MARTE-based Design of a Multimedia Application and Formal Analysis

Adolf Abdallah, Abdoulaye Gamatié, Jean-Luc Dekeyser

▶ **To cite this version:**

Adolf Abdallah, Abdoulaye Gamatié, Jean-Luc Dekeyser. MARTE-based Design of a Multimedia Application and Formal Analysis. FDL 2008, Sep 2008, Stuttgart, Germany. pp.6. inria-00567972

HAL Id: inria-00567972

https://inria.hal.science/inria-00567972

Submitted on 23 Feb 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MARTE-based Design of a Multimedia Application and Formal Analysis

Adolf Abdallah, Abdoulaye Gamatié, Jean-Luc Dekeyser
USTL / CNRS / INRIA
Email: {adolf.abdallah, abdoulaye.gamatie, jean-luc.dekeyser}@lifl.fr

***Abstract*— Digital television (DTV) is an advanced broadcasting technology that is spreading fast today. It gives broadcasters the capability to send programs with a better picture and sound quality. Moreover, broadcasters can send several programming choices, called multicasting. DTV consists of a high-performance system combining both control and intensive data processing. In this paper, we first show how the OMG MARTE profile can serve to model such a system. Then, we use the synchronous approach to formally check some temporal properties of the expected system implementation for validation purpose.**

I. INTRODUCTION

A. Context and motivation

In distributed multimedia applications, the amount of data computation is exponentially increasing and is becoming more and more complex. One reason to that is the simultaneous streaming video which requires a large amount of computing resources.

To face this challenging design problem, numerous solutions have been proposed. *Model Driven Engineering* (MDE) is one of them. It has been initiated by the Object Management Group (OMG). In MDE, models are considered as primary engineering artifacts and can be applied to describe various aspects of a system, e.g. software, hardware.

The study presented in this paper deals with the modeling and analysis of high-performance systems according to the MARTE (Modeling and Analysis of Real-time and Embedded systems) profile [6]. We use the repetitive structure modeling to specify the data-intensive part and the TIME sub-profile to describe the temporal properties of system implementation. In order to formally validate the modeled design, these properties are analysed using the specific notion of affine clocks of the synchronous language SIGNAL [5]. The whole study is illustrated on a system example consisting of a digital television.

B. Case study: digital television (DTV)

1) Overview of the DTV *process:* DTV[1] is an advanced broadcasting technology where TV signals are delivered in a much more efficient way than the current analog TV. Clearer picture and sound are one of the advantages of DTV. Other enhancements are the wide screen picture and the multiple channel emission (multicasting) or parallel streaming. An easy way to visualise the great advantage of the DTV is to consider a live sporting event such as a soccer tournament. Some of the soccer fans watching the game on TV prefer to walk around the field with their favourite player. Others would like to watch the game from a global view. This kind of interaction is possible in DTV through the parallel streaming technology. DTV allows a variety of screen formats: 640*480 pixels, 1080*720 pixels and 1920*1080 pixels.

Fig. 1 describes the process from the beginning, by taking several pictures per second, to the end by displaying these pictures on the consumer's screen. The process begins with a DTV camera in the field taking approximately 60 pictures per second. Here, four data streams corresponding to four different channels, are processed in parallel. These multiple data streams are then combined into one signal on a shared medium using a multiplexer.

2) Downscaling and compression: A high definition (HD) image emission constitutes a flow of 75 Mbytes per second. The DTV bandwidth is 19.39 Mbytes per second, which is five times greater that the bandwidth of standard TV signals.

A two phase treatment is done on the video image sequence so that the four channels can fit the bandwidth all together. Fig. 1 depicts the different computations that are applied to the HD images. First, a typical downscaler converts the HD video signal (1920,1080) to a standard definition (SD) video signal (640,480). The downscaling process contains two treatments. The first is an horizontal filter, HF, which reduces the number of pixels in a line from 1920 down to 640 by interpolating packets of six pixels. The second is a vertical filter, VF, which reduces the number of lines in a frame from 1080 down to 480 by interpolation packets of six pixels.

Once the video is downscaled, the data streaming must be compressed so that all four channels can fit the bandwidth. Therefore, DTV uses MPEG-2 compression to conserve as much bandwidth as possible.

