Applying Timed-Arc Petri Nets to improve the performance of the MPEG-2 Encoding Algorithm[*]

Fernando L. Pelayo Fernando Cuartero Valentín Valero Hermenegilda Macia Maria L. Pelayo
Escuela Politécnica Superior de Albacete
University of Castilla-La Mancha
Campus Universitario s/n. 02071 Albacete, Spain
{fpelayo, fernando, valentin, hmacia, mpelayo}@info-ab.uclm.es

Abstract

We use a timed extension of Petri nets, the so called Timed-Arc Petri Nets, for the specification and analysis of the MPEG–2 Video Encoder. We have computed bounds for the necessary time to encode each type of frame, also we present an improvement on the encoding process which takes advantage of the potential parallelism degree of the MPEG–2 video encoding algorithm, so reaching a 90% of reduction on the time requirements, with respect to the original MPEG–2 encoder.

Keywords: Formal Methods, Timed-Arc Petri Net, Performance Evaluation, MPEG–2 Video Encoder.

1 Introduction

Formal Models are used to describe and to analyze the behaviour of computer systems. Among these models, we have Process Algebras, Event Structures, Markov Chains, Petri Nets and some others. Software designers work gladly with process algebras, since they have a syntax very similar to a programming language, but they are not able, in general, to capture real concurrency, and even formal verification is a bit harder than in other formalisms like Petri Nets. A survey of the different approaches to introduce time in Petri nets is presented in [3]. We can identify a first group of models, which assign time delays to transitions, either using a fixed and deterministic value [11, 12] or choosing it from a probability distribution.Finally, we have also some models that introduce time on tokens [6, 13]. In such a case tokens become classified into two different classes: available and unavailable ones. Available tokens are those that can be immediately used for firing a transition, while unavailable cannot. We have to wait for a certain period of time for these tokens to become available, although it is also possible for a token to remain unavailable forever (such tokens are said to be *dead*). More recently, Cerone and Maggiolo-Schettini [4] have defined a very general model (statically timed Petri nets), where timing constraints are intervals statically associated with places, transitions and arcs. Thus, models with timing constraints attached only to places, transitions or arcs can be obtained by considering particular subclasses of this general framework.

Timed-Arc Petri nets [1, 5, 7, 13] are a timed extension of Petri nets in which tokens have associated a non-negative real value indicating the elapsed time from its creation (*its age*), and arcs from places to transitions are also labelled by time intervals, which establish restrictions on the age of the tokens that can be used to fire the adjacent transitions. As a consequence of these restrictions some tokens may become *dead*, because they will be never available, since they are too old to fire any transitions in the future. The interpretation and use of Timed-Arcs Petri nets can be obtained from a collection of processes interacting with one another according to a rendez-vous mechanism. Each process may execute either local actions or synchronization ones. Local actions are those that the process may execute without cooperation from another process, and thus in the Petri net model of the whole system they would appear as transitions with a single precondition place, while synchronization actions would have several precondition places, which correspond to the states at which each involved process is ready to execute the action. Then, each time interval establishes some timing restrictions related to a particular process (for instance the time that a local processing may require). In consequence, the firing of a synchronization action can be done in a time window, which depends on the age of the tokens on its precondition places.

Therefore, Timed-Arc Petri nets are a very appropriate model for the description of concurrent systems with time restrictions, such as manufacturing systems, real-time systems, process control, workflow systems, etc. In this paper we use this concrete model of specification to analyze the concurrent behaviour of the MPEG-2 video encoder, taking into account the time required to complete each task in the encoding process. From this analysis we will conclude that

[*]This work has been partially supported by the CICYT project "Performance Evaluation of Distributed Systems", TIC2000-0701-c02-02

the performance of this process can be improved by exploiting the intrinsic parallelism in it.

The MPEG standards were designed with these two requirements:

- The need for a high compression, which is achieved by exploiting both spatial and temporal redundancies within an image sequence.
- The need for random access capability, which is obtained by considering a special kind of pictures (I pictures), which are encoded with no reference to other frames, only exploiting the spatial correlation in a frame.

In the literature we can found several publications on performance improving of MPEG standards. Most of these works focus their improvements on parallelizing the distribution of data among the processors by either distributing different partitions of the same frame (spatial parallelism) or different GoPs (Groups of Pictures/Frames) to the various processors (temporal parallelism) [2, 9], but it is usual to consider the codification of each frame as a sequential process.

On the other hand, Formal Methods are able to analyze the potential improvement obtained when parallelizing algorithms. In [10] and [14] the MPEG–2 video encoder was modelled and analyzed by the authors by means of the Markovian process algebra **ROSA** and and by timed-arc Petri nets, respectively. There, we presented the performance improvements on this algorithm when having two processors.

