Generation of a Reconfigurable Probabilistic Decision-Making Engine based on Decision Networks: UAV Case Study

Sara Zermani
Université de Guyane, Espace-Dev, UMR 228, Cayenne, France
Sara.Zermani@gmail.com

Catherine Dezan
Université de Brest, Lab-STICC, CNRS, UMR 6285, Brest, France
Catherine.Dezan@univ-brest.fr

Abstract

Making decisions under uncertainty is a common challenge in numerous application domains, such as autonomic robotics, finance and medicine. Decision Networks are probabilistic graphical models that propose an extension of Bayesian Networks and can address the problem of Decision-Making under uncertainty. For an embedded version of Decision-Making, the related implementation must be adapted to constraints on resources, performance and power consumption. In this paper, we introduce a high-level tool to design probabilistic Decision-Making engines based on Decision Networks tailored to embedded constraints in terms of performance and energy consumption. This tool integrates high-level transformations and optimizations and produces efficient implementation solutions on a reconfigurable support, with the generation of HLS-Compliant C code. The proposed approach is validated with a simple Decision-Making example for UAV mission planning implemented on the Zynq SoC platform.

2012 ACM Subject Classification Computer systems organization → Self-organizing autonomic computing; Computer systems organization → Embedded hardware

Keywords and phrases Decision networks, Bayesian networks, HLS, FPGA

Digital Object Identifier 10.4230/OASIcs.ASD.2019.9

Category Interactive Presentation

Funding This work is supported by CNRS fundings through the PICS project SWARMS.

1 Introduction

Embedded Decision-Making is necessary in a number of contexts including medical applications and autonomous vehicles. Embedded solutions make it possible to adapt to the constraints of a real-time response that would be impossible with centralized off-board decision making due to the need for communication media access. For instance, in the case of unmanned aerial vehicles (UAVs) or intelligent vehicles, on-board decision making based on image recognition enables real-time responses to propose appropriate action (e.g., to continue tracking or to dismiss) in an autonomous manner. Many examples of embedded decisions have been recently tested [26] [20] [6].

Among the techniques available for Decision-Making, three approaches have recently emerged in the literature [3]: 1) Multicriteria Decision-Making techniques, 2) Mathematical Programming techniques and 3) Artificial Intelligence. Nevertheless, to deal with uncertainty of the environment and of the system (external or internal hazards), fuzzy techniques [2] or stochastic/probabilistic models such as Bayesian Networks [21] are used. In this paper, we focus on Decision Networks (also called Influence Diagrams), which are considered as an

extension to Bayesian Networks (BN), including a Decision-Making (DM) mechanism based on utility tables. Compared with a fuzzy model, a Decision Network (DN) is specified by a causality graph that encapsulates the expert knowledge of the system and provides a good comprehensive and efficient formulation for the designer.

For an embedded version of a Decision-Making mechanism, the following features are of major importance: 1) achieving (real-time) performance under the constraints of memory or computation related to the embedded system, 2) ensuring quality of service under power consumption constraints. For Bayesian Networks, literature on embedded implementation can be found for both software [22] and hardware versions, the latter with reconfigurable hardware [16] [18] [25] using Field-Programmable Gate Array (FPGA). In [25], the authors propose BN in reconfigurable hardware, but the decision mechanism that integrates temporal specification in temporal logic is operated by the embedded processor. For embedded DN, the ARPHA framework [22] [7] makes it possible to design specific failure scenarios from fault tree specifications. ARPHA generates an embedded software version of a DN, but does not propose any hardware alternative. To the best of our knowledge, no hardware implementation of DN on reconfigurable platforms have been proposed as alternative embedded solutions.

In this paper, we propose a design tool to generate a reconfigurable implementation of Decision Networks from high-level specifications of a Decision-Making engine. This is the first design tool that proposes automatic generation of Decision Networks on FPGA. The main contributions are: 1) the specific High Level Synthesis (HLS) transformations to produce a HLS-Compliant C Code, 2) the generation of adequate HLS directives for efficient implementation on an FPGA/SoC (System On Chip) platform.

The paper is organized as follows. Section 2 gives an introduction to Decision Networks. Section 3 introduces the design tool to generate the embedded Decision-Making engine from DN specifications, with a specific focus on the dedicated HLS transformations and HLS directives for an efficient implementation on the FPGA/SoC platform. Section 4 presents a case study that validates the flow up to implementation on a Zynq platform.

2 Background on Decision Networks and Probabilistic Beliefs

Decision Networks extend Bayesian Networks to provide a mechanism for making rational decisions by combining probability and utility theory. In addition to chance nodes defined by a BN, a DN also includes utility and decision nodes. Decision nodes represent the set of choices open to the decision maker, while utility nodes are used to express preferences among possible states of the world represented by the chance nodes and decision nodes.

2.1 Bayesian networks for probabilistic beliefs

BNs are probabilistic graphical models used to understand and control the behaviour of systems [8] by providing diagnoses. They are composed of nodes and oriented arcs between nodes representing the knowledge expertise of the system. In Fig. 1 a), we propose to evaluate the probability of a UAV increasing or decreasing its altitude (U_A) based on the information given by two sensors: the altimeter sensor (S_A) and the barometer sensor (S_B). The BN nodes of the networks represent random variables whose values can depend on specific states, and the arcs of the network indicate the conditional dependencies represented by the conditional probabilities defined with probability tables (CPTs). In the example, each node has two states, represented by the values *inc* and *dec*. The probabilities of the BN are also known as the parameters of the network.