3) MPEG-*2 compression:* In MPEG-2, a bit rate reduction system operates by removing redundant information from the signal at the coder prior to transmission.

It then inserts the lost information through the decoder. Nevertheless, MPEG-2 uses the transport stream protocol on the bytes received from the downscaler. This protocol was designed to carry digital video over medias that are possibly lossy (e.g. broadcasting). The basic unit of this protocol is a 188 bytes packet. The MPEG coder receives a flow of 160 bytes payload from the downscaler and adds to it a sync byte, three one-bit flags and a 13-bit PID.

Using this technique, it is possible to transfer several video channels simultaneously over the same frequency channel.

[1] `http://electronics.howstuffworks.com/dtv.htm`.

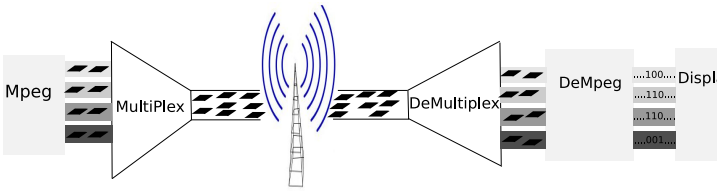


Fig. 1. An overview of the DTV system.

Once the consumer receives the numerous packets through the shared medium, a reverse process is applied on these packets. Using a De-multiplexer, the original channels are extracted. These channels, represented as a flow of 188 bit packets, are afterwards decompressed using the MPEG-2 reverse algorithm.

In the next sections, we use the GASPARD design environment [8], which is devoted to the development of high-performance embedded systems based on the MDE approach. Several transformation chains provided in the environment for code generation towards different target technologies: synchronous languages for formal validation, SystemC for simulation, VHDL for circuit synthesis, etc.

The remainder of the paper is structured as follows: in Sections II and III, we respectively address the modeling of functional and implementation-related aspects of the DTV example. We mainly use the MARTE repetitive and time structures. In Section IV, we introduce the SIGNAL language, which is used to analyze non functional properties of the DTV model. Finally, we give concluding remarks in Section V.

II. MODELING OF FUNCTIONAL ASPECTS

A. Repetitive structure modeling in MARTE

The *repetitive structure modeling* (RSM) package of MARTE [6] proposes concepts that enable to describe both data parallelism and task parallelism in embedded systems. It originally relies on the domain-specific language ARRAY-OL dedicated to intensive multidimensional signal processing [3]. Fig. 2 depicts the basic stereotypes associated with the RSM package. Due to lack of space, we only detail the stereotypes that are necessary to understand the model of DTV functionality presented in Section II-B.

According to the execution model of ARRAY-OL, a data-parallel task T is repeated or replicated into several task instances $\{T_{i,i\in 1..k}\}$ that take as inputs subsets of data extracted from the inputs of T, which are multidimensional arrays. These subsets of array elements are referred to as *patterns* or *tiles*. For a given task repetition, the number k of task instances T_i is given by the *repetition space* associated with the *repetitive* task T.

The Tiler stereotype, described in Fig. 2, expresses how multidimensional arrays are tiled by patterns. When applied to a delegation connector, a Tiler connects an external port with a port of an internal part. The shape of the external ports defines the shape of input/output arrays of a task. The port shape of the internal part defines the pattern shape and the shape of the part itself defines the repetition space.

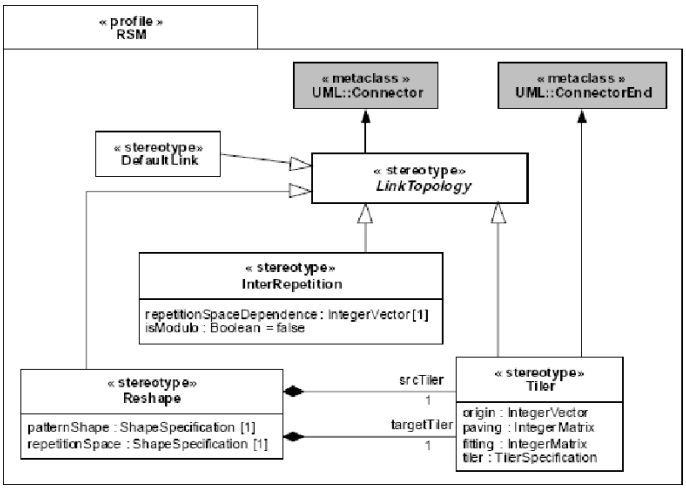


Fig. 2. RSM package specification in MARTE.

