Specification and Analysis of the MPEG–2 Video Encoder with Timed-Arc Petri Nets [1]

Valentín Valero [2] Fernando L. Pelayo [3] Fernando Cuartero [4]
Diego Cazorla [5]

Departamento de Informática
Universidad de Castilla-La Mancha
Escuela Politécnica Superior de Albacete. 02071 - SPAIN

Abstract

Petri nets are a very suitable model for the description and analysis of concurrent systems. Several timed extensions of Petri nets have been defined to capture some additional aspects, concerning with the behaviour in time of the described systems. In this paper we illustrate the use of timed-arc Petri nets for the modelling of timed concurrent systems, using the MPEG–2 video encoder as an example. From the analysis of the model we conclude that the performance of the encoding process could be improved by introducing some minor changes on the encoder.

1 Introduction

Petri nets are a very suitable model for the description and analysis of concurrent systems. They have a graphical nature and they have a solid mathematical foundation supporting them. Furthermore, one of the main advantages of Petri nets is that they capture *true concurrency*, i.e., they are able to model the simultaneous execution of actions in the system. A Petri net consists of a set of places and transitions, as well as a set of arcs connecting places with transitions and transitions with places. Places and transitions are called the nodes of the net; then, when there is an arc connecting a pair of nodes (x, y), we say that x is a precondition of y and y is a postcondition of x. In order to capture the dynamic behaviour of the modelled system places are annotated

[1] This work has been supported by the CICYT project "Performance Evaluation of Distributed Systems",TIC2000-0701-C02-02.
[2] Email:valentin@info-ab.uclm.es
[3] Email:fpelayo@info-ab.uclm.es
[4] Email:fernando@info-ab.uclm.es
[5] Email:dcazorla@info-ab.uclm.es

with a number of *tokens*. Thus, markings are defined as functions from P to $\mathbb{N}$, where P is the set of places of the net.

However, Petri nets do not consider quantitative aspects in the specifications, like the time required for the execution of transitions. Then, in the last three decades, some effort has been made in order to include time in Petri nets. A survey of the different approaches to introduce time in Petri nets is presented in [5]. The first group of models assign time delays to transitions, either using a fixed and deterministic value [11,12] or choosing it from a probability distribution [3]. Other models use time intervals to establish the enabling times of transitions [10]. Some models introduce time on tokens [2,4,13]; here, 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. Occurrence of a transition has 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 [6] 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 (TAPNs) [1,4,7,8,13] 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*, in the sense that they will never be available, since they are too old to fire any transitions in the future. The interpretation and use of Timed-Arc 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 every involved processes 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 and workflow systems. In this paper we show the use of TAPNs for the modelling of concurrent systems. As an illustration, we model the MPEG–2 Video Encoder by using TAPNs. The

ISO/IEC 13818–2 standard [9], 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 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.

Then, we have modelled this encoding algorithm with TAPNs, and from the analysis of the model we have concluded that a better performance could be obtained in the encoding process by introducing some minor changes into 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 codification of the B-images.

The paper is structured as follows. In Section 2 we present Timed-Arc Petri nets and their semantics, in Section 3 we describe the MPEG–2 encoding algorithm and the corresponding TAPN that models it. Finally, the analysis of the algorithm and some conclusions are presented in Section 4.

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 a time interval associated with them. The interval limits the age of the tokens that are needed 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 these tokens are about to become too old for any transition to consume. More generally, in the model we consider [6] 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 external context 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. Furthermore, it is possible that some tokens become *dead*, that is definitely useless because their increasing age will not allow the firing of any of their postcondition transitions in the future.

[6] Other proposals of timed-arc Petri nets [8] enforce the firing of transitions with an earliest and maximal firing rule.

Definition 2.1 Timed-arc Petri nets
We define a Timed-Arc Petri net (TAPN) as a tuple [7] $N = (P, T, F, \mathit{times})$, 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.:

$$\mathit{times} \,:\, F \cap (P \times T) \longrightarrow \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$$

When $\mathit{times}(p,t) = (x_1, x_2)$ we write $\pi_i(p,t)$ to denote x_i, for $i = 1, 2$. We will also say that $x \in \mathit{times}\,(p,t)$ if and only if $\mathit{times}(p,t) = (x_1, x_2)$ and $x_1 \leq x \leq x_2$.