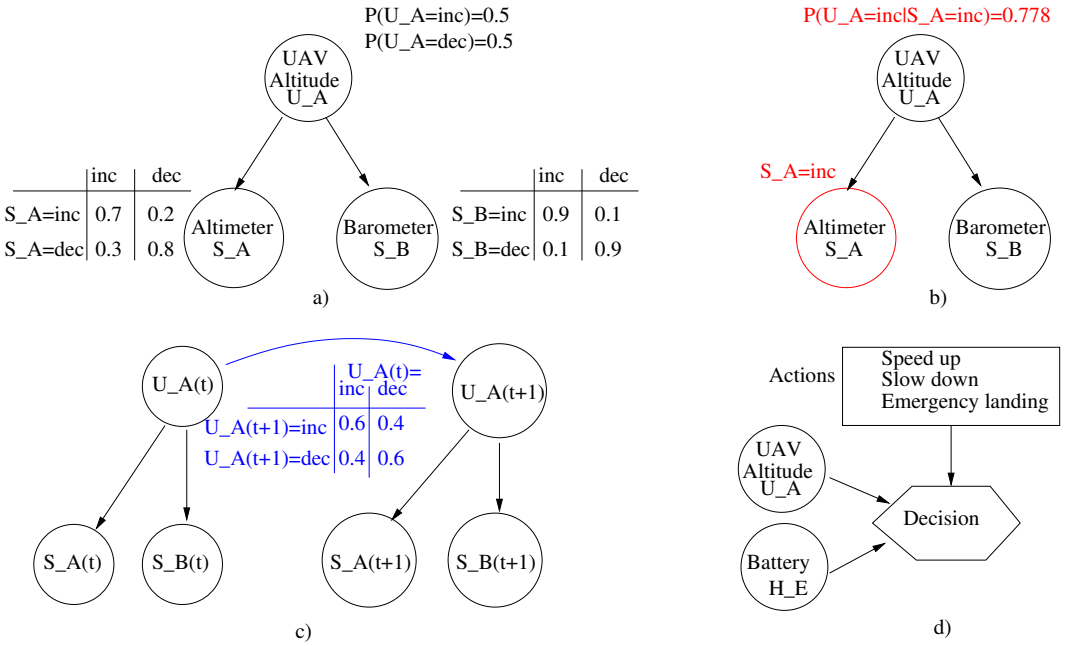

Figure 1 BN principle and DN example: a) BN example, b) Inference illustration, c) Dynamic BN, d) DN example.

To obtain the probability of one variable being in a specific state (A), we use an inference mechanism that takes into account some observations (evidence indicators) of the system in order to compute the posterior probabilities over A using Bayes' theorem (Eq. 1).

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A) \tag{1}$$

We use Bayes' theorem for our example (illustration in Fig. 1 b) to compute the probability of the variable U_A being in a state "increasing" by taking into account the value of the altimeter S_A as evidence.

This is done as follows:

$$\begin{aligned} P(U\_A = Inc|S\_A = Inc) &= \frac{P(S\_A = Inc|U\_A = Inc)P(U\_A = Inc)}{P(S\_A = Inc)} \\ &= \frac{0.7 * 0.5}{0.7 * 0.5 + 0.2 * 0.5} = 0.778 \end{aligned}$$

We can thus say that the probability of the altitude status being "increasing" is equal to 0.778. If, for example, we add observations from the barometer sensor, also giving it an increasing value, the probability $P(U\_A = Inc|S\_A = Inc, S\_B = Inc)$ increases to 0.969.

The probabilities can change over time. So it could be more appropriate to use dynamic BN to model the change. In the case of a UAV, if we have a greater chance of maintaining an increasing altitude if the UAV is already in this situation, and can model this with time-dependent variables as proposed in Fig. 1 c).

As BN are dedicated to computing the probabilities of the states, we need an extra mechanism to express the decision making that takes into account these values for the choice of appropriate actions to safely continue a mission. We therefore use the Decision Networks for the Decision-Making process.

2.2 Example of a Decision Network

DN are directed acyclic graphs (DAG) with nodes belonging to three different categories: chance nodes (ellipses or circles in graphical notation), which represent (as in BN) discrete random variables; decision nodes (rectangles), which represent actions or decisions with a predefined set of states; and value nodes (diamonds), which represent utility (or cost) measures associated with random or decision variables. Edges represent direct (possibly causal) influence between connected objects. An example of Decision-Making (see Fig. 1 d)), linking the two BNs (representing the probabilities of the UAV Altitude state and of the well-functioning Battery state), and the possible actions (Speed Up action, Slow Down action and Emergency Landing action) to be chosen with the respect of a utility table. The utility table (detailed in the Table 1) gives a score for each action relative to the BNs, and the choice of the adequate action is given by computing a utility function (U_f) for each action.

Table 1 Utility table for decision making with DN associated with the actions Speed Up (SU), Slow Down (SD) and Emergency Landing (EL).

UAV Alt. (U_A)	inc						dec					
Battery (H_E)	ok			bad			ok			bad		
Actions (A)	SU	SD	EL	SU	SD	EL	SU	SD	EL	SU	SD	EL
U	100	0	0	0	0	100	0	100	0	0	0	100

To compute the utility function we need the probabilities provided by the BNs. The utility function of each action is equal to the sum of the products of the action with the adequate probability concerning the UAV state and the battery state. The action to be chosen is the action that has the highest utility function.