A tiler uses three main information (see the attributes of the corresponding stereotype) to define the tiling operation:

- origin vector, which specifies the origin of the reference tile in the array;
- fitting matrix, which specifies how the patterns are filled with array elements;
- paving matrix, which specifies how an array is covered by pattern elements.

The Reshape stereotype extends the UML2 Connector metaclass. It enables to represent complex link topologies in which the elements of a multidimensional array are redistributed in another array. The major difference between the Tiler and Reshape stereotypes is that the former is used for delegation connector (i.e. between the port of a component and the ports of its parts) and the latter is used for connector that link two parts. The InterRepetition stereotype is used to specify a data dependency between repetitive task instances. For a full description of all these stereotypes, the reader may refer to the specification document of MARTE.

B. Modeling of the DTV functionality

A pixel represents the intensity of the three basic colours red, green and blue. The number of colors that can be represented depends on the number of bits that constitute the pixel. For instance, common values are 8 bit per pixel, 16 bpp, 24 bpp and 48 bpp. The latter is used in high quality scanners. In our modeling, we consider that each pixel is composed of 8 bits that can represent 256 colours.

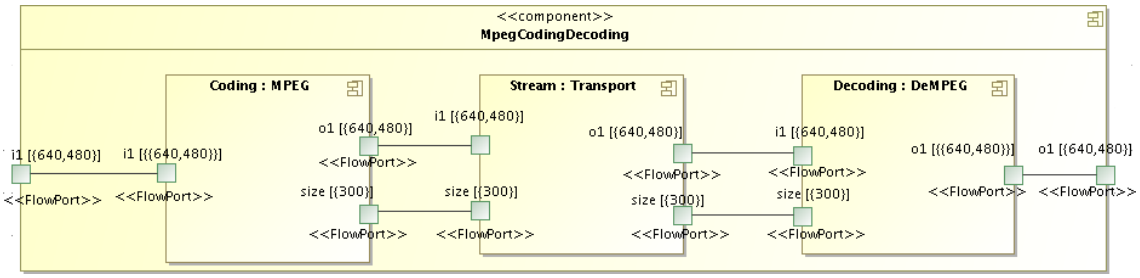


Fig. 3. A zoom on the MPEGCodingDecoding component.

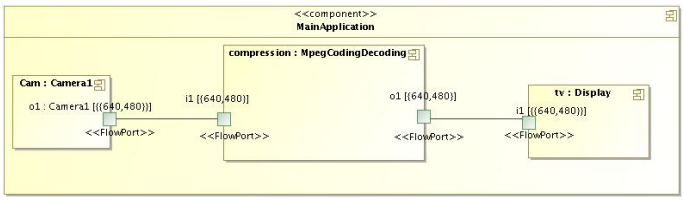


Fig. 4. A partial view of the DTV system model.

Fig. 4 depicts a partial view of the DTV system model. Only one camera, called Cam, is shown here for the sake of clarity. Note that since the DTV system aims at providing consumers with multiple channel emission (multicasting), there should be more than one image source. Actually, DTV comprises four synchronized cameras producing image flows that can be potentially displayed on a screen simultaneously according to consumer's demand.

To illustrate how the intensive data processing part is modeled with the MARTE concepts introduced previously in Section II-A, we focus on the MPEG coder specification.

Fig. 3 describes the three processes that compose the MPEGCodingDecoding component. Images are first compressed using the MPEG-2 algorithm and then are sent to the consumer through a transport stream. Once the consumer receives the compressed data, he performs a decompression on them using the MPEG-2 reverse algorithm.