In this paper we present an improvement for the encoder by considering the parallelization within the encoding process of each type of image, so we have modelled this encoding algorithm with TAPNs, and we have concluded from the analysis of the model that a better performance could be obtained in the encoding process by introducing some minor changes on the encoder. Specifically, we have computed some bounds for the time required to encode each type of image of a video sequence, and we have concluded that some improvements can be introduced in the encoding process of the B-images mainly, and that a full parallel version of this algorithm could reduce the time requirements in a magnitude order (reduction of about a 90%).

The paper is structured as follows. In Section 2 we present the Timed-Arc Petri Nets and their semantics. In Section 3 we describe how the MPEG–2 encoder works, and we show in Section 4 a MTAPN that models this algorithm and its analysis. The concluding part of the paper is presented in Section 5.

2 Timed-Arc Petri Nets

We deal with timed-arc Petri nets, which have their tokens annotated with an age (a real value indicating the elapsed time from its creation) and arcs connecting places with transitions have associated a time interval, which limits the age of the tokens that must be consumed to fire the adjacent transition.

However, a transition is not forced to be fired when all its preconditions contain tokens with an adequate age, and the same is true even if the age of any of these tokens is about to expire. More in general, in the model we consider[1] there is not any kind of urgency, what we can interpret in the sense that the model is *reactive*, as transitions will be only fired when the environment requires it. But then, it can be the case that the external context may lose the ability to fire a transition if some needed tokens become too old. Even more, it is possible that some tokens become *dead*, which means definitely useless because their increasing age will not allow in the future the firing of any of their postcondition transitions.

Definition 1 *(Timed-arc Petri nets)*
We define a timed-arc Petri net (TAPN) as a tuple[2] $N = (P, T, F, times)$*, where* P *is a finite set of* places*,* T *is a finite set of transitions (*$P \cap T = \emptyset$*),* F *is the* flow relation *(*$F \subseteq (P \times T) \cup (T \times P)$*), and times is a function that associates a closed time interval to each arc* (p, t) *in* F*, i.e.*

$$times : F|_{P\times T} \longrightarrow \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$$

When $times(p, t) = [x_1, x_2]$ *we write* $\pi_i(p, t)$ *to denote* x_i*, for* $i = 1, 2$*.*

As usual for any $x \in P \cup T$*, we define the set of preconditions of* x *by* ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$*, and analogously, we define the set of postconditions of* x *by* $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$*.*

As we previously mentioned, tokens are annotated with real values, so markings are defined by means of multisets on $\mathbb{R}_0^+$*. More exactly, a marking* M *is a function:* $M : P \longrightarrow \mathcal{B}(\mathbb{R}_0^+)$ *where* $\mathcal{B}(\mathbb{R}_0^+)$ *denotes the set of finite multisets of non-negative real numbers. As usual, each place is annotated with a certain number of tokens, but each one of them has associated a non-negative real number (its age). We will denote the set of markings of* N *by* $\mathcal{M}(N)$*, and using classical set notation, we will denote the number of tokens on a place* p *by* $|M(p)|$*.*

As initial markings we only allow markings M *such that for all* $p \in P$*, and any* $x > 0$ *we have* $M(p)(x) = 0$ *(i.e., the* initial age *of any token is 0). Then, we define* marked timed-arc Petri nets *(MTAPN) as pairs* (N, M)*, where* N *is a timed-arc Petri net, and* M *is an initial marking on it. As usual, from this initial marking we will obtain new markings, as the net evolves, either by firing transitions, or by time elapsing. In consequence, given a non-zero marking, even if we do not fire any transitions at all, starting from this*

[1]Other proposals of Timed-arc Petri nets [7] enforce the firing of transitions with an earliest and maximal firing rule.

[2]We consider only arcs with weight 1 to simplify some definitions, but the extension to general arcs with greater weights is straightforward.

marking we get an infinite reachability set of markings, due to the token aging. These reachable markings will reflect the state of the net at each instant of its evolution.

A timed-arc Petri net with an arbitrary marking can be graphically represented by extending the usual representation of P/T nets with the corresponding time information. In particular we will use the age of each token to represent it. Therefore, MTAPNs have initially a finite collection of zero values labelling each marked place.

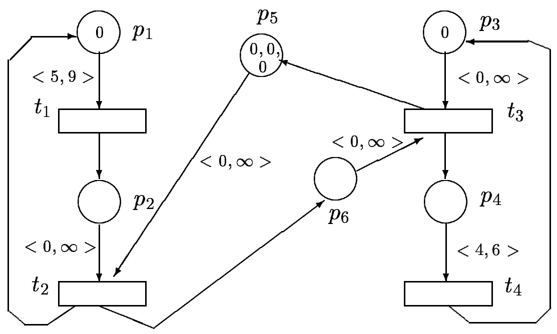

Figure 1. Timed-arc Petri net modelling the Producer-Consumer problem

Fig. 1 shows a MTAPN modelling a producer/consumer system, where we represent by transition t_1 the action corresponding to the manufacturing process of the producer, which takes between 5 and 9 units of time, and by t_2 the action of including the generated object into the buffer. Notice that the initial tokens on p_5 represent the capacity of the buffer (3), and the arc connecting this place with t_2 is labelled by the interval $< 0, \infty >$, because these tokens can be consumed at any instant in the future. Tokens on p_6 represent the objects on the buffer which have not been yet consumed. Transition t_3 models the action of taking out an object from the buffer, which can occur at any instant. Finally, transition t_4 models the processing that makes the consumer for the objects extracted from the buffer, and this action takes between 4 and 6 units of time.