Let us consider the example of Fig. 1 d):

$\text{Action\_}to\text{\_activate} = Max(U\_f_{SU}, U\_f_{SD}, U\_f_{EL})$

where each $U\_f_k (k = \{SU, SD, EL\})$ is equal to

$$U\_f_k = \sum_i \sum_j U(A = k, U_A = i, H_E = j) * P(U_A = i) * P(H_E = j)$$
$$(i = \{inc, dec\} \quad and \quad j = \{ok, bad\})$$

In this example, if we take the BN probabilities $P(U_A = inc) = 0.9$ and $P(H_E = ok) = 0.8$ and the utility table in Table 1, then $U\_f_{SU} = 72$, $U\_f_{SD} = 8$ and $U\_f_{EL} = 20$, which means that the action chosen is SU ("Speed up") this case.

3 Design Tool Proposal

The proposed design tool (Fig. 2) incorporates the two following main layers:

1. In the first layer, the Bayesian core tool takes a DN as input. The DN specifications can be expressed in *.net* format or *.m* format to make them compatible with other tools such as Genie [14] or BNT [19] for Matlab. First, a series of dedicated high-level

transformations are proposed for the BN part: AC compilation based on model patterns and evidence optimization. Then, optimizations on the whole DN part are proposed, such as bitwidth optimization and a functional/structural decomposition based on the choice of the elementary function for a hierarchical decomposition before the generation of the C-code.

2. In the second layer, a refinement of C-code is proposed by introduction of HLS directives for code parallelization, memory and interface management. This latter C-HLS compliant code is tailored for complete FPGA implementation on ZedBoard.

The first layer proposes high-level transformations and provides parallel opportunities for the code, independent of the target platform. The second layer gives a more practical guide for parallel implementation on a FPGA/SoC platform. This second layer is platform dependent. In this presentation, we focus on specific high-level transformations for BN and on the generation of C-code in order to show the ability of the design tool to generate parallelism. We therefore do not detail the bitwidth adaptation and functional/structural decomposition. The probabilities are defined here in a floating-point representation in the different examples of Section 4, but they can easily be limited to fixed-point representation, thus saving some FPGA resources at the same time.

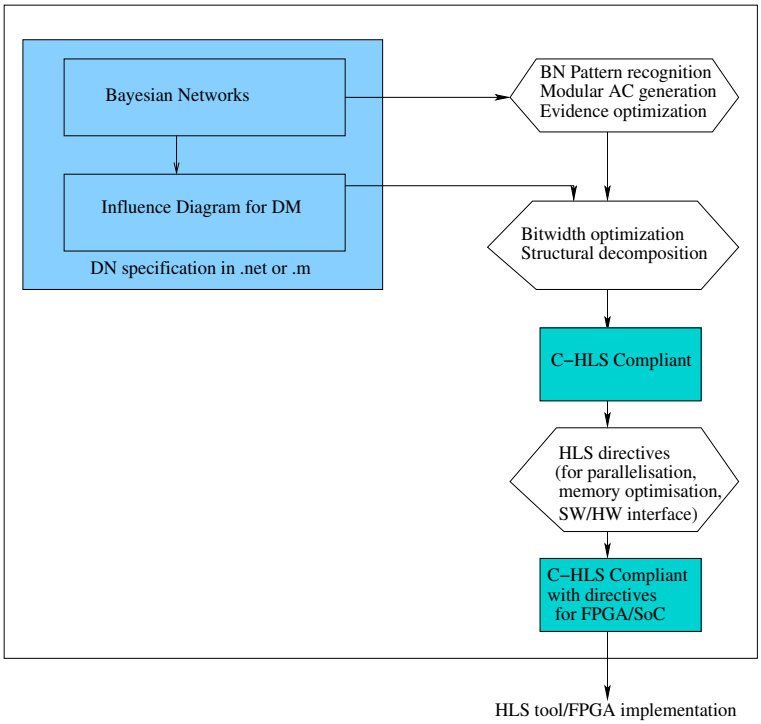

Figure 2 Proposed HLS Design tool for Decision Networks on an FPGA support.

3.1 Bayesian core tool for DN: Dedicated high-level transformations for DN

In this section, we start with an introduction to the intermediate representation (arithmetic circuit compilation) used to synthesize the BN specifications. High-level transformations are then proposed to generate a synthesizable C-code for DN specifications.

3.1.1 Arithmetic circuit (AC) compilation for BN inference

BN inference algorithms are used to answer queries when computing posterior probabilities. Classical inference algorithms are based on propagation on the junction tree. A major problem for embedded systems is the complexity of the computation. Algorithms based on ACs are powerful and can produce predictable real-time performances [4] [10].

The AC representation of BN can be built from the multilinear function (MLF) f [11] associated with the marginal probabilities of the BN (Figure 3). The leaves of the arithmetic circuit are λ (evidence indicator) and θ (network parameter), and the inner nodes of the tree represent a multiplication (*) or an addition (+), alternately. To compute the posterior probability $P(x|e)= \frac{f(x,e)}{f(e)}$ (where x is a variable and e the evidence) for the diagnostic, two steps are required: the first to evaluate *f(e)* and the second to compute the circuit derivatives to obtain *f(x,e)*$= \frac{\partial f}{\partial \lambda_x}$.