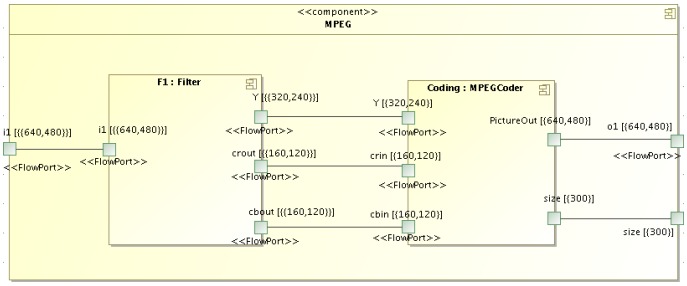


Fig. 5. Model of the MPEG coder.

A zoom on the MPEG component is shown in Fig. 5. It is composed of a filter and an MPEG coder. The filter extracts the three basic colours used in video and digital photography. Y is the luma component and Cb and Cr are the blue and red chroma components. The MPEG coder takes the filtered image and compresses it so that it can be sent to the receiver through a shared medium.

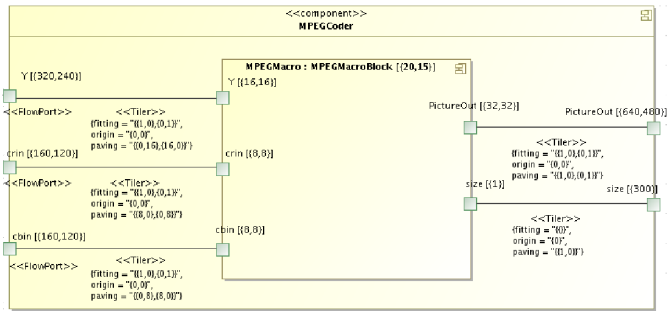


Fig. 6. Data-parallelism in the MPEG coder.

Fig. 6 depicts our specification of the MPEG coder itself using a repetitive task MPEGCoder expressing data parallelism. The execution of MPEGCoder is such that its part, represented by MPEGMacro, is replicated into 20*15 instances. The multiplicity notation $[\{20, 15\}]$ denotes the repetition space of the MPEGCoder task.

The patterns consumed by the MPEGMacro part consist of two-dimensional arrays: a 16*16 pixels array for luminance, and two 8*8 pixels arrays for the red and blue chrominances. The resulting outputs are a two-dimensional array mbout and a mono-dimensional array size. The outputs mbout and size are respectively the resulting compressed image and the size of the image. MPEGMacro consumes patterns from input arrays via tilers. Each tiler contains an origin vector, a fitting matrix and a paving matrix. E.g., the luminance Y is an input array of size (320,240). The MPEGMacro consumes patterns composed of 16*16 pixel arrays. The paving matrix, valued $((0, 16), (16, 0))$, enables to determine the origin of each pattern associated with a repeated instance of MPEGMacro. The fitting matrix, valued $((1, 0), (0, 1))$, enables to fill compact rectangular patterns with array elements.

The downscaling process is done automatically through an option in the digital camera video. Here, the pictures received by the broadcaster are assumed to have been already downscaled. For more details on the specification of the downscaling process using the repetitive structure modeling, the reader can refer to [4].

III. Modeling of temporal aspects

A. Time specification in MARTE

Time is present in different design activities of an embedded system: specification, design, validation, etc. The sub profile TIME of MARTE has been introduced to model timing aspects during system design [1]. In MARTE, *logical clocks* as well as *physical clocks* can be defined. The main difference between physical and logical clocks is that the former has a reference to physical time while the latter has a reference to events. The occurrences of these events define the clock instants, also referred to as *ticks*. Fig. 7 depicts the MARTE extension about time modeling in UML.

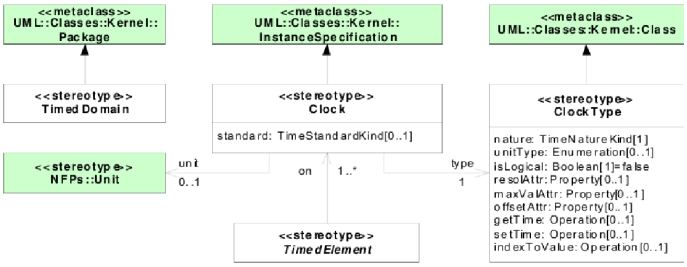


Fig. 7. TIME Sub Profile specification.

