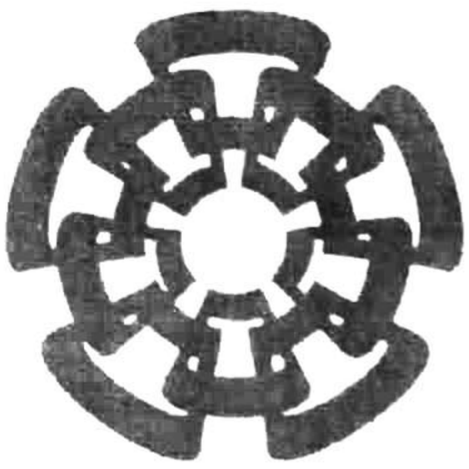
DO NOT COPY
ISSN 1548-7741
2012 ACM



Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional
Unidad Guadalajara

Modelado y Scheduling Térmico para MPSoCs usando Redes de Petri Continuas Temporizadas

Tesis que presenta:

Gaddiel Desirena López

para obtener el grado de:

Maestro en Ciencias

en la especialidad de:

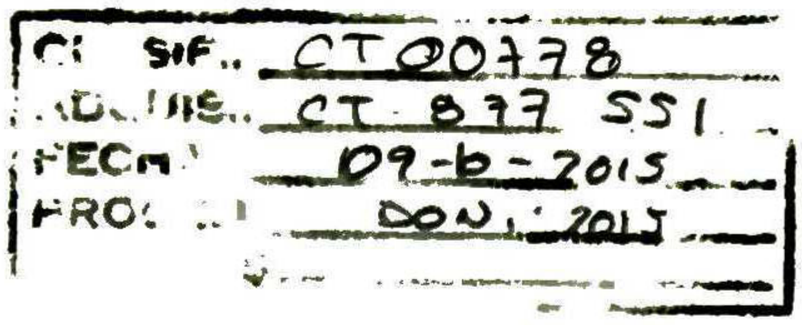
Ingeniería Eléctrica

CINVESTAV
IPN
ADQUISICION
LIBROS

Director de Tesis

Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, Guadalajara, Jalisco, Diciembre de 2014.


CL SIF.. CT 20778
..DLINE.. CT 877 SSI
FECH 09-b-2015
PRO.. DOJ 2015

Modelado y Scheduling Térmico para MPSoCs usando Redes de Petri Continuas Temporizadas

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:

Gaddiel Desirena López

Ingeniero en Mecatrónica

Universidad Tecnológica de la Mixteca 2006-2011

Becario de CONACYT, expediente no. 282123

Director de Tesis

Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, Diciembre de 2014.

Resumen

En este trabajo se propone una metodología para el modelado térmico de un sistema de multiprocesadores en chip ($MPSoC$) y una metodología para el modelado continuo de un conjunto de tareas periódicas, ambas usando Redes de Petri Continuas Temporizadas ($TCPN$). El modelo térmico del $MPSoC$ captura la generación de calor, conducción, convección y la potencia consumida por la ejecución de tareas de una manera sencilla evitando la etapa de calibración (identificación de parámetros) que utilizan algunas otras técnicas de modelado térmico. La metodología de modelado térmico propuesta se basa en técnicas de diferencias finitas, donde la precisión de la solución se determina por el número de lugares de la red de Petri. La metodología de modelado de tareas propuesta considera los tiempos de ejecución, periodos y tiempos límite de entrega que son representados como tasas de disparos en las transiciones de la red de Petri. Utilizando el enfoque propuesto, el modelo térmico del $MPSoC$ y el modelo de un conjunto de tareas se fusionan para obtener un único modelo global. Con el fin de demostrar la utilidad del modelo derivado, este trabajo muestra dos posibles schedulers térmicos, uno basado en la estrategia de control predictivo (MPC) y el otro basado en un esquema de control continuo.

Abstract

This work proposes a thermal modeling methodology for Multiprocessor Systems on a Chip ($MPSoC$) and a fluidized modeling methodology for a set of periodic tasks, both using Timed Continuous Petri Nets ($TCPN$) as a formal representation tool. The $MPSoC$ thermal model captures heat generation, conduction, convection and power consumption due to task executions in a simple manner avoiding the tuning stage (parameter identification) of previous reported modeling techniques. The proposed methodology is related to finite difference techniques where the solution's precision is determined by the number of Petri net places. The task fluidized modeling methodology represents execution times, periods and due dates of tasks as firing transition rates. Using the proposed approach, both the $MPSoC$ and the set of tasks executed in the $MPSoC$ are represented in a single formalism, simplifying further analysis stages. In order to show the usefulness of the derived model, this paper shows two possible thermal-aware schedulers, one based on a Model Predictive Control (MPC) strategy and the other based on a continuous control scheme.