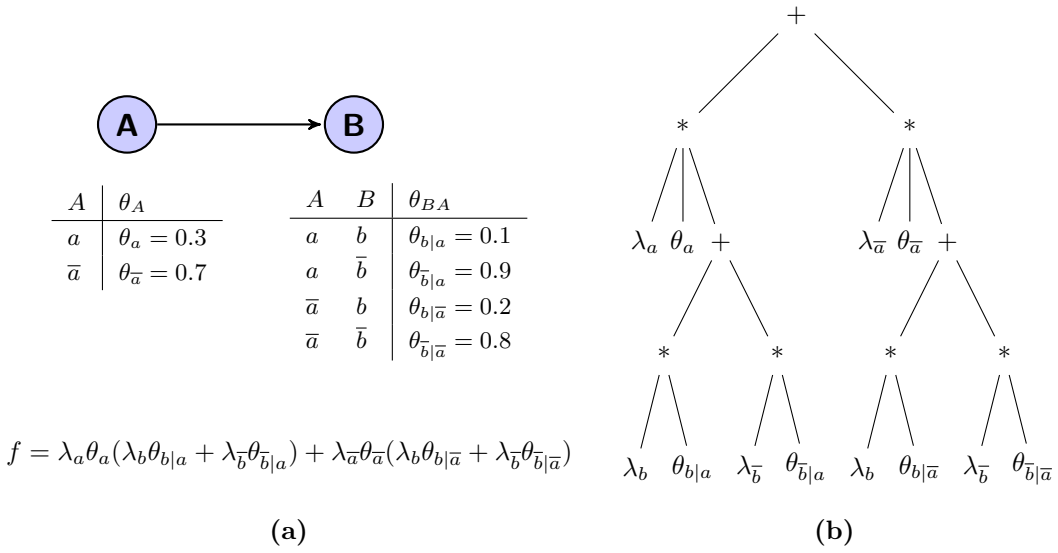

Figure 3 (a) A Bayesian network with a multilinear function. (b) The corresponding arithmetic circuit.

3.1.2 Modular AC generation associated with model patterns

It is possible to simplify the generation of AC if the BN structures correspond to identified patterns. For instance, in the examples of Section 4, the structure of BN is clearly repetitive and matches the pattern we named the FMEA_HM pattern, described in Fig. 4. Other patterns can also be used, like the SWHM (software Health Management) model proposed in [24]. The BN of FMEA_HM pattern evaluates the probability of well functioning for a specific item in the system providing the diagnosis, also called HM (Health Management).

In such cases, the generation of the AC uses a modular approach of inference computation by taking advantage of the factorization of the MLF and the regular structure associated with the pattern. In the case of an FMEA_HM pattern, the AC MLF factorization is based on the relationship between child and parent nodes. The sub-MLFs for child nodes (if there is no conditional dependence between them) are represented by a + in the AC, and parent nodes combine them with a *. In this model, all error type nodes and all monitor nodes are children of the U node (unobservable status of the system). Since there is no conditional

dependence between these nodes, their MLF can be calculated separately and in parallel, similar to the sub-BNs of each error type, where the parent node is the error type and the child nodes are the monitor and appearance context nodes. This allows us to have a hierarchical and modular AC.

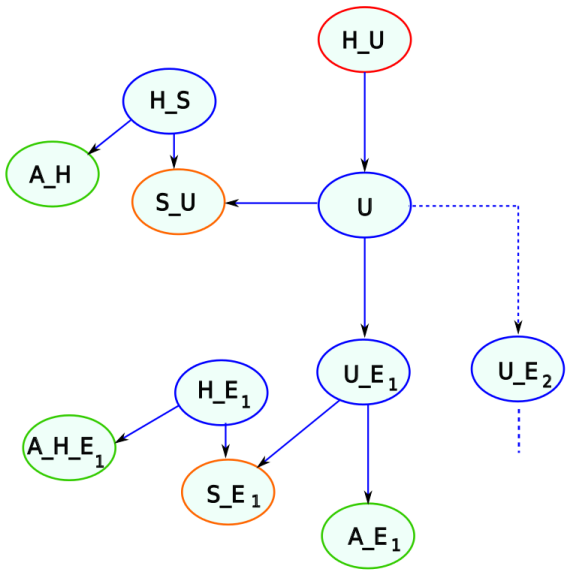

Figure 4 FMEA_HM pattern for BN where the S_U, S_Ei nodes represent the sensor nodes of either a hardware or a software monitor, the nodes U, U_Ei are the internal states possibly affected by an error Ei, the H_U node represents the health of the system, H_S and H_Ei nodes represent the health of the sensor, and the A_H, A_H_Ei are the appearance contexts.

This principle can be easily extended to Dynamic Bayesian Networks by considering the temporal variables.

3.1.3 BN optimization based on evidence

In an AC, we can see that the values $(\lambda_x, \lambda_{\bar{x}})$ of evidence of a variable X are equal to:

- $(\lambda_x, \lambda_{\bar{x}}) = (1, 1)$ when there is no observation on the variable X,
- $(\lambda_x, \lambda_{\bar{x}}) = (1, 0)$ when the evidence is on the variable X and the observation is x,
- $(\lambda_x, \lambda_{\bar{x}}) = (0, 1)$ when the evidence is on the variable X and the observation is $\bar{x}$.

Furthermore, in our examples in Section 4, two types of node (observable and unobservable) are known: C, A and S nodes are observable (evidence), so they take the values $(1, 0)$ or $(0, 1)$ for evidence. The other nodes are unobservable, so they take the values (1,1) for evidence. These observations and the symmetrical structure of the AC make it possible to reduce the AC as follows:

- Delete the left (or right) topmost branch containing a C (or an H) node, and in all sub-branches where C (or H) appears, replace the probability parameters by a choice between the right or left value. We can simplify here, because the value of $(\lambda_x, \lambda_{\bar{x}})$ of these nodes is never equal to 1 at the same time,
- Repeat this procedure for all C and H nodes,
- Fix all the λ_x values for the unobservable nodes at $(1, 1)$.