Let us observe that if the enabling time for the firing of one of these transitions (t_1 or t_4) expires, the system eventually becomes deadlocked, because we obtain a *dead* token either on p_1 or on p_4.

Let us now see how we can fire transitions, and how we model the time elapsing.

Definition 2 *(Firing rule)*
Let $N = (P, T, F, times)$ be a TAPN, M a marking on it, and $t \in T$.

(i) We say that t is enabled *at the marking M if and only if: $\forall p \in {}^\bullet t \; \exists x_p \in \mathbb{R}_0^+$ such that $M(p)(x_p) > 0 \wedge x_p \in times(p, t)$, i.e., on each precondition of t we have some token whose age belongs to $times(p, t)$.*

(ii) If t is enabled at M, it can be fired, and by its firing we reach a marking M', defined as follows:

$$M'(p) = M(p) - C^-(p, t) + C^+(t, p), \; \forall p \in P$$

where both the subtraction and the addition operators work on multisets, and:

- $C^-(p,t) = \begin{cases} \{x_p\} & \textit{if } p \in {}^\bullet t, x_p \in times(p,t) \\ & \textit{and } x_p \in M(p) \\ \emptyset & \textit{otherwise} \end{cases}$

- $C^+(t,p) = \begin{cases} \emptyset & \textit{if } p \notin t^\bullet \\ \{0\} & \textit{otherwise} \end{cases}$

Thus, from each precondition place of t we remove a token fulfilling (i), and we add a new token (0 aged) on each postcondition place of t.

As usual, we denote these evolutions by $M[t\rangle M'$, but it is noteworthy that these evolutions are in general non-deterministic, because when we fire a transition t, some of its precondition places could hold several tokens with different ages that could be used to fire it. Besides, we see that the firing of transitions does not consume any time. Therefore, in order to model the time elapsing we need the function age, defined below. By applying it we age all the tokens of the net by the same time:

(iii) The function $age : \mathcal{M}(N) \times \mathbb{R}_0^+ \longrightarrow \mathcal{M}(N)$ is defined by:

$$age(M, x)(p)(y) = \begin{cases} M(p)(y - x) & \textit{if } y \geq x \\ 0 & \textit{otherwise} \end{cases}$$

The marking obtained from M after x units of time without firing any transitions will be that given by $age(M, x)$.

Although we have defined the evolution by firing single transitions, this can be easily extended to the firing of *steps* or *bags* of transitions; those transitions that could be fired together in a single step could be also fired in sequence in any order, since no *aging* is produced by the firing of transitions. In this way we obtain step transitions that we denote by $M[R\rangle M'$. Finally, by alternating step transitions and time elapsing we can define a timed step semantics, where timed step sequences are those sequences $\sigma = M_0[R_1\rangle_{x_1} M_1 \ldots M_{n-1}[R_n\rangle_{x_n} M_n$, where M_i are markings, R_i multisets of transitions and $x_i \in \mathbb{R}_0^+$, in such a way that $M_i[R_{i+1}\rangle M'_{i+1}$ and $M_{i+1} = age(M'_{i+1}, x_{i+1})$. Note that we allow $x_i = 0$ in order to capture the execution in time zero of two causally related steps.

Then, given a MTAPN (N, M_0), we define $[M_0\rangle$ as the set of reachable markings on N starting from M_0, and we say that N is bounded if for every $p \in P$ there exists $n \in \mathbb{N}$ such that for all $M \in [M_0\rangle$ we have $|M(p)| \leq n$.

A token on a place p at a marking M is said to be *dead* if it can never be used to fire any transitions, i.e., it will remain on its place forever, just growing up as time elapses. Thus, we say that a marking is *dead* when either it has no tokens or all its tokens are *dead*.

In a previous paper [13] we have shown that TAPNs have a greater expressiveness than PNs, even although TAPNs are not Turing complete, because they cannot correctly simulate a 2-counter machine. In that paper we proved that reachability is undecidable for TAPNs. Other properties that we have studied in a more recent paper [5] are coverability, boundedness and detection of dead tokens, which are all decidable for TAPNs. Decidability of coverability has been also proved in [1] for an extended version of TAPNs, in which all arcs can be annotated with bags of intervals in $\mathbb{N} \times (\mathbb{N} \cup \{\infty\})$.