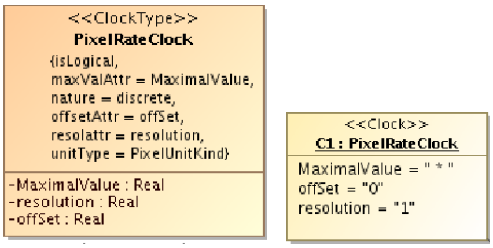
The clock stereotype is a model element that represents an instance of ClockType and extends the InstanceSpecification of UML. It can be only applied to instance specifications of a class stereotyped by ClockType. A clock is characterised by its resolution, and optionally by an offset and a maximalValue. The unit type of a clock by default is tick.

The ClockType stereotype is related indirectly to the Clock stereotype. A ClockType is a classifier of clock which means that the former defines the characteristics of the represented time in the latter through its attributes. The attribute Standard of the clock stereotype is a reference to time system adopted by the clock. It is not defined in logical clocks. The attribute nature of the ClockType stereotype specifies whether the time is discrete or dense. UnitType is the different type of units supported by the ClockType. The Boolean attribute isLogical specifies whether the ClockType stereotype reads a logical time or a physical time. When it is true, the clock reads logical time. Otherwise, it reads a chronometric time. The maxValAttr attribute refers to the maximal value of the associated clock, value at which the clock rolls over. Finally, the resolAttr attribute determines the resolution of the associated clock.

B. Modeling of DTV implementation properties

1) Clock specification using MARTE *profile:* In this section, we define for the DTV application logical clocks as well as their associated properties in a high level specification using the TIME sub-profile of MARTE. First, we show how to create logical clocks and then how to make use of them.

Fig. 8(a) depicts the definition of a logical clock. We have created a class PixelRateClock with ClockType as stereotype. This class has three attributes: MaximalValue, offSet and resolution. MaximalValue specifies the maximal value of the



(a) Definition (b) Instantiation

Fig. 8. Creation of logical clocks.

associated clock, value at which the clock rolls over. The offSet property determines the initial instant of the associated clock. The resolution attribute defines the resolution of the associated clock.

Now that we have defined the clock, we need to instantiate it. Fig. 8(b) depicts the instantiation of the clock PixelRateClock called C1. In the same way, we have defined two other clocks, MpegPacketClock and DisplayClock, and instantiated them with C2 and C3 (see Fig. 9).

Each of the different applications composing the DTV system works according to different rhythms. For example, the HD camera (representing the sensor in Fig.1) emits a flow of pixels continuously in time and over a certain frequency. Therefore, we attach to the sensor the logical clock C1 defined previously. The unit of C1 is the pixel production.

Once the broadcaster receives the image flow, these images are compressed using the MPEG-2 algorithm. Coding and decoding works on the same rhythm and thus are considered as synchronous. The MPEG coding algorithm, the transport stream and the decoding algorithm have the same clock MpegPacketClock. Therefore, we assign the clock C2 to the MpegCodingDecoding component.

On the other hand, a buffer is inserted between the MPEG decoder and the display process so that it gathers the different parts of the picture. Thus, we attach to the Display process the logical clock C3 which is an instance of the DisplayClock clock. Fig. 9 specifies the clock assignment for the whole system. To assign a clock instance to a process, we create a class with the process as its classifier. Then, we stereotype this class with TimedProcessing and we specify the corresponding clock using the meta-attribute on.

2) Constraint specification using MARTE *profile:* In the previous section, we have instantiated the PixelProductionRate and called it C1. The clock C1 has three attributes, resolution with the value 1 which means that its rhythm is the same as the Base clock, an offset with the value 0 and a maximalValue which is infinite. In the same way, the clocks C2 and C3 are assigned to the MpegCodingDeconding and Display processes.