3.1.4 C-synthesizable code for DN

The decision-making approach is based on the utility function equation, taking the HM results from BNs, actions and utility table as input. The C-synthesizable code is generated for BN in two ways: a) in a hierarchical way, by choosing several kinds of elementary block, or b) in a flat way. In a hierarchical version, the elementary blocks are chosen considering the structure of the AC tree.

The C-synthesizable code of the DN is given by the following algorithm:

Algorithm 1 Decision Making.

Require: Proba from BNs HM_1, HM_2, ..., HM_n, actions A_1, A_2, ..., A_n and the utility table U

for all states i_1 of HM_1 **do**
  ...
  for all states i_n of HM_n **do**
    for all actions A_k **do**
      // Compute utility function for each action

$$U_{f\_A_k} = \begin{array}{l} U_{f\_A_k} + U(A_k, HM_1 = i_1, ..., HM_n = i_n) \\ *P(HM_1 = i_1) * ... * P(HM_n = i_n) \end{array}$$

    end for
  end for
  ...
end for
return Maximum of $U_f\_A_k$

3.2 Generation of HLS directives for a SoC implementation on ZedBoard

3.2.1 ZedBoard target for SoC/FPGA implementation

Our design approach is characterized by the separation of processing components from functional programmable components. The proposed design targets the ZedBoard incorporating a hybrid Zynq processor [9]. As shown in Figure 5, the architecture is built around the ARM Cortex-A9 processor (Zynq processing system PS). The processor communicates with dedicated HW accelerators using programmable logic through an Advanced eXtensible Interface (AXI) bus.

For the SoC implementation on ZedBoard, numerous HLS directives have been proposed [1]. Here, we list only the main ones used for this experimentation.

3.2.2 Parallelization directives at function calls and for loops

In order to increase the parallelism, the following main directives are chosen:

1. **INLINE:** Inlines this function call (does not create a separate level of RTL hierarchy) and allows resource sharing and optimization across hierarchy levels.
2. **UNROLL:** Duplicates computation inside the loop and increases computation resources, decreases number of iterations.
3. **PIPELINE:** Pipelines computation within the loop (or region) scope and increases throughput and computation resources.

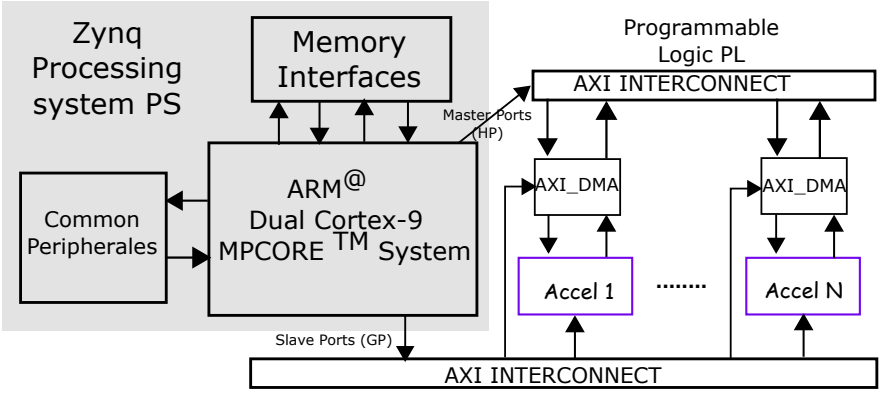

Figure 5 The hybrid architecture of the Zynq processor for SoC implementation.

Most of the time, the resources available for the IP are limited because other applications can share the resource. So the parallelization directives are used considering resource constraints.

3.2.3 Memory management directives

For the storage of the parameters of the Bayesian network, two main options are used for an embedded version inside the programmable logic component:

1. **BRAM:** with the directives *array_map* or *array_partition*
2. **LUT:** default option

The data organization should be addressed correctly taking into account the interface mechanism.

3.2.4 Interface management

The inputs of the SoC are the evidence of the networks that are data provided by sensors. These are stored in an external memory (DDR). As for the evidence of the networks, the parameters of the BN are inputs of the IP and can be stored either inside the SoC (BRAM, LUT or CACHE) or outside in an external memory. This choice depends on the designer's needs and on the possible changes of the network parameter values. To access the external memory, different interface options are possible:

1. **STREAM:** AXI stream
2. **DMA:** Master DMA

4 Case Study of a Simple UAV Mission Plan

In this section, we present a case study that validates the design flow up to implementation on the Zynq platform. In this section we consider a simple UAV mission that address two main failure scenarios; one related to GPS failure and the other to the battery failure. Both can be expressed by BN, considering the FMEA_HM pattern.

4.1 Bayesian networks for the health of the GPS receiver and of the battery

The accuracy and reliability of the position given by a GPS depend on contextual factors affecting the satellite signal during its propagation or its reception. The sources of error can be identified at the system level by means of additional bias in the computation of

pseudo-range measurements [17] [23]. These measurements can tune the GPS positioning accuracy from slightly imprecise to completely faulty. They can be improved by introducing observations [12] or real environments [13]. The main GPS localization errors are illustrated in Fig. 6 with a BN.