Agradecimientos

A mis padres Cristino y Emilia, con quienes estaré toda mi vida agradecido por su inmenso cariño, comprensión y apoyo.

A mis hermanos Jalil y Cristian, por los momentos divertidos y agradables que me han otorgado, por compartir conmigo las diferentes etapas de mi vida y por siempre estar ahí cuando más los necesito.

A Elena, por darme la oportunidad de vivir muchos días felices, por todo su apoyo incondicional, sus consejos y por siempre creer en mi.

A mis tíos y primas Jesús, Rosalinda, Linda y Lupe, por el apoyo incondicional en todo momento y por nunca dejar de animarme a continuar en este camino.

A mi asesor el Dr. Antonio por la orientación, paciencia y dedicación para poder llevar a cabo este trabajo.

A mis compañeros y amigos de mi generación, por la amistad que me han brindado durante este tiempo.

Finalmente, agradezco a CONACYT por el apoyo económico brindado durante el desarrollo de mi maestría.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El desarrollo de la ciencia y tecnología en la creación de dispositivos diversos conducen a la búsqueda de sistemas que operen con mayor capacidad de cómputo. En la última década, el desarrollo para nuevas arquitecturas en sistemas con procesadores se han ido reemplazando por la integración de un mayor número de procesadores dentro de un solo chip, formando lo que se conoce como los sistemas multiprocesadores en chip o *MPSoC*'s por sus siglas en inglés.

Los *MPSoC*'s satisfacen las necesidades de procesamiento actuales; además son una solución eficiente a algunas limitantes tecnológicas. Actualmente son los más utilizados para aplicaciones modernas compuestas por sistemas complejos, como sistemas multimedia, telefonía móvil, aplicaciones de red, etc. Sin embargo, ante la complejidad de las aplicaciones existentes, se tiene como resultado un incremento de potencia, lo que está directamente relacionado con el aumento de temperatura en los procesadores. Como consecuencia de este incremento, el rendimiento y la vida de uso de los procesadores disminuye. A mayor aumento en la potencia generada, mayor aumento de temperatura, menor rendimiento y mayor costo en el sistema. La temperatura máxima en los chips es una restricción en la etapa de diseño de un *MPSoC*. Datos recientes muestran que más del 50 % de todas las fallas en los circuitos integrados están relacionados con problemas térmicos.

Como resultado, el control de la temperatura en *MPSoC*'s se ha convertido en una de las principales preocupaciones en el diseño de sistemas electrónicos. Con el fin de controlar eficazmente esta variable, actualmente se hacen consideraciones térmicas en todas las fases de diseño, desde el proceso de fabricación hasta la etapa de implementación. Dada la importancia de la temperatura en estos dispositivos, varios investigadores se han centrado en el análisis térmico y la forma de controlar la generación de calor en *MPSoC*'s obteniendo técnicas diversas.

Las técnicas que se usan para el control de temperatura en sistemas electrónicos son generalmente llamados técnicas de gestión térmica (DTM por sus siglas en inglés), las cuales pueden clasificarse en técnicas en el diseño o técnicas sobre ejecución. Las técnicas sobre ejecución, comúnmente tienen como objetivo alcanzar un máximo rendimiento bajo un límite de temperatura máximo. No obstante para el uso de dichas técnicas, es necesaria una herramienta para modelar la temperatura o bien disponer de sensores en todo el sistema para su medición, lo que resulta sumamente difícil. Así, Muchos de los trabajos relacionados con estas técnicas optan por crear un modelo que describa el comportamiento térmico de un $MPSoC$.

Actualmente, modelos térmicos de $MPSoC$'s han sido desarrollados usando métodos de diferencias finitas, elementos finitos, volúmenes finitos [1],[2] ó análogamente usando modelos de circuitos RC [3],[4],[5],[6]. Todas estas técnicas conducen a modelos térmicos muy exactos. Sin embargo, los métodos numéricos como las diferencias finitas o elementos finitos no proveen un modelo en espacio de estados, y por lo tanto los resultados obtenidos no pueden utilizarse en etapas de control. Por otro lado, los métodos basados en RC requieren una etapa de calibración que tiene que ver con la experiencia empírica del ingeniero.