3 MPEG–2 Digital Video Coding Standard

The ISO/IEC 13818–2 standard [8], commonly known as MPEG–2, is a standard intended for a wide range of applications, including Video–on–Demand (VoD), High Definition TV (HDTV) and video communications using broadband networks.

The MPEG digital video coding techniques are statistical in nature. Video sequences usually contain statistical redundancies in both temporal and spatial directions. The basic statistical property upon which MPEG compression techniques rely is inter–pixel region correlation. The contents of a particular pixel region can be predicted from nearby pixel regions within the same frame (intra–frame coding) or from pixel regions of a nearby frame (inter–frame coding).

Perhaps the ideal method for reducing temporal redundancy is one that tracks every pixel from frame to frame. However, this extensive search is computationally expensive. Under the MPEG standards, this search is performed by tracking the information within 16×16 pixels regions, called macroblocks. Given two contiguous frames, $frame(t)$ and $frame(t-1)$, for each macroblock in $frame(t)$, the coder determines the best matching macroblock in $frame(t-1)$ and calculates the translation of the picture macroblock between frames, obtaining the motion vector. Using the corresponding macroblock from $frame(t-1)$, the temporal redundancy reduction processor generates a representation for $frame(t)$ that contains only the motion vector and the prediction error (changes between the two frames), see Fig. 2.

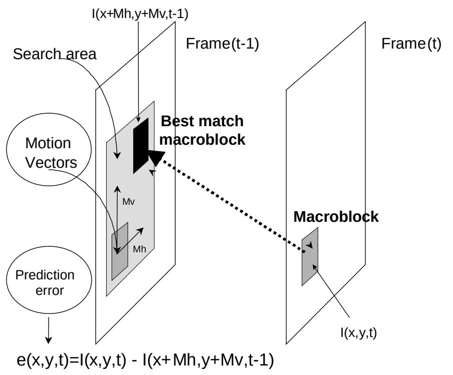

Figure 2. Motion vector and prediction error

This technique is called *motion compensated prediction*.

In order to reduce spatial redundancy a coding method, DCT (Discrete Cosine Transform), is used. A major objective of this *transform domain coding* is to make small enough as many transform coefficients as possible, so that they are insignificant and need not to be coded for transmission. Low DCT coefficients are related to low spatial frequencies within image blocks and high DCT coefficients to high frequencies. This property is used to remove subjective redundancies contained in the image data, taking into account the human visual systems criteria. Since the human viewer is more sensitive to reconstruction error related to low spatial frequencies than to high ones, a frequency adaptive weighting (quantization) of the coefficients, according to the human visual perception is often employed to improve the visual quality of the decoded images.

The combination of the two techniques described above, temporal motion compensated prediction and transform domain coding, are the key elements of the MPEG coding.

The MPEG–2 has to achieve the requirement of random access and high compression, thus this standard specifies three types of compressed video frames/pictures: I pictures, P pictures and B pictures. I pictures (intracoded pictures) are coded with no reference to other frames, exploiting only spatial correlation in a frame. They allow fast random access but offer moderate compression. P pictures (predictive coded pictures) are coded by using motion compensated prediction of a previous I or P picture. The compression for P pictures is higher than for I pictures. Finally, B pictures (bidirectionally–predictive coded pictures) are obtained by motion compensation from both past and future reference frames (I or P pictures), and provide the highest degree of compression.

A group of consecutive I, P and B pictures form a structure called Group of Pictures (GoP). A video sequence may be seen, then, as a sequence of GoPs. The pictures may be arranged in a sequence with a high degree of flexibility depending on the applications requirements. Thus, a video sequence coded using only I pictures allows the highest degree of random access, but achieves the lowest compression. A sequence code with I and P pictures (e.g. IPPPIPPP. . .) achieves moderate compression and certain degree of random access. Finally, a sequence that incorporates the three kinds of pictures (e.g. IBBPBBP. . .) may achieve high compression and reasonable random access, but also increases the coding delay significantly.

In order to understand how the MPEG–2 encoder works we will consider a GoP consisting of the frames IBBP[3]. Despite B pictures appear before P pictures, the coding order is IPBB because B pictures require both past and future frames of the original video sequence as references, see Fig. 3.

[3]Although the most common GoP is formed of the sequence IBBPBBPBBPBBPBBP, the GoP configuration here considered has all types of images and the study of the biggest one is quite long but similar in essence to the one here presented, then we have chosen the GoP IBBP.

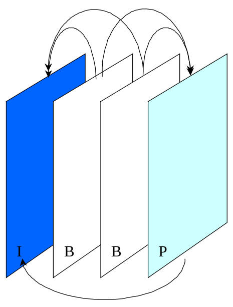

Figure 3. Encoding order for an IBBP GoP

First, for every kind of frame, an elementary compression technique which makes use of specific physiological characteristics of human eye may be used: the human eye is more sensitive to changes in brightness than to changes in chromaticity. Therefore the MPEG–2 coding schema first divide images into YUV components (one luminance and two chrominance components). Then the chrominance components are subsampled relative to the luminance component with a ratio specific to particular applications (e.g. 4:1:1)

The first frame in a GoP (I picture) is encoded in intra mode without references to any past or future frames. At the encoder the DCT is applied to each macroblock and then is uniformly quantized (Q). After quantization, it is coded using a variable length code (VLC) and sent to the output buffer. At the same time the reconstruction (IQ) of all non–zero DCT coefficients belonging to one macroblock and subsequent inverse DCT (IDCT) give us a compressed I picture which is stored temporarily in the Frames Store (FS), see Fig. 4.