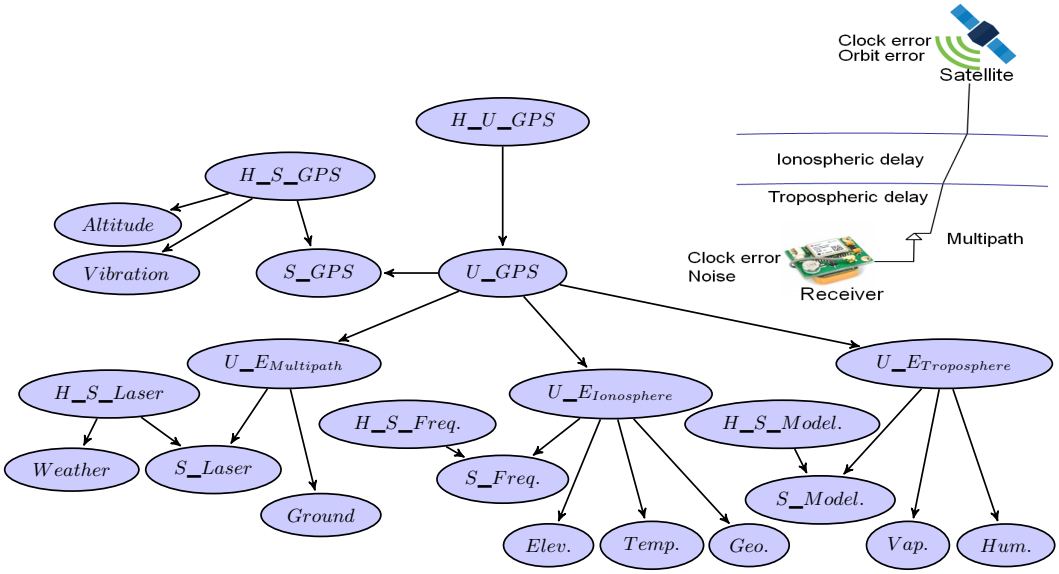

Figure 6 GPS potential errors and BN description.

To model battery behaviour, we use a dynamic BN to represent the linear progression of the energy consumption over time.

This dynamic Bayesian network is described at two time steps (2TBN, cf Figure 7) to take into account the previous past value. External events such as strong wind can increase the energy consumption, as may any application/component used in a period of time. Each status of an application (U node) is associated with a command node (C) that represents an action to enable or disable the application. The battery level is given by a sensor that can fail; its health is reinforced by the appearance contexts (low temperature, for example).

4.2 Decision making with an influence diagram

Figure 8 shows the decision-making mechanism of the mission considering the two cases of failure (GPS failure (GPS HM), Battery failure (Energy HM)) and three actions (Nothing to report, Change localization method, Emergency Landing). This figure illustrates the monitoring of the GPS HM, the energy consumption HM and the DM at each time. Figure 8.(a) shows the interest of the context appearance nodes, which reinforces the confidence in error types and sensor health. For example, from time $t = 0$ until $t = 4$, we observe no problem in the GPS localization and, due to the evidence on appearance contexts, the probability of the health of the system grows from 0.835 to 0.915. At time $t = 4$, a problem in the GPS localization is observed. Without observation on the appearance context nodes, the probability of the health of the system decreases to 0.365, but according to the evidence on appearance contexts it decreases to 0.143. Figure 8.(b) shows monitoring in a nominal case for energy consumption. According to observation of sensors and appearance contexts, the energy consumption is healthy but decreases over time because the related Bayesian

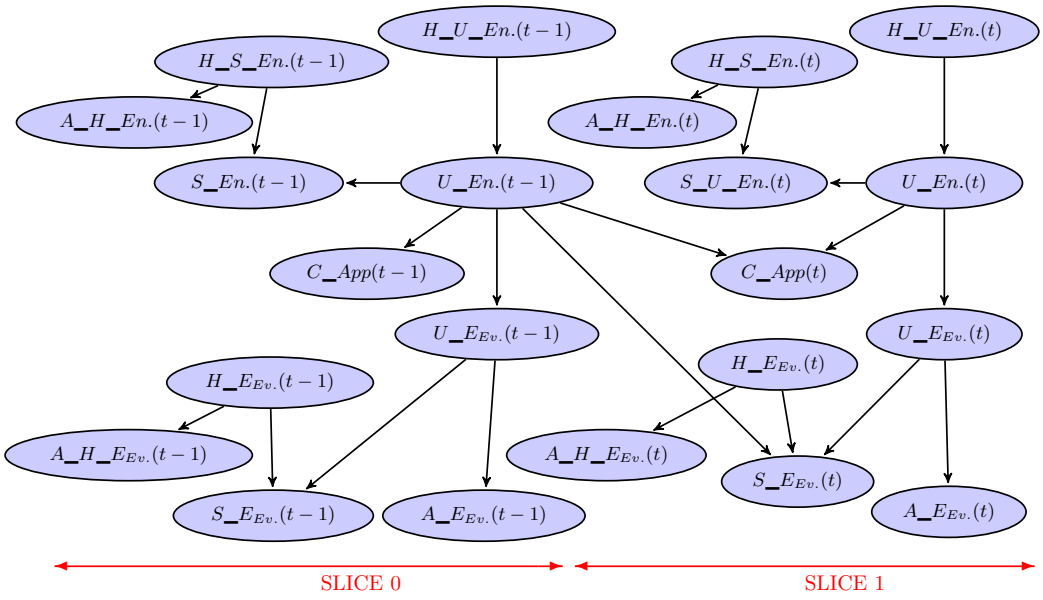

Figure 7 2TBN for Health Management for the battery.