The MPEG coding treats blocks of $16 * 16$ pixels. This is equivalent to say that the MpegCodingDeconding component is activated every $16 * 16^{th}$ tick of the clock associated with camera1 component. We can deduce an affine relation between C1 and C2 as specified in the clock constraint

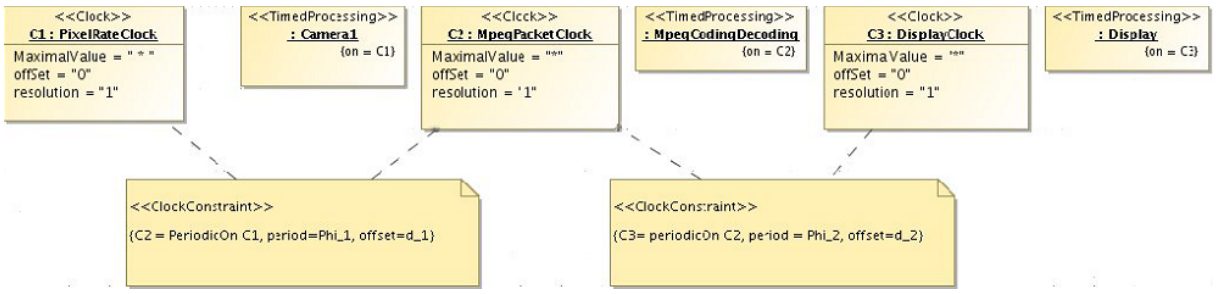


Fig. 9. Specification of clock properties in DTV.

specified in Fig. 9: C2 = PeriodicOn C1, period = phi_1, offset=d_1. Such a constraint states that each phi_1th occurrence of C1 there will be an occurrence of C2 and the first occurrence of C2 appears at the d_1th instant of C1. Here, one may consider that phi_1 = 16*16 if the clock C1 corresponds to the pixel production rate.

In addition, during the execution of the DTV system, an image is displayed only when the buffer inserted between the MPEG decoder and the display contains all pixels that form complete images. This also induces another affine relation between C2 and C3, which is also specified in Fig. 9.

IV. Analysis of design properties

Now, we address the formal analysis of the temporal properties specified in the previous section. We consider an encoding of the identified the clock constraints using synchronous languages, which offer very suitable tools to deal with logical clock properties in system design. In the next, we first introduce the SIGNAL language. Then, we use its concepts to reason on clock constraints.

A. The synchronous language SIGNAL

1) Overview of basic concepts: The synchronous language SIGNAL [5] belongs to the family of data-flow languages. It handles unbounded series of typed values $(x_\tau)_{\tau\in\mathbb{N}}$, called *signals*, denoted as `x` and implicitly indexed by discrete time. At a given instant, a signal may be present, at which point it holds a value; or absent and denoted $\perp$. The set of instants where a signal `x` is present represents its *clock*, noted `^x`. Two signals `x` and `y`, which have the same clock are said to be *synchronous*, noted `x ^= y`. A *process* (or *node*) is a system of equations over signals that specifies relations between values and clocks of the signals. A *program* is a process. SIGNAL relies on the following six primitive constructs:

Relations. `y:= f(x1,...,xn)` $\stackrel{def}{\equiv} \forall\tau \geq 0 : y_\tau \neq\perp\Leftrightarrow x1_\tau \neq\perp\Leftrightarrow ... \Leftrightarrow xn_\tau \neq\perp$, and $y_\tau = f(x1_\tau, ..., xn_\tau)$.

Delay. `y:= x $ 1 init c` $\stackrel{def}{\equiv} \forall\tau\ x_\tau \neq\perp\Leftrightarrow y_\tau \neq\perp$ *and* $\forall\tau > 0$: $y_\tau = x_k$, *where* $k = max\{\tau' \mid \tau' < \tau \text{ and } x_{\tau'} \neq\perp\}, y_0 = c$.

Undersampling. `y:= x when b` where `b` is Boolean $\stackrel{def}{\equiv} \forall\tau \geq 0 : y_\tau = x_\tau$ if $b_\tau = true$, else $y_\tau = \perp$. The expression `y:= when b` is equivalent to `y:= b when b`.

Deterministic merging. `z:= x default y` $\stackrel{def}{\equiv} \forall\tau \geq 0 : z_\tau = x_\tau$ if $x_\tau \neq\perp$, else $z_\tau = y_\tau$.

Composition. `(| P | Q |)` $\stackrel{def}{\equiv}$ union of the equations of `P` and `Q`. This operator is commutative and associative.

Hiding. `P where x` $\stackrel{def}{\equiv}$ `x` is local to the process `P`.

SIGNAL offers a process frame that enables the definition of *sub-processes* (see the `where` part in Fig. 11).