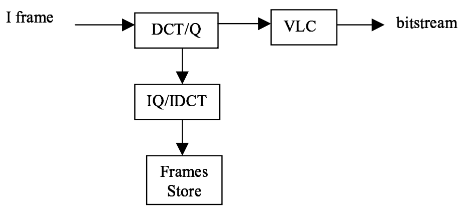

Figure 4. Block diagram for I pictures

If the input is coded either as P or B pictures, then the encoder does not code the picture macroblocks directly. Instead, it codes the prediction errors and the motion vectors.

With P pictures, for each macroblock in the current picture, the motion estimation gives us the coordinates of the macroblock in the I picture that best matches its characteristics and thus, the motion vector may be calculated. The motion compensated prediction error is obtained by subtracting each pixel in a macroblock with its motion shifted counterpart in the previous frame. The prediction error and the motion vectors are coded (VLC) and sent to the output buffer. As in the previous case, a compressed P picture is stored in the Frames Store, see Fig. 5.

With B pictures, the motion estimation process is performed twice: for a past picture (I picture in this case), and for a future picture (P picture). Prediction error and both motion vectors for each macroblock are coded (VLC) and sent to the output buffer. It is not necessary to store in FS the compressed B picture since it will never be taken as reference, see Fig. 6.

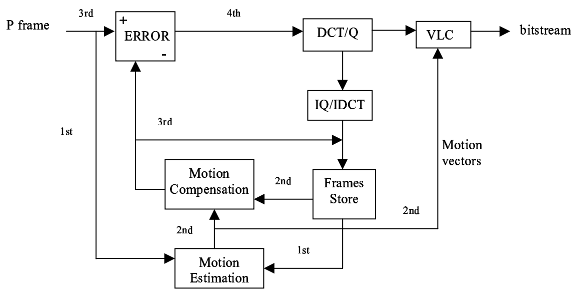

Figure 5. Block diagram for P pictures

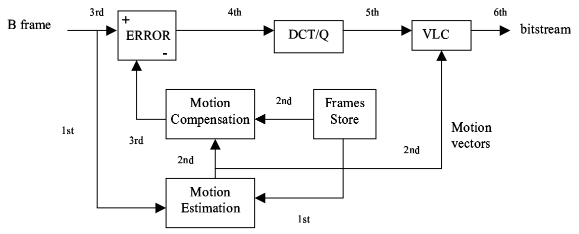

Figure 6. Block diagram for both B pictures

4 Timed-Arc Petri Net modelling the MPEG–2 encoding process of IBBP

The MATPN modelling the encoder is constructed by components. We first show the net of Fig. 7 which captures the encoding process described in the block diagram shown in Fig. 4, i.e., the *Discrete Cosine Transformation/Quantization*, **DCT/Q**, and both the *Variable Length Coding Process*, **VLC** (then it will be sent to the output buffer with no significative temporal cost), and the *Quantization/Discrete Cosine Transformation Inverses*, **IQ/IDCT** of one I-frame (finally, it will be copied into the Frames Store in order to be taken as reference for the encoding process of next P and B frames until next I one).

In the same way, nets in Figs. 8 and 9 capture the encoding process described in block diagrams of Figs. 5 and 6 respectively, which correspond with the encoding processes of P and B frames.

The Timed-Arc Petri Net of Fig. 11 models the whole MPEG–2 encoding algorithm[4]; The left-hand side of this Fig. 11 describes the first part of the encoding algorithm, which corresponds to the generation of both I and P encoded pictures (but remember that even if P pictures are generated before the B pictures, they will appear the last in the

[4]For simplicity, we omit in all these pictures the label of the arcs when they are labelled by $<0, \infty>$.

final video sequence). Once the places *I-B1, I-B2, P-B1, P-B2* are marked the second part of the net becomes activated (right-hand side), which models the *B1* and *B2* picture encoding process. *Iout* (*Pout* resp.) represents the output of an encoded *I-picture* (*P-picture*), while places *B1out* and *B2out* represent the output of B1 and B2 pictures.