Network evolves dynamically. Figure 8.(c) shows the evolution of Decision Making over time based on the health probability of the GPS localization and energy consumption. From time $t = 0$ until $t = 4$, the maximum of the utility functions is "nothing to report", which reflects the case of "no problem in the mission". At time $t = 4$, the maximum of the utility functions is "change localization method", which reflects a problem in the mission, i.e., the GPS failure scenario is detected.

4.3 SoC implementation

For our case study, Table 2 shows the evaluation of resources used for our implementation in terms of Bloc RAM (memory BRAM), Digital Signal Processors (DSP), Look Up Tables (LUT) and Flip Flop registers (FF). The total number of DSPs on the ZedBoard equals 220, that of LUTs is 53200 and that of FFs is 106400. The results are given for hardware solutions maximizing parallelism. The parallelism is better exploited in the case of GPS because there is less conditional dependency between nodes. The complete DM model uses the same number of DSPs as in case of GPS HM, which is explained by resource sharing.

Table 2 Resources used by the BN model for the programmable logic part.

	BRAM	DSP	LUT	FF
GPS localization HM	0%	34%	33 %	9%
Energy consumption HM	0%	29%	27%	7%
GPS HM+ Energy HM+ Decision	0%	34 %	51%	14%

Table 3 shows the performance for the HW/SW implementation. We observe a good HW speed-up in all cases because the SW implementation is sequential. A better speed-up is observed in the energy case, which is due to the computational complexity from the dynamic BN. The complete DM model has a good speed-up, which could be improved, but only at

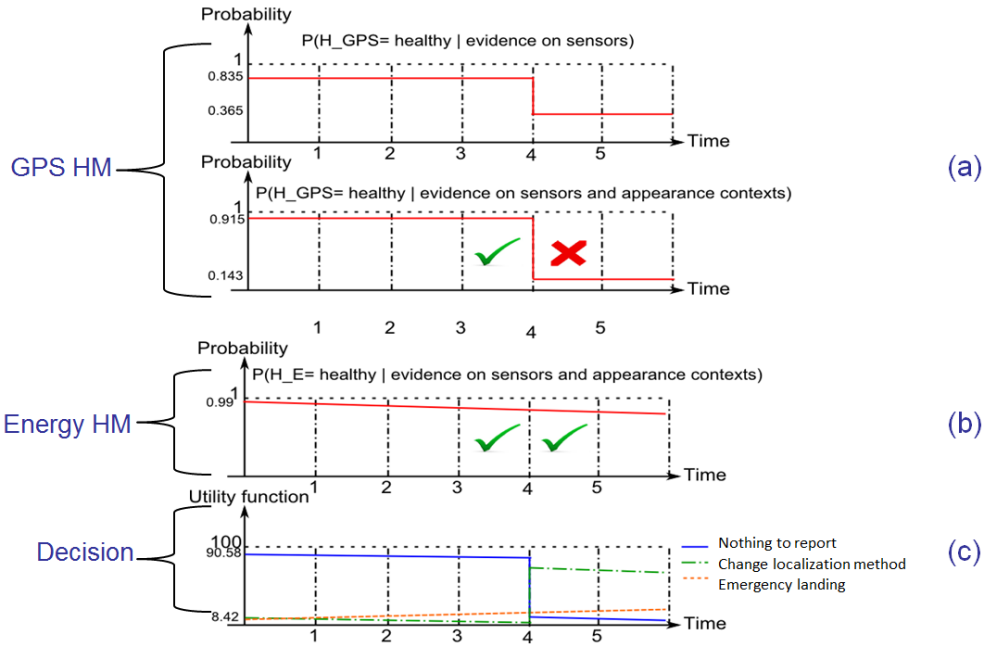

Figure 8 Results for the Decision Making of a simple UAV mission.

the expense of more resources. We also observe that although the communication time to send the network parameters is high, this can be eliminated or reduced if a local memory is used, which also means an increase in HW resources.

Table 3 The HW/SW performance of the BN nodes and DM, where HW Speed-up =SW execution time/HW execution time.

	SW time (cycles)	HW time (cycles)	HW Speed-up
GPS localization HM	955	391	2.42
Energy consumption HM	1268	339	3.74
GPS HM+ Energy HM+ Decision	2190	698	3.13

Table 4 Performance, Resource and Energy consumption of the Decision Making mechanism.

	Resource				Latency (cycles)	Speed-up	Energy cons.(μJ)
	BRAM	DSP	LUT	FF			
HM/DM in HW	14%	50%	34%	21%	419	6.28	9.81
HM in HW only	13%	44%	42%	20%	665	3.96	15.11

The results in Table 4 show the interest of a complete solution implemented in HW in terms of performance and energy consumption. Nevertheless with three actions, a good speed-up is already obtained for both versions of the DM, if HM are implemented in HW.

5 Summary and Conclusions

We present an original design tool for the implementation of Decision Networks on FPGA. This tool helps the designer to specify and implement the Decision-Making process under real-time constraints and energy constraints, making it suitable for embedded solutions on autonomous vehicles. We detail the design flow, giving the specific HLS optimizations and transformations available to generate implementation on an FPGA/SoC platform. We propose a validation example that demonstrates the interest of such a tool by providing efficient HW implementation for the Decision-Making engine.