As we previously mentioned, tokens are annotated with real values, so markings are defined by means of multisets over $\mathbb{R}_0^+$. 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 (bags) of non-negative real numbers [8]. Thus, as usual, each place is annotated with a certain number of tokens, but each one of them has an associated 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 at a marking M 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 marking we get an infinite reachability set of markings, due to the *token aging*.

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 place.

In Fig. 1 we show a MTAPN modelling a producer/consumer system, where we have represented 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

[7] We consider only arcs with weight 1 to simplify some definitions, but the extension to general arcs with greater weights is straightforward.
[8] Using classical notation, we consider that a multiset B over a set X is defined as a function $B \,:\, X \to \mathbb{N}$.

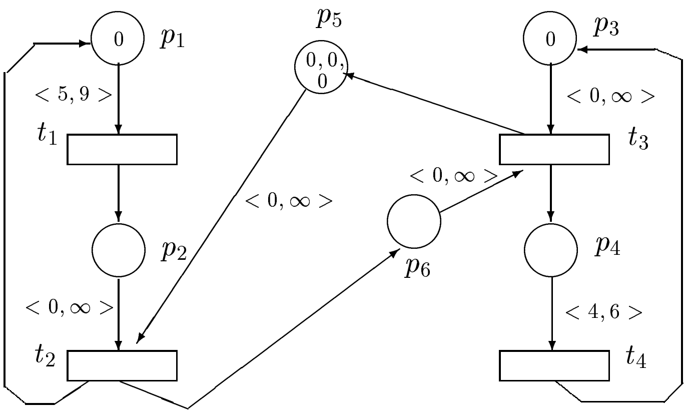

Fig. 1. Timed-Arc Petri net modelling a producer-consumer problem

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 p_4.

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

Definition 2.2 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 the marking has some token the age of which 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 for each place p as follows:

$$M'(p) = M(p) - C^-(p, t) + C^+(t, p)$$

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

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

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

Thus, from each precondition place of t we remove a token fulfilling (i), and we add a new token with age 0 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:

$\forall M \in \mathcal{M}(N), \forall x, y \in \mathbb{R}_0^+, \forall p \in P$:

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

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 $\sigma = M_0[R_1\rangle_{x_1} M_1 \dots M_{n-1}[R_n\rangle_{x_n} M_n$, 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 all the x_i be 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 $|M(p)| \leq n$.

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 [7] 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 Encoding Algorithm

MPEG standards were designed with two requirements in mind, namely, the need for a high compression, and the need for random access capability. These techniques exploit the fact that video sequences usually contain statistical redundancies in both temporal and spatial directions. Thus, MPEG digital video coding techniques are statistical in nature. Specifically, 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 redundancies 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 encoder determines the best matching macroblock in $frame(t-1)$ and calculates the *motion vector*, which captures the macroblock translation information. Therefore, the *temporal redundancy reduction processor* generates a representation for $frame(t)$ by using the corresponding macroblock from $frame(t-1)$, and this representation only contains the motion vector and the prediction error (changes between both frames). This technique is called *motion compensated prediction*.

In order to reduce spatial redundancies a DCT (Discrete Cosine Transform) is used. With this coding process some subjective redundancies in the image are removed, on the basis of human visual criteria.

The combination of these two techniques described above are the key elements of the MPEG encoding process. Furthermore, in order to achieve the requirement of random access and high compression, the MPEG–2 standard specifies three types of compressed video frames: 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. P pictures (predictive coded pictures) are coded by using motion compensated prediction of a previous I or P picture. Finally, B pictures (bidirectionally–predictive coded pictures) are obtained by motion compensation by using past and future reference frames (I or P pictures).

A group of consecutive I, P and B pictures constitute a structure called Group of Pictures (GoP). Therefore, a video sequence may be seen as a sequence of GoPs.