1.2. Trabajos Previos

Es bien sabido que la temperatura juega un papel muy importante en el diseño de dispositivos electrónicos. Por lo que, no considerar un manejo eficiente de la temperatura en estos dispositivos puede producir serios problemas. Las técnicas actuales para el control de temperatura en procesadores se les conoce como técnicas de *gestión dinámica térmica (DTM)*, estas técnicas comúnmente están diseñadas para hacer frente a los problemas de control de temperatura y el consumo de energía en los chips. Mientras se tenga una buena regulación en la temperatura, la confiabilidad en el sistema puede mejorar.

Las DTM se refieren a aquellas técnicas que permiten modificar automáticamente las características de un procesador en tiempo de ejecución, como las variables de voltaje, frecuencia y potencia de disipación, de modo que se garantice el cumplimiento de las tareas mientras se regule la temperatura. Muchas técnicas de DTM obtienen la información en tiempo de ejecución y por medio de ésta se toman acciones para gestionar la temperatura. La información más importante que se requiere para llevar a cabo una toma de decisión es la temperatura; esta información se puede medir o estimar utilizando un modelo térmico. Un gran número de técnicas para la gestión térmica han sido estudiadas y muchas de éstas utilizan un circuito RC equivalente para modelar la temperatura [7], [8], [9], [10]. La dualidad que existe entre la transferencia de calor y el fenómeno eléctrico hace que este modelo sea fácil de entender y de implementar. Un modelo RC considera el flujo de calor como una corriente que pasa a través de una resistencia térmica, creando una diferencia de temperatura análoga al voltaje. El modelo que proponen en [7] el cual dan por nombre HotSpot, toma

ventaja de la dualidad entre sistemas eléctricos y térmicos. Aquí, cada unidad del chip es modelada como un circuito RC, obteniendo un modelo con cualquier granularidad deseada. Sin embargo, la derivación del modelo principalmente se basa en el cálculo de la capacitancia térmica y las resistencias térmicas relacionadas a la conductividad térmica. La desventaja en este modelo reside en que la capacitancia térmica es calculada por los parámetros isotrópicos del material, pero además se requiere de un factor de escalamiento que es un valor que se obtienen empíricamente.

Las técnicas para el control de temperatura se pueden encontrar en la literatura; entre ellas se encuentran: al clock gating [11], dynamic voltage frecuency scaling (DVFS) [12], task migration [13] y algunos métodos híbridos [14], [15], [16], [17] que combinan una o más técnicas mencionadas. Aunque estas técnicas son diferentes y son usadas en sistemas con características distintas, todos preservan la misma idea, esto es modificar la potencia consumida por el procesador de tal forma que la generación de calor sea mínima y exista una distribución de temperatura uniforme en el sistema.

En [13] se proponen algunos esquemas para el scheduling de tareas tomando en cuenta la temperatura. Una de las heurísticas que propone es la llamada *coolest* y es la más simple ya que asigna una tarea activa al procesador con menor temperatura. Una segunda propuesta es el *coolest-FLP*, donde el principio es el mismo, pero adicionalmente el scheduler asigna una mayor prioridad a los procesadores que tienen vecinos ociosos. Dado que estos vecinos no transfieren calor, entonces la prioridad de asignar una tarea al procesador se incrementa. Un tercer esquema propuesto es el *Adaptive-Random*, este proporciona una política probabilista que actualiza la probabilidad de asignar una carga de trabajo a los procesadores y está basado en el análisis de un historial de temperatura del procesador. En [18] se propone un algoritmo de escalamiento de frecuencia y voltaje en el procesador, aquí se considera que la relación entre la potencia, el voltaje y la frecuencia está dada por la ecuación $P \alpha V_{dd}^2 f$, lo que implica que con sólo escalar dinámicamente la frecuencia y el voltaje es posible reducir la temperatura. Otro enfoque propuesto por Brooks y Martonosi en [14] es el uso de técnicas microarquitecturales que regulan el ancho de una instrucción al procesador. En [9] se propone un scheduling para temperatura en sistemas de tiempo real mediante MILP (Mixer-integer linear programing), este algoritmo consiste en encontrar una solución óptima mediante la programación lineal entera. Sin embargo, esta formulación proporciona una buena solución para problemas con instancias muy pequeñas, el tiempo de simulación para un número de procesadores mayor a 20 se vuelve excesivamente largo.

