer, Berlin, Germany.

Korb, K. and A. Nicholson, 2004. Bayesian Artificial Intelligence. CRC Press, USA.

Lauritzen, S. and D. Spiegelhalter, 1988. Local computations with probabilities on graphical structures and their application to expert systems. J. Royal Stat. Soc. Ser. B., 50: 157-224.

Lee, C., M. Potkonjak and W.H. Mangione-Smith, 1997. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. Proceedings of the 13th Annual International Symposium on Microarchitecture, December 1-3, 1997, Research Triangle Park, NC., pp: 330-335.

Lee, E., Y. Park and J.G. Shin, 2009. Large engineering project risk management using a bayesian belief network. Exp. Syst. Appl., 36: 5880-5887.

Lee, E.A., 2003. Overview of the Ptolemy project. Technical Report UCB/ERL M03/25, University of California, USA.

Li, M., R. Sasanka, S. Adve, Y.K. Chen and E. Debes, 2005. The ALPBench benchmark suite for complex multimedia applications. Proceedings of the International Workload Characterization Symposium, October 6, 2005, Austin, TX., USA., pp: 34-45.

Li, M. and L. Wang, 2011. Feature fatigue analysis in product development using bayesian networks. Exp. Syst. Appli., 38: 10631-10637.

Lieverse, P., P. van der Wolf, K. Vissers and E. Deprettere, 2001. A methodology for architecture exploration of heterogeneous signal processing systems. J. VLSI Sig. Process. Syst. Signal, Image Video Technol., 29: 197-206.

Mueller, W., J. Ruf, D. Hoffmann, J. Gerlach, T. Kropf and W. Rosenstiel, 2001. The simulation semantics of systemC. Proceedings of the International Conference on Design Automation Test in Europe, March 12-16, 2001, Munich, pp: 64-70.

Nadkarni, S. and P.P. Shenoy, 2004. A causal mapping approach to constructing Bayesian networks. Decis. Support Syst., 38: 259-281.

Neapolitan, R., 2003. Learning Bayesian Networks. Prentice Hall, USA.

Olson, J.T., J.W. Rozenblit, C. Talarico and W. Jacak, 2007. Hardware/software partitioning using bayesian belief networks. Trans. Syst. Man and Cyber., 37: 655-668.

Parakhine, A., J. Leaney and T. O'Neill, 2008. Design guidance using simulation-based bayesian belief networks. Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, March 31-April 4, 2008, Belfast, pp: 76-84.

Paulin, P.G., C. Pilkington and E. Bensoudane, 2002. StepNP: A system-level exploration platform for network processors. Des. Test Comput., 19: 17-26.

Pimentel, A.D., L.O. Hertzbetger, P. Lieverse, P. van der Wolf and E.E. Deprettere, 2001. Exploring embedded-systems architectures with Artemis. Computers, 34: 57-63.

Shankaran, N., J. Balasubramanian, D. Schmidt, G. Biswas, P. Lardieri, E. Mulholland and T. Damiano, 2006. A framework for (re) deploying components in distributed real-time embedded systems. Proceedings of the ACM Symposium on Applied Computing, April 23-27, 2006, Dijon, France, pp: 737-738.

Thiele, L., S. Chakraborty, M. Gries and S. Kunzli, 2002. A Framework for evaluating design tradeoffs in packet processing architectures. Proceedings of the 39th International Conference on Design Automation, June 2002, New Orleans, LA., pp: 880-885.

Vila-Frances, J., J. Sanchis, E. Soria-Olivas, A.J. Serrano, M. Martinez-Sober, C. Bonanad and S. Ventura, 2013. Expert System for Predicting unstable angina based on Bayesian networks. Exp. Syst. Applic., 40: 5004-5010.

Watthayu, W. and Y. Peng, 2004. A Bayesian network based framework for multi-criteria decision making. Proceedings of the 17th International Conference on Multiple Criteria Decision Analysis, August 6-4, 2004, Canada, pp: 6-11.

Xu, B.G., 2012. Intelligent fault inference for rotating flexible rotors using bayesian belief network. Exp. Syst. Applic., 39: 816-822.

Yet, B., K. Bastani, H. Ragarjo, S. Lifvergren, W. Marsh and B. Bergman, 2013. Decision support system for Warfarin therapy management using Bayesian networks. Decision Support Syst., 55: 488-498.

Zhang, L., X. Wu, L. Ding and M.J. Skibniewski and Y. Yan, 2013. Decision support analysis for safety control in complex project environments based on bayesian networks. Exp. Syst. Appl., 40: 4273-4282.