The block diagram of the MPEG encoder is depicted in Figure 2. In order to understand how the MPEG–2 encoder works, we will consider a typical GoP consisting on the frames IBBP. Despite the B pictures appearing before the P picture, the encoding order is IPBB because B pictures require both past and future frames as references.

The first frame in a GoP (I picture) is encoded in intra mode without references to any past or future frames. The DCT is applied to each macroblock and then it is uniformly quantized (Q). After quantization, it is encoded using a variable length code (VLC) and it is sent to the output buffer. At the same time the reconstruction (IQ) of all non-zero DCT coefficients belonging to one

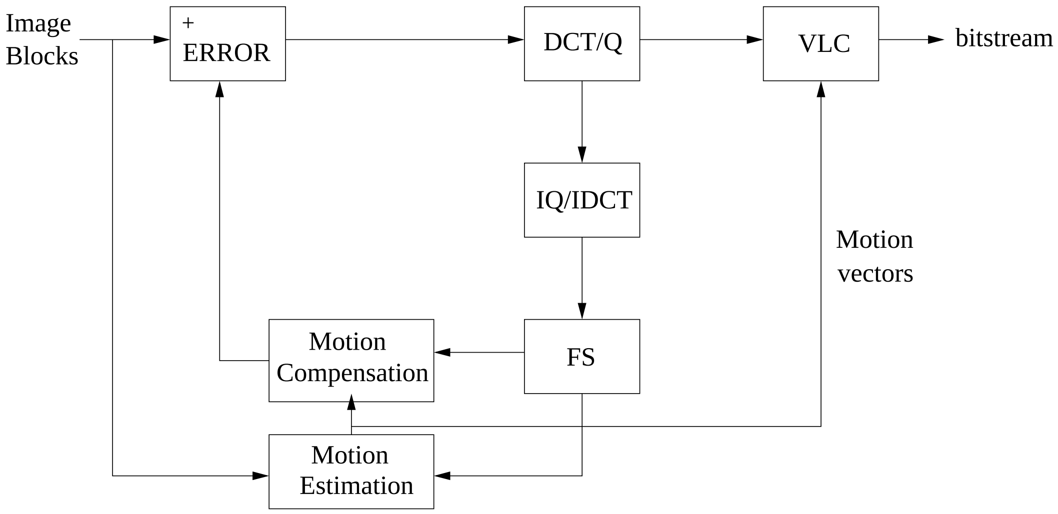

Fig. 2. Block diagram of the MPEG–2

macroblock and the Inverse DCT (IDCT) give us a compressed I picture which is stored temporarily in the *Frame Store (FS)*.

When the input is coded either as P or B pictures, 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 Frame Store.

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 errors and both motion vectors for each macroblock are coded (VLC) and sent to the output buffer. Notice that the compressed B pictures are not stored in the Frame Store, since they are not needed to calculate any other pictures.

3.1 Timed-Arc Petri Net modelling the MPEG–2

Figure 3 shows the Timed-Arc Petri net modelling the MPEG–2 encoding algorithm [9]. The left side figure 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 final video sequence). Once the places *I-B1,*

[9] For simplicity, we omit in this picture the label of the arcs when they are labelled by $<0,\infty>$.