1.2.1. Metodología Propuesta

En este trabajo se propone una metodología de modelado que obtiene un modelo térmico de un $MPSoC$ usando Redes de Petri Continuas Temporizadas ($TCPN$). Esta metodología divide al $MPSoC$ en pequeños volúmenes de control los cuales están representados por un

lugar p_i en la red, donde los fenómenos de conducción, convección y generación de calor ocurren en cada volumen de control y pueden representarse mediante módulos térmicos en una red de Petri. Después, considerando que existe la superposición de los fenómenos térmicos, todos los módulos térmicos en redes de Petri se fusionan en un único modelo global que representa el fenómeno térmico del $MPSoC$. El modelo que se obtiene hereda todas las ventajas de las redes de Petri, tales como una representación gráfica y una fuerte teoría matemática de fondo. Más aún, el cálculo de EDO's (ecuaciones diferenciales ordinarias) corresponde a una discretización central en el espacio, mientras que en el tiempo se mantiene continuo, por lo que el modelo también exhibe ventajas en comparación al enfoque de diferencias finitas, pero evitando la etapa de calibración. Además, el enfoque propuesto conduce a un espacio de estados de la temperatura en el $MPSoC$. Por lo tanto, la metodología propuesta puede utilizarse para el diseño de control o schedulers térmicos.

Con la finalidad de aplicar un scheduler térmico en el $MPSoC$ se propone una metodología de modelado con el mismo formalismo ($TCPN$) para el arribo y ejecución de tareas periódicas en los procesadores del $MPSoC$. Los parámetros temporales de las tareas son representados en este modelo. El periodo se modela como la creación de una nueva instancia de tarea a una tasa que está en función del periodo de la tarea. El tiempo límite de entrega (deadline) se usa para estimar las tareas que terminarían su ejecución después de su deadline. Los ciclos de cómputo (ciclos de CPU) están representados por las tasas de disparo de las transiciones las cuales generan calor al ser disparadas. Así, este modelo se fusiona con el modelo térmico, por lo que un solo modelo en $TCPN$ modela el tiempo de ejecución de las tareas, el consumo de potencia, la generación de calor, la conducción y convección en el $MPSoC$. Esta fusión permite controlar la temperatura en el $MPSoC$ seleccionando una apropiada secuencia de tareas para su ejecución.

1.3. Objetivos

Los objetivos de esta tesis son los siguientes:

Obtener una metodología de modelado térmico para un $MPSoC$ usando redes de Petri Continuas Temporizadas ($TCPN$). Validación del modelo obtenido.

- Obtener una metodología de modelado para tareas periódicas e independientes asignadas a un $MPSoC$ usando redes de Petri Continuas Temporizadas ($TCPN$).
- Obtener un modelo global en redes de Petri Continuas Temporizadas ($TCPN$) para diseño de schedulers térmicos.
- Propuestas de schedulers térmicos usando los modelos propuestos.

1.4. Estructura de la tesis

El documento está organizado de la siguiente manera.

En el Capítulo 2 se introducen los preliminares básicos de las redes de Petri como una herramienta eficiente para el modelado. Se abordan conceptos de transferencia de calor, se muestra un panorama general de sistemas de tiempo real y finalmente se presenta una breve descripción del problema de scheduling.

El Capítulo 3 presenta una metodología de modelado para un sistema de multiprocesadores en chip ($MPSoC$) usando Redes de Petri continuas Temporizadas ($TCPN$). Se muestran ejemplos del uso de la metodología y finalmente se valida el modelo mediante un software de elementos finitos ($ANSYS$).

El Capítulo 4 propone un modelo en redes de Petri para la asignación y ejecución de tareas periódicas e independientes. Se hace la integración de los modelos de asignación, ejecución con el modelo térmico del Capítulo 3 para obtener un sistema que capture el comportamiento térmico causado por la ejecución de tareas en un $MPSoC$.

El Capítulo 5 muestra la utilidad del modelo global derivado mediante $TCPN$, para ello se propone un scheduler MPC y un scheduler continuo. Para ilustrar el funcionamiento de los schedulers propuestos se presentan algunos ejemplos.

Capítulo 2

Preliminares

En este capítulo se presentan conceptos básicos de redes de Petri (PN), redes de Petri temporizadas (TPN), redes de Petri continuas (ContPN) y redes de Petri continuas temporizadas (TCPN) como herramienta alternativa para modelar el comportamiento y estructura de sistemas. Para una explicación mas detallada de las (PN) consulte [19], [20]

2.1. Redes de Petri (PN)

Las redes de Petri (PN) son consideradas como una herramienta muy efectiva para el estudio de sistemas con características como concurrencia, distribución y paralelismo de eventos. Las PN´s fueron diseñadas principalmente para modelar procesos de manufactura, sistemas de tráfico, sistemas logísticos, etc., debido a la capacidad que tienen de representar el fenómeno de concurrencia y sincronizan en estos sistemas. Una ventaja importante de las PN´s es su representación gráfica, el cual puede ser analizado de manera formal mediante una ecuación de estados obteniendo con ello información del comportamiento dinámico del sistema.