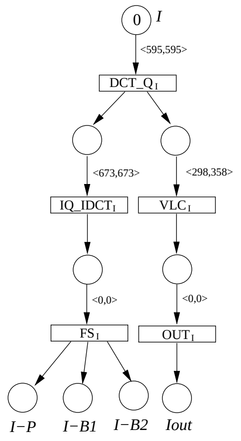

Figure 7. Partial TAPN modelling the encoding of frame I

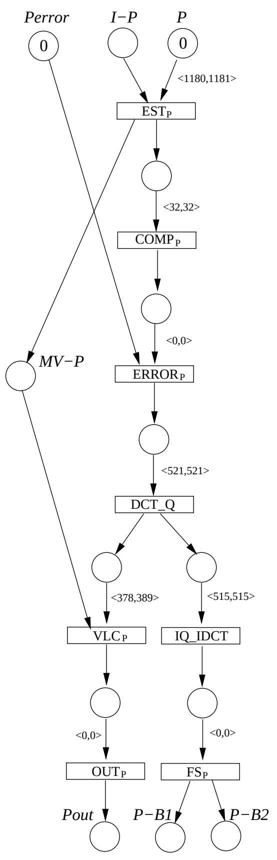

Figure 8. Partial TAPN modelling the encoding of frame P

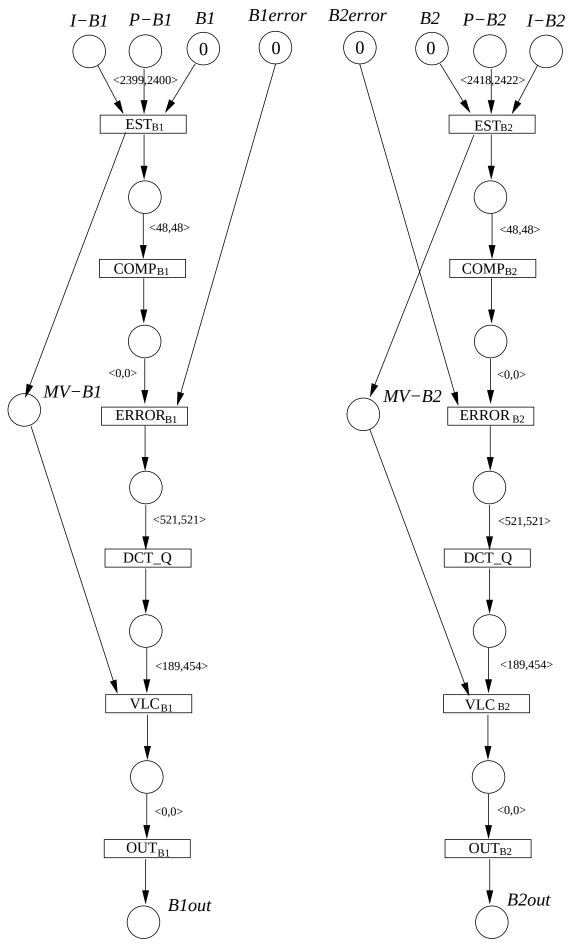

Figure 9. Partial TAPN modelling the encoding of both B-frames

The time intervals that label the arcs connecting places with transitions have been obtained from several real measurements, by coding the “Composed” Video sequence. This experiment has been repeated 3×10 times, and the results being reported below are therefore the minimum and the maximum of all these trials[5]. During these trials, no other operations were taken place in our experimental setup. The “Composed” Video sequence (format PAL CCIR601, 720x576 pixels) is a representative video sequence which has several different motion levels, and we have encoded it by using a completely software-based MPEG–2 video encoder derived from that developed in Berkeley, which is freely available in *ftp://mm-ftp.cs.berkley.edu/pub/multimedia/mpeg/encode*, other versions could be obtained from the MPEG Home Page, *http://www.mpeg.org*.

In order to get the real values for the different elements of the encoder we have included some patches into the source code, which correspond with the beginning and the end of the elementary actions that we have taken as transitions of

[5]These values were obtained in a single processor Pentium II - 350MHz platform with 64MB RAM.

the MTAPN which captures the algorithm. The real values thus obtained for I and P pictures being in the output buffer ($Iout$ and $Pout$ resp.) are shown in Tab. 1, as well as the times for encoding the complete GoP (both $Bout$ places).

Picture	Min	Average	Max
I	893 ms	918 ms	953 ms
P	3688 ms	3709 ms	3738 ms
GoP	11567 ms	11821 ms	12085 ms

Table 1. Measured real values

We may compute time the required to reach every place of a TAPN, by constructing a state reachability graph ([13]). With that process we obtain a time interval for every place (Fig. 10 captures how the TAPN partially evolves when time elapses).

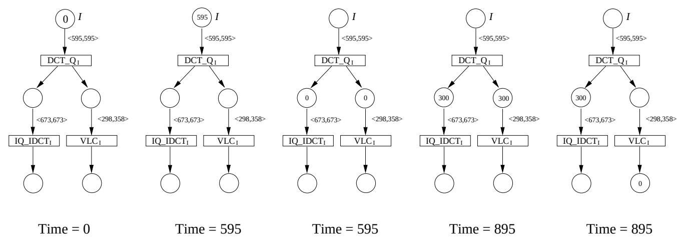

Figure 10. Partial TAPN evolution modelling I-frame coding while time elapsing

In our case, for the TAPN modelling the MPEG–2 encoder we have obtained the results shown in Tab. 2.