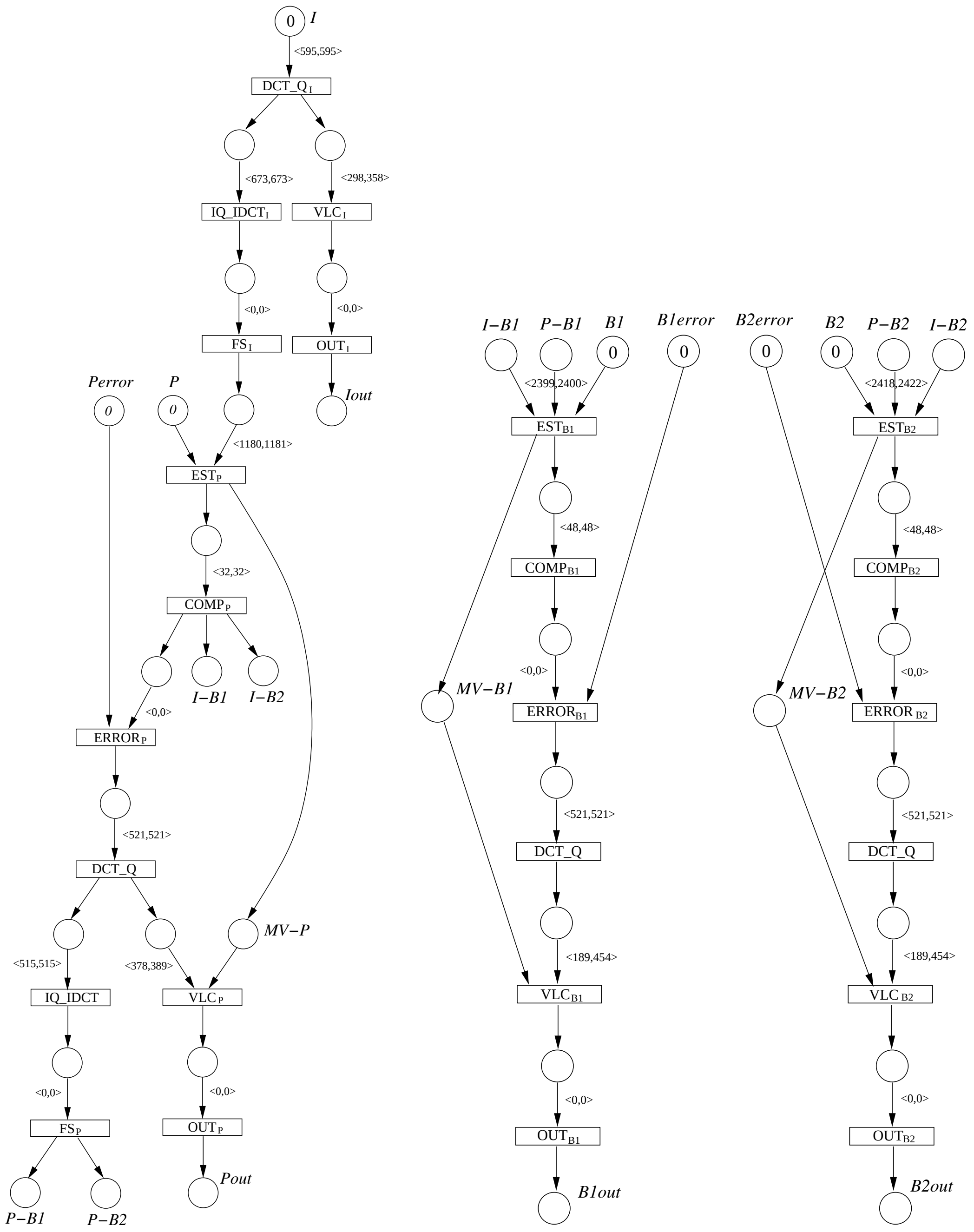

Fig. 3. TAPN modelling the MPEG–2

I-B2, P-B1, P-B2 are marked the second part of the net becomes activated (right part figure, where these places have been pictured again in that figure for a better readability), which models the *B1* and *B2* picture encoding process. *Iout* (respectively, *Pout*) represents the output of an encoded *I picture* (*P picture*), while places *B1out* and *B2out* represent the output of B1 and B2 pictures.

The time intervals that label the arcs connecting places with transitions have been obtained from several real measurements, by coding the so called

"Composed" video sequence. This experiment has been repeated a number of times, and the results being reported below are the minimum and the maximum of all these trials [10]. During these trials, no other operations were taking 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 the MPEG Home Page: http://drogo.cslet.stet.it/mpeg.

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 described in the specification of the algorithm.

The real values thus obtained for I and P pictures reaching the output buffer are shown in Table 1 (measured values), as well as the times for encoding the complete GoP.