Una red de Petri se representa gráficamente con un grafo bipartita orientado, formado por lugares, transiciones y arcos. Los lugares y transiciones están representados como circunferencias y por segmentos rectilíneos respectivamente. Los arcos (representados por flechas) conectan los lugares y las transiciones. A continuación se presentan las definiciones básicas para redes de Petri.

2.1.1. Definiciones Básicas de Redes de Petri

Definición 2.1 ***Redes, Conjuntos*** $\bullet x$ ***y*** $x\bullet$ ***y Sub-redes***

Una Red de Petri N es una 3-tupla (P,T,F) donde P y T son conjuntos finitos y disjuntos y F es una función definida en $P \cup T$, $F \subseteq P \times T \cup T \times P$, tal que $F \cap (P \times P) = F \cap (T \times T) = \emptyset$.

*Los elementos en P son llamados **lugares** y gráficamente se representan por circunferencias. Los elementos T son llamados **transiciones**, representados por segmentos rectilíneos. El conjunto F es llamada la **relación de flujo** de la red, representada por **arcos** (flechas) que van de los lugares a las transiciones o de las transiciones a los lugares. De manera general se suele llamar a los elementos en $P \cup T$ **nodos** de N o elementos de N.*

Dado un nodo de la red N, el conjunto $\bullet x = \{y | (y,x) \in F\}$ es el pre-conjunto de x y el conjunto $x\bullet = \{y | (x,y) \in F\}$ es el post-conjunto de x. Los elementos en el conjunto $\bullet x (x\bullet)$ de un lugar x son las transiciones de entrada (salida). De manera similar, los elementos en $\bullet x (x\bullet)$ son los lugares de entrada (salida) de la transición x. Dado un conjunto X de nodos se define $\bullet X = \{\cup_{x \in X} (\bullet x)\}$ y $X\bullet = \{\cup_{x \in X} (x\bullet)\}$.

Una tripleta (P', T', F') es llamada sub-red de N si $P' \subseteq P$, $T' \subseteq T$ y $F' = F \cap ((P' \times T') \cup (T' \times P'))$. Si X es un conjunto de elementos de N, entonces la tripleta $(P \cap X, T \cap X, F \cap (X \times X))$ es una sub-red generada por X.

Definición 2.2 ***Caminos, Circuitos***

*Un **camino** en una red (P,T,F) es una secuencia no vacía $x_1, x_2, \ldots, x_k$ de nodos que satisfacen $(x_1,x_2), \ldots, (x_{k-1}, x_k) \in F'$ Un camino $x_1, x_2, \ldots, x_k$ se dice que va de x_1 a x_k.*

Un camino que va de un nodo x a un nodo y es un **circuito** sin ninguno de sus elementos se repite y ademas $(y,x) \in F$. Note que una secuencia conteniendo solo un elemento es un camino, pero no un circuito, porque para cada nodo x, $(x,x) \notin F$.

Una red (P,T,F) es llamada **conexa** si para cualesquiera dos nodos x, y satisfacen $(x,y) \in (F \cup F^{-1})^*$ Donde para cualquier conjunto Q, Q^* es la cerradura transitiva de Q. Una red (P,T,F) es **fuertemente conexa** si $\forall x, y \in P \cup T, (x,y) \in F^*$, es decir, para cualesquiera dos nodos existe un camino que lleva de x a y.

Ejemplo 2.1 Sea N una red de Petri representada gráficamente por la Fig. 2.1, donde los conjuntos de lugares y transiciones son:

$$P = \{p_1, p_2, p_3, p_4, p_5\} \tag{2.1}$$

$$T = \{t_1, t_2, t_3, t_4, t_5\} \tag{2.2}$$

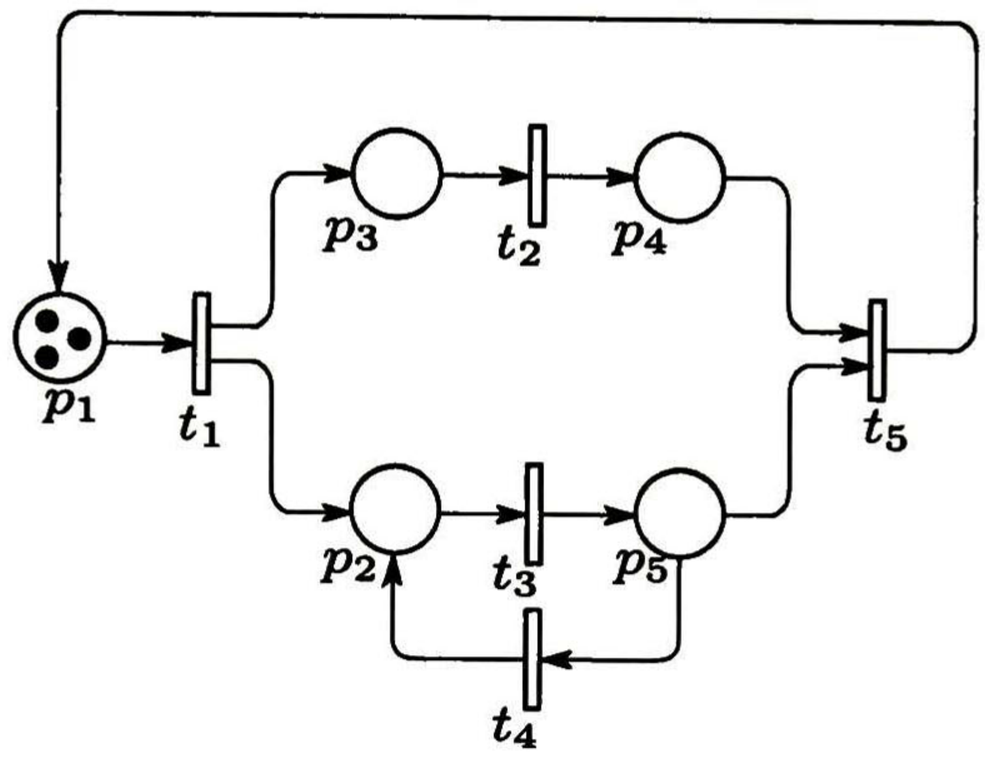


Figura 2.1: Ejemplo de una red de Petri

La relación de flujo para esta red es:

$$F = \left\{ \begin{array}{l} (p_1,t_1),(p_3,t_2),(p_2,t_3),(p_4,t_5),(p_5,t_5),(p_5,t_4), \\ (t_1,p_3),(t_1,p_2),(t_2,p_4),(t_3,p_5),(t_5,p_1),(t_4,p_2) \end{array} \right\} \tag{2.3}$$

Algunos conjuntos *Pre* y *Post* son:

$$\begin{array}{l} \bullet t_5 = \{p_4,p_5\}\ t_5\bullet = \{p_1\} \\ \bullet p_2 = \{t_1,t_4\}\ p_2\bullet = \{t_3\} \end{array} \tag{2.4}$$

Para esta red, se observa que $p_1t_1p_2t_3p_5$ es un camino y $p_1t_1p_3t_2p_4t_5$ es un circuito.

Definición 2.3 ***Marcados***

El ***marcado*** *de una red* (P,T,F) *es un mapeo* $m : P \to \{\mathbb{N} \cup 0\}$. *El marcado es representado por un vector* $[m(p_1), m(p_2), \ldots, m(p_n)]^T$, *donde* $p_1, p_2, \ldots, p_n$ *es una numeración fija y arbitraria de* P. *El marcado de una red de Petri esta representado por* $m(p)$ *marcas o el numero* $m(p)$ *en el lugar* p. *Un lugar* p *esta marcado con* m *marcas si* $m(p) > 0$. *Un conjunto de lugares* P_R *se dice que esta marcado si alguno de los lugares de* P_R *esta marcado.*

Gráficamente, las marcas se representan por puntos negros. De la Fig. 2.1 el marcado se puede representar como un vector $m_0 = [3\ 0\ 0\ 0\ 0]^T$. el cual indica el numero de marcas en cada lugar.

Definición 2.4 ***Peso de los arcos***

El ***peso de los arcos*** *es una función* $w : F \to \mathbb{N}$*, el cual asocia un numero natural a cada arco.*

El peso del arco equivale a la cantidad de arcos que existe entre un nodo a otro. Gráficamente, los pesos se escriben cerca de los arcos. Cuando todos los arcos tienen peso igual a 1, la red se dice que es ordinaria y el peso no se escribe.