Comparing these results with those shown in Tab. 1 we can see that there are strong differences on the required time for completing the encoding process of a GoP. For instance, to complete the first GoP the measured values are $[11567\,ms, 12085\,ms]$ while in the TAPN we have obtained $[6672\ ms, 6962\ ms]$. These strong differences are due to the important fact that the encoder only uses a single processor, whereas our TAPN model captures all the intrinsic parallelism of the encoding process, so that with the analysis of the TAPN we are obtaining the required times provided that we have as many processors as needed to take advantage of this parallelism (for a single GoP two processors would be enough).

Let us now consider the encoding process of N GoPs, according to the TAPN the whole encoding would take $[6672 + 1268 * (N - 1)\ ms, 6962 + 1268 * (N - 1)\ ms]$, and thus, the average time for encoding each GoP would converge to 1268 ms. This can be interpreted as the time needed to encode a GoP for an infinite video sequence provided that the number of processors is big enough.

There have been studied the processors requirements by analyzing the MTAPN of Fig. 11 and the results so provided are given in the map of Fig. 12, which shows the number of processors required as function of the encoding time (it can be observed that at the beginning is required a single processor until $\sim 595ms$ when the place DCT_Q_I is reached, from this time to $\sim 900ms$ two processors are needed until place $Iout$ is reached then one of the procs. is released . . . from $\sim 6672ms$ on, when the couple of procs. which are encoding the first pair of B images are released, the procs. requirement map repeats the sequence with a period of $\sim 1268ms$) according to the TAPN of Fig. 11, i.e. by exploiting all the possible parallelism of the encoding process.

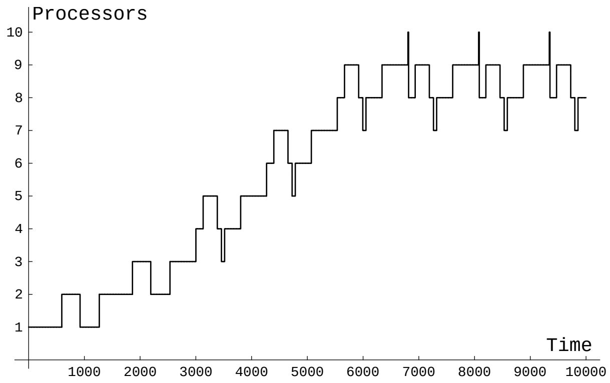

Figure 12. Graph relating the number of required processors and the encoding time (ms)

From that figure we conclude that the best performance with the GoP distribution IBBP, could be obtained with "only" **10 processors**.

5 Conclusions and Future Work

We have presented an application of Formal Methods, specifically Timed-Arc Petri Nets, to the performance analy sis of a real algorithm for video compressing, the MPEG-2.

The main conclusion from this analysis is that the performance of the encoding algorithm for a single GoP can be improved by exploiting the potential parallelism which can support the encoding process of any GoP, thus with some minor changes in the code (mainly by encoding both B-images in parallel) the performance can be improved in a factor near to 50% by using two processors.

Moreover, with an encoder allowing all the possible parallelism, according to Fig. 11, this improving factor could be increased about to 90%; specifically we have obtained an average time of 1268 ms (12085 ms measured in Tab. 1 for the original, *sequential*, version of MPEG–2) for encoding each single GoP of an infinite video sequence, provided that we have as many processors as required, 10 in this case.

The authors' current work is mainly focused on:

- Implementing the proposed improvements on a real set-up made up of eleven processors Pentium-IV.
- Analyzing some other GoP configurations in order to get the appropriate balance among compression factor, random access capability, processors requirements and time performance.

Picture	Min 1st GoP	Average 1st GoP	Max 1st GoP	Min Nth GoP	Average Nth GoP	Max Nth GoP
I	893 ms	918 ms	953 ms	893 + 1268 * (N-1) ms	918 + 1268 * (N-1) ms	953 + 1268 * (N-1) ms
P	3379 ms	3386 ms	3391 ms	3379 + 1268 * (N-1) ms	3386 + 1268 * (N-1) ms	3391 + 1268 * (N-1) ms
GoP	6672 ms	6828 ms	6962 ms	6672 + 1268 * (N-1) ms	6828 + 1268 * (N-1) ms	6962 + 1268 * (N-1) ms