Table 1
Times for encoding the pictures

	Measured		TAPN	
Picture	Min	Max	Min	Max
I	893 ms	953 ms	893ms	953 ms
P	3688 ms	3738ms	3379ms	3391ms
GoP	11567 ms	12085 ms	6673ms	6962ms

4 Analysis and Conclusions

We may compute the times to reach every place of a TAPN, by constructing a state reachability graph. This graph can be constructed for bounded Timed-Arc Petri nets (see [13] for a detailed description); this construction is based on the fact that for every place p of a Timed-Arc Petri net we can find a maximal value $Max(p)$ for the age of its tokens to influence the activation of its postcondition transitions, because the tokens on that place with an age exceeding that maximal value can only be consumed by the firing of some transition $t \in p^\bullet$ for which $\pi_2(p,t) = \infty$.

Concretely, we may define:

$$Max(p) = Max\{\pi_i(p,t) \,|\, t \in p^\bullet,\ \pi_i(p,t) < \infty,\ i = 1,2\}$$

Obviously, in order to fire such a transition t the age of the involved token on p is unimportant once it exceeds $Max(p)$. This means that in order to construct

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

the state graph we can represent by the single value $1 + Max(p)$ the whole interval $[1 + Max(p), \infty]$.

In our case, with that state graph we may obtain a time interval for reaching every place of the TAPN, concretely we have obtained the results shown in the last two columns of Table 1 (TAPN values). Notice that the time required to reach the place *Iout* (respectively, *Pout*) is the time required to generate the I (respectively, P) picture. Furthermore, the time required for encoding the complete GoP is the time required to reach both places *B1out* and *B2out*.

We can observe that the most significant differences with the measured values appear in the times obtained for completing the GoP encoding. These 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. In our case two processors would be enough.

Consequently, the main conclusion of this analysis of the MPEG–2 encoder by using TAPNs is that the performance of the encoding algorithm can be improved by a factor of nearly 50% by using two processors. Moreover, in bigger GoPs this improvement factor increases a little more, since the major cost of the encoding process is due to the B-images, which can be encoded in parallel.

Acknowledgement

The authors would like to thank the reviewers for their comments and suggestions, which have helped to improve this paper significantly.

References

[1] P.A. Abdulla and A. Nylén. *Timed Petri Nets and BQOs.* In Proc. ICATPN 2001, Lecture Notes in Computer Science, vol. **2075**(2001), pp. 53–70.

[2] W.M.P. van der Aalst. *Interval Timed Coloured Petri Nets and their Analysis.* Lecture Notes in Computer Science, vol. **691** (1993), pp. 451–472.

[3] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte and A. Cumani. *On Petri Nets with Stochastic Timing.* Proc. of the International Workshop on Timed Petri Nets, IEEE Computer Society Press, pp. 80–87. 1985.

[4] T. Bolognesi, F. Lucidi and S. Trigila. *From Timed Petri Nets to Timed LOTOS.* Proc. of the Tenth International IFIP WG6.1 Symp. on Protocol Specification, Testing and Verification. 1990.

[5] Fred D.J. Bowden. *Modelling time in Petri nets.* Proc. Second Australia-Japan Workshop on Stochastic Models. 1996.

[6] Antonio Cerone and Andrea Maggiolo-Schettini. *Time-based expressivity of time Petri nets for system specification.* Theoretical Computer Science, vol. **216**(1999), pp. 1-53.

[7] D. de Frutos, V. Valero and O. Marroquín. *Decidability of Properties of Timed-Arc Petri Nets.* Proc. ICATPN 2000, Lecture Notes in Computer Science, vol. **1825** (2000), pp. 187–206.

[8] Hans-Michael Hanisch. *Analysis of Place/Transition Nets with Timed-Arcs and its Application to Batch Process Control.* Application and Theory of Petri Nets, LNCS vol. **691** (1993), pp:282–299.

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

[10] P. Merlin. "A Study of the Recoverability of Communication Protocols". PhD. Thesis, Univ. of California. 1974.

[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, pp. 75–93. 1977.

[13] V. Valero, D. de Frutos and F. Cuartero. *On Non-decidability of Reachability for Timed-Arc Petri Nets.* Proc. 8th Workshop on Petri Nets and Performance Models, PNPM'99, pp. 188–196. 1999.