Definición 2.5 ***Regla de disparo***

Un marcado m ***habilita*** *una transicion* t *si para cada lugar* $p \in \bullet t$*,* $m(p) \geq w(p,t)$*. Si* t *está habilitada en el marcado* m' *(escrito como* $m \to^t m'$*) que es definido por cada* p *por:*

$$m'(p) = \begin{cases} m(p) & \text{si } p \notin \bullet t \text{ y } p \notin t\bullet \\ m(p) - w(p,t) & \text{si } p \in \bullet t \text{ y } p \notin t\bullet \\ m(p) + w(t,p) & \text{si } p \notin \bullet t \text{ y } p \in t\bullet \\ m(p) - w(p,t) + w(t,p) & \text{si } p \in \bullet t \text{ y } p \in t\bullet \end{cases} \tag{2.5}$$

$w(p_i,t)$ *marcas se extraen de cada lugar* $p_i \in \bullet t$ *y se añaden* $w(t,p_j)$ *marcas a los lugares* $p_j \in t\bullet$*. Un marcado se dice muerto si no habilita alguna transición.*

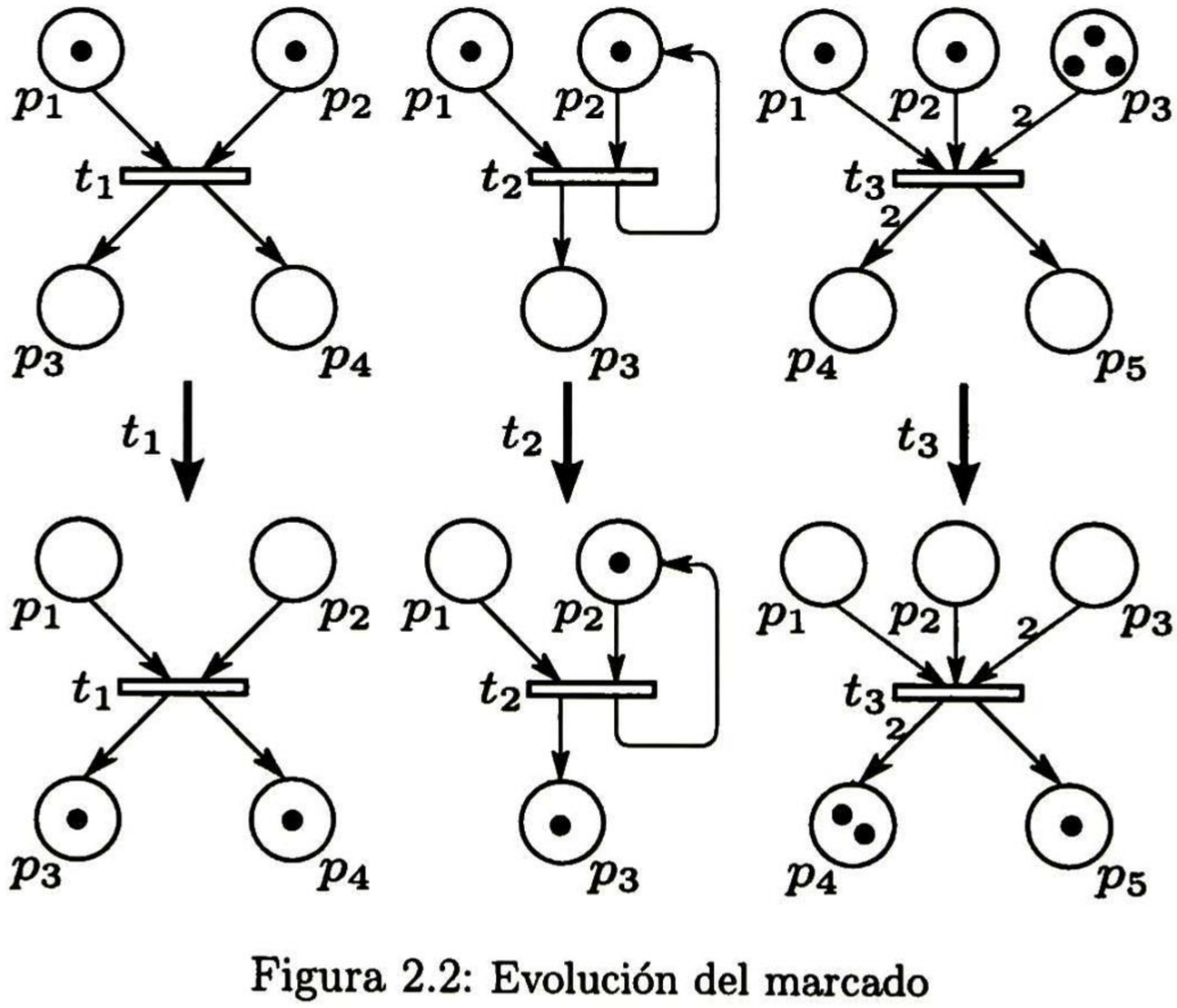


Figura 2.2: Evolución del marcado

Ejemplo 2.2 En la Fig. 2.2 se muestra gráficamente la evolución del marcado de tres diferentes Redes de Petri. El marcado que existe en las tres redes de Petri habilitan las transiciones t_1, t_2 y t_3 para su disparo.