Table 2. Values obtained from the TAPN

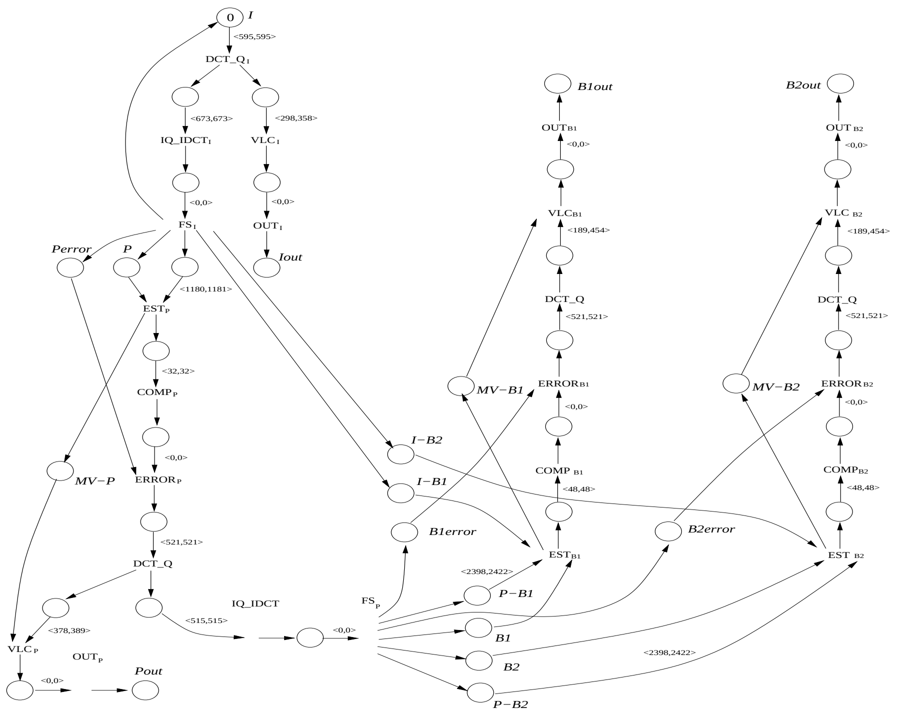

Figure 11. TAPN which models the MPEG–2 encoding process of a GoP

References

[1] P. Abdulla and A. Nylén. Timed Petri Nets and BQOs. *Proc. ICATPN 2001, Lecture Notes in Computer Science*, 207(5):53–70, 2001.

[2] S. M. Akramullah, I. Ahmad, and M. L. Lion. Performance of software-based MPEG-2 video encoder on parallel and distributed systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(4):687–695, 1997.

[3] F. Bowden. Modelling time in Petri nets. In *Proc. of Second Australia-Japan Workshop on Stochastic Models*, 1996.

[4] A. Cerone and A. Maggiolo-Schettini. Time-based expressivity of time petri nets for system specification. *Theoretical Computer Science*, 216(1-2):1–53.

[5] D. de Frutos, V. Valero, and O. Marroquín. Decidability of properties of timed-arc Petri nets. *Proc. of ICATPN 2000, Lecture Notes in Computer Science*, 1825:187–206, 2000.

[6] W. V. der Aalst. Interval timed coloured petri nets and their analysis. *Lecture Notes in Computer Science*, 691:451–472, 1993.

[7] H.-M. Hanisch. Analysis of place/transition nets with timed-arcs and its application to batch process control. *Application and Theory of Petri Nets, Lecture Notes in Computer Science*, 691:282–299, 1993.

[8] ISO/IEC 13818–2 Draft International Standard Generic Coding of Moving Pictures and Associated Audio. Recommendation H.262.

[9] T. Olivares, F. Quiles, A. Garrido, P. Garcia, and L. Orozco-Barbosa. The need of multicast predictive nfs servers for high-speed networks used as parallel multimedia platforms. In *ICPP´ 01, IEEE Computer Society Press*, pages 391–396, 2001.

[10] F. L. Pelayo, F. Cuartero, V. Valero, and D.Cazorla. Analysis of the MPEG-2 encoding algorithm with ROSA. *Electronic Notes on Theoretical Computer Science*, 80(To appear), *http://www.elsevier.nl/locate/entcs/volume80.html*.

[11] C. Ramchandani. *Performance Evaluation of Asynchronous Concurrent Systems by Timed Petri Nets*. PhD. Thesis, Massachusetts Institute of Technology, Cambridge, 1973.

[12] J. Sifakis. Use of petri nets for performance evaluation. *Proc. of the Third International Symposium IFIP W.G.7.3., Measuring, Modelling and Evaluating Computer Systems. Elsevier Science Publishers*, pages 75–93, 1977.

[13] V. Valero, D. de Frutos, and F. Cuartero. On non-decidability of reachability for timed-arc petri nets. In *Proc. 8th Workshop on Petri Nets and Performance Models, PNPM'99*, pages 188–196, 1999.

[14] V. Valero, F. L. Pelayo, F. Cuartero, and D. Cazorla. Specification and analysis of the MPEG-2 video encoder with Timed-Arc Petri Nets. *Electronic Notes on Theoretical Computer Science*, 66(2):125–136, *http://www.elsevier.nl/locate/entcs/volume66.html*, 2002.