Because of the complexity related to BN and DN definition and to their related classical inefficient implementation using junction trees [15], decision-making mechanisms based on DN were not really considered as a possible engine for embedded applications until now. Nevertheless, new compilation methods based on AC and on pattern identification for the BN description can help to achieve efficient implementation on both CPU and FPGA. These methods also help us to define a completely new embedded Decision-Network engine on the both supports. The present paper addresses this opportunity.

As perspectives for future work, we propose to integrate the runtime constraints of the sensors to fit the constraints of the mission in a more realistic manner. In this way, we could extend and couple the presented offline tool, which provides HW/SW implementation of mission planning, by adding an online engine that can choose the most appropriate version of the decision core implementation considering the CPU load, the system constraints in terms of energy and timing, and which can take into account service quality requirements. If the implementation of the decision process can be achieved on an FPGA/SoC support, the HW/SW alternatives can be chosen dynamically at runtime in the same way as this is possible for other applications, such as those for image processing [5].

References

1 Vivado-HLS. `https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html/`. Accessed: 2016-09-30.

2 Kai-Yuan Cai and Lei Zhang. Fuzzy reasoning as a control problem. *IEEE Transactions on Fuzzy Systems*, 16(3):600–614, 2008.

3 Junyi Chai, James NK Liu, and Eric WT Ngai. Application of decision-making techniques in supplier selection: A systematic review of literature. *Expert Systems with Applications*, 40(10):3872–3885, 2013.

4 Mark Chavira and Adnan Darwiche. Compiling Bayesian Networks with Local Structure. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 1306–1312. Professional Book Center, 2005. URL: `http://www.ijcai.org/papers/0931.pdf`.

5 Hanen Chenini, Dominique Heller, Catherine Dezan, Jean-Philippe Diguet, and Duncan Campbell. Embedded real-time localization of UAV based on an hybrid device. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1543–1547. IEEE, 2015.

6 Hsin-Han Chiang, Yen-Lin Chen, Bing-Fei Wu, and Tsu-Tian Lee. Embedded driver-assistance system using multiple sensors for safe overtaking maneuver. *IEEE Systems Journal*, 8(3):681–698, 2014.

7 Daniele Codetta-Raiteri and Luigi Portinale. Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 45(1):13–24, 2015.

8 Robert G Cowell. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business, 2006.

9 L.H. Crockett, R. Elliot, and M. Enderwitz. *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, 2014. URL: `http://books.google.fr/books?id=9dfvoAEACAAJ`.

10 Adnan Darwiche. A differential approach to inference in Bayesian networks. *J. ACM*, 50(3):280–305, 2003. `doi:10.1145/765568.765570`.

11 Adnan Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305, 2003.

12 Fabio Dovis, Bilal Muhammad, Ernestina Cianca, and Khurram Ali. A Run-Time Method Based on Observable Data for the Quality Assessment of GNSS Positioning Solutions. *Selected Areas in Communications, IEEE Journal on*, 33(11):2357–2365, 2015.

13 Vincent Drevelle and Philippe Bonnifait. Reliable positioning domain computation for urban navigation. *Intelligent Transportation Systems Magazine, IEEE*, 5(3):21–29, 2013.

14 Marek J Druzdzel. SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models. In *Aaai/Iaai*, pages 902–903, 1999.

15 Frank Jensen, Finn V Jensen, and Søren L Dittmer. From influence diagrams to junction trees. In *Uncertainty Proceedings 1994*, pages 367–373. Elsevier, 1994.

16 Z. Kulesza and W. Tylman. Implementation Of Bayesian Network In FPGA Circuit. In *Proceedings of the International Conference on Mixed Design of Integrated Circuits and System*, pages 711–715, June 2006. `doi:10.1109/MIXDES.2006.1706677`.

17 Richard B Langley. The GPS error budget. *GPS world*, 8(3):51–56, 1997.

18 Mingjie Lin, Ilia Lebedev, and John Wawrzynek. High-throughput Bayesian Computing Machine with Reconfigurable Hardware. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 73–82, New York, NY, USA, 2010. ACM. `doi:10.1145/1723112.1723127`.

19 Kevin Murphy et al. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

20 Kourosh Noori and Kouroush Jenab. Fuzzy reliability-based traction control model for intelligent transportation systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(1):229–234, 2013.

21 Judea Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

22 Luigi Portinale and Daniele Codetta-Raiteri. ARPHA: AN FDIR ARCHITECTURE FOR AUTONOMOUS SPACECRAFTS BASED ON DYNAMIC PROBABILISTIC GRAPHICAL MODELS. In *Proc. IJCAI workshop on AI on Space, Barcelona*, 2011.

23 Daniel Salós, Christophe Macabiau, Anaïs Martineau, Bernard Bonhoure, and Damien Kubrak. Nominal GNSS pseudorange measurement model for vehicular urban applications. In *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, pages 806–815. IEEE, 2010.

24 Johann Schumann, Timmy Mbaya, Ole Mengshoel, Knot Pipatsrisawat, Ashok Srivastava, Arthur Choi, and Adnan Darwiche. Software health management with Bayesian networks. *Innovations in Systems and Software Engineering*, 9(4):271–292, 2013.

25 Johann M Schumann, Kristin Y Rozier, Thomas Reinbacher, Ole J Mengshoel, Timmy Mbaya, and Corey Ippolito. Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems. *International Journal of Prognostics and Health Management*, 6, 2015.

26 Poorani Shivkumar. Intelligent controller for electric vehicle. In *2008 IEEE International Conference on Sustainable Energy Technologies*, pages 978–983. IEEE, 2008.