2) Affine clocks: In order to address clock synchronization issues in the DTV application, we consider the *affine* clock notion of SIGNAL. Affine clocks have been defined as an extension of the usual clock notion of SIGNAL so as to enable the modeling and validation of real time systems with intensive data processing [7].

Definition 1: An *affine transformation* of parameters (n, ϕ, d) applied to a clock c_1 produces a clock c_2 by inserting $(n-1)$ instants between any two successive instants of c_1, and then counting on this fictional set of instants each d^{th} instant, starting with the ϕ^{th}. Clocks c_1 and c_2 are said to be in (n, ϕ, d)-affine relation, noted as $c_1 \stackrel{(n,\phi,d)}{\rightarrow} c_2$ (see example in Fig. 10).

	0	1	2	3	4	5	6	7	8	9	10	...
c_1:	tt	$\perp$	$\perp$	tt	$\perp$	$\perp$	tt	$\perp$	$\perp$	tt	$\perp$	...
c_2:	$\perp$	tt	$\perp$	$\perp$	$\perp$	tt	$\perp$	$\perp$	$\perp$	tt	$\perp$	...

Fig. 10. Trace where c_1 and c_2 are in $(3, 1, 4)$-affine relation.

In affine clock systems, two different signals are said to be synchronizable if there is a dataflow preserving way to make them actually synchronous. The SIGNAL *clock calculus* (i.e. its static analysis phase) has been extended in order to address such synchronizability issues with the associated compiler.

B. Dealing with synchronisation issues in DTV

As described in Fig. 9, there are three identified logical clocks in the system: C1, C2 and C3, which are respectively associated with the Camera, MpegCodingDeconding and Display components. The relations between these clocks are affine. Therefore, we can use SIGNAL affine clocks to encode these clock relations in the DTV as follows:

$$(P_1)\text{: } c_1 \stackrel{(1,\phi_1,d_1)}{\rightarrow} c_2 \text{ and } (P_2)\text{: } c_2 \stackrel{(1,\phi_2,d_2)}{\rightarrow} c_3;$$

where c_1, c_2 and c_3 represent respectively the affine clocks corresponding to C1, C2 and C3. The properties (P_1) and (P_2) expresses the pairwise relations between these clocks.

According to [7], various operations can be defined on affine relations. Among them, we can mention the *composition*

operation, noted $*$. Typically, the above two affine relations could be composed as follows: $c_1 \overset{(1,\phi_1,d_1)}{\rightarrow} c_2 * c_2 \overset{(1,\phi_2,d_2)}{\rightarrow} c_3$.

The result of this composition yields a direct affine relation relation between c_1 and c_3. It has been demonstrated that it is equivalent to: $c_1 \overset{(1,\phi_1+d_1\phi_2,d_1d_2)}{\rightarrow} c_3$.

The above clock relations formally characterize a logical temporal behavior of our DTV system model. This description can be used to reason now about some system requirements as illustrated below. For instance, if a television broadcaster wants to address the quality of service (QoS) of the images sent to the consumers, he may need some particular image production rates that will provide consumers with a very high quality image display. In this context, let us consider a QoS criteria that requires an image display frequency such that the following constraint between the above pixel production rate c_1 and a new image display rate c_4 are satisfied:

$$(P_3)\text{: } c_1 \overset{(1,\phi_3,d_3)}{\rightarrow} c_4.$$

We can use affine clocks to check the compatibility of the QoS constraint P_3 with the design constraints P_1 and P_2. By compatibility, we mean that the respective clocks c_3 and c_4, associated with image display rate in design and QoS, are synchronizable. The following equivalences allows one to straightforwardly check the synchronizability of these clocks: $\phi_1 + d_1\phi_2 = \phi_3$ and $d_1d_2 = d_3$.

Fig. 11 shows a SIGNAL program that describes the above clock relations in the DTV system: P_1, P_2 and P_3. This program called `DTV`, takes as input a clock `c1` representing the pixel production rate (line **3**), and generates the other clocks mentioned before: `c2`, `c3` and `c4` (line **4**). The parameters of the identified affine relations are specified here as static parameters of the SIGNAL program (line **2**).

```
1  process DTV =
2   { integer d1, phi1, d2, phi2, d3, phi3;}
3   (? integer c1;
4    ! integer c2,c3,c4; )
5   (| c2 := affine_sample{phi1,d1}(c1)
6    | c3 := affine_sample{phi2,d2}(c2)
7    | c4 := affine_sample{phi3,d3}(c1)
8    |)
9   where
10   process affine_sample =
11    { integer phi, d; }
12    ( ? x;
13      ! y; )
14    (| v ^= x
15     | v:= (d-1) when(zv=0) default (zv-1)
16     | zv := v$ init phi
17     | y := x when (zv=0)
18     |)
19   where
20     integer v, zv;
21  end;%process affine_sample%
22 end;%process DTV
```

Fig. 11. Specification of affine clock relations in SIGNAL.

The affine relation is encoded by the `affine_sample` sub-process defined from line **10**. For more details on this specific process, the reader can refer to [2].

In order to address the previous synchronizability constraint with the SIGNAL compiler depending on particular parameter values, we have to explicitly specify the constraint in the `DTV` program. The equations of Fig. 12, which encode the constraint, are therefore inserted in the program after line **7**.

```
1  (| C_c4 := ^c4 default not(c4 ^+ c3)
2   | C_c3 := ^c3 default not(c4 ^+ c3)
3   | assert(C_c4=C_c3) when(^C_c4)
4      when(phi3=phi1+d1*phi2) when(d3=d1*d2)
5   |)
```

Fig. 12. Specification of a clock synchronizability constraint.

Using the resulting modified program with the SIGNAL compiler, one is able to formally check the compatibility of the design properties and the QoS requirement according to particular values of affine relation parameters.

V. CONCLUSIONS

In this paper, we showed how the concepts of the MARTE standard profile can serve to model high-performance multimedia applications. We used the repetitive structure modeling to describe the data intensive part, and the TIME sub-profile to specify temporal properties of these applications w.r.t. particular implementation choices. This has been illustrated on a simplified case study consisting of a digital television. On the other hand, we showed how synchronous languages, such as SIGNAL, are used to encode and analyze some temporal properties of the resulting model for formal validation purpose.

Beyond the specific topic of this paper, the richness of the MARTE profile in terms of concepts offers an interesting common modeling basis to adequately specify many design features of embedded systems, from both software and hardware viewpoints. As a complement, powerful techniques and tools already exist, which enable to analyze the properties of obtained models. All this favors development frameworks where design exploration is made efficient. This is the major goal of environments such as GASPARD, which already adopts the MARTE profile, and integrates different target technologies for formal verification, system simulation and execution, and circuitry synthesis.

REFERENCES

[1] Charles André, Frédéric Mallet, and Robert de Simone. Modeling time(s). In *Proceedings of Model Driven Engineering Languages and Systems (MoDELS'07), Nashville, USA*, volume 4735 of *Lecture Notes in Computer Science*, pages 559–573. Springer, 2007.

[2] L. Besnard, T. Gautier, and P. Le Guernic. SIGNAL reference manual., 2007. `www.irisa.fr/espresso/Polychrony`.

[3] P. Boulet. Formal Semantics of ARRAY-OL, a Domain Specific Language for Intensive Multidimensional Signal Processing. Technical report, INRIA, France, March 2008. available online at `http://hal.inria.fr/inria-00261178/fr`.

[4] Pierre Boulet. Array-OL revisited, multidimensional intensive signal processing specification. Research Report RR-6113, INRIA, February 2007. `http://hal.inria.fr/inria-00128840/en`.

[5] P. Le Guernic, J.-P. Talpin, and J.-C. Le Lann. Polychrony for System Design. *Journal for Circuits, Systems and Computers*, 12(3):261–304, April 2003.

[6] Object Management Group. A uml profile for MARTE 2007. `http://www.omgmarte.org`.

[7] I.M. Smarandache, T. Gautier, and P. Le Guernic. Validation of mixed SIGNAL-ALPHA real-time systems through affine calculus on clock synchronisation constraints. In *World Congress on Formal Methods (2)*, pages 1364–1383, 1999.

[8] The DaRT team. GASPARD design environment, 2008. `https://gforge.inria.fr/projects/gaspard2`.