Definición 2.6 ***Secuencias de disparo, marcados alcanzables***

Sea m un marcado de la red N. Si $m \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} m_n$ son disparos de las transiciones habilitadas, entonces $\sigma = t_1 t_2 \ldots t_n$ es una ***secuencia de disparos*** *que va de m a m_n y se escribe $m \xrightarrow{\sigma} m_n$. En la misma forma se puede representar la secuencia vacia ϵ, es decir, $m \xrightarrow{\epsilon} m$ para cualquier marcado m.*

Se escribe $m \xrightarrow{*} m'$, cuando m' es **alcanzable** desde m, es decir, que existe una secuencia de disparos σ tal que $m \xrightarrow{\sigma} m'$. El conjunto de todos los marcados alcanzables desde m es denotado pos $RS(m)$.

Si $m \xrightarrow{t_1} m_1 \xrightarrow{t_2} m_2 \xrightarrow{t_3} \ldots$ para una secuencia infinita de transiciones $\sigma = t_1 t_2 t_3 \ldots$ entonces σ es una secuencia infinita de disparos y se escribe como $m \xrightarrow{\sigma}$.

Definición 2.7 ***Matrices Pre, Post y de Incidencia***

Sea N la red (P, T, F). La matriz Post, denotada por C^+ de orden $|P| \times |T|$ es definida por:

$$Post(p,t) = C^+ \begin{cases} 0 & \text{si } (p,t) \notin F \\ w(p,t) & \text{si } (p;t) \in F \end{cases} \tag{2.6}$$

La matriz Pre, denotada por C^- de orden $|P| \times |T|$ es definida por:

$$Pre(p,t) = C^- \begin{cases} 0 & \text{si } (t,p) \notin F \\ w(p,t) & \text{si } (t,p) \in F \end{cases} \tag{2.7}$$

La **matriz de incidencia** que se denota por C esta definida como la diferencia entre las matrices *Pre* y *Post*, como sigue:

$$C = Pre - Post = C^+ - C^- \tag{2.8}$$

De manera similar a representaciones vectoriales de cualquier mapeo, la representación matricial de la matriz de incidencia depende de la enumeración de lugares y transiciones.

Definición 2.8 *El vector columna asociado a la transición t se denota como $\mathbf{t}$. De la misma manera, el vector fila que está asociado al lugar p se denota por $\mathbf{p}$.*

La entrada $C(p,t)$ en la matriz de incidencia representa el cambio de marcado del lugar p ocasionado por el disparo de la transición t. Entonces, si la transición t está habilitada en el marcado m y $m \xrightarrow{t} m'$ entonces $m' = m + \mathbf{t}$. Para una generalización de esta ecuación a secuencias de transiciones, se necesita la siguiente definición.

Definición 2.9 ***Vectores de Parikh de secuencias de transiciones.***

Sea (P,T,F) una red y sea σ una secuencia de transiciones. ***El vector de Parikh*** $\overrightarrow{\sigma} : T \to \{\mathbb{N} \cup \not\vdash\}$ *de σ mapea cada transición $t \in T$ al número de ocurrencias de t en σ. Si $T = \{t_1, t_2, t_3, t_4\}$, el vector de Parikh de una secuencia $\sigma_1 = t_1 t_3 t_1 t_1 t_2$ es*

$$\overrightarrow{\sigma_1} = [3\ 1\ 1\ 0]^T \tag{2.9}$$

y el de $\sigma_2 = t_4$ es

$$\overrightarrow{\sigma_2} = [0\ 0\ 0\ 1]^T \tag{2.10}$$

Para cualquier transición t

$$\mathbf{t} = C\overrightarrow{t} \tag{2.11}$$

Así ocurre que si $m \xrightarrow{t} m'$, entonces $m' = m + C\overrightarrow{t} = m + \mathbf{t}$ (donde m y m' son vectores columna). Para una secuencia de disparos finita cualquiera σ tal que $m \xrightarrow{\sigma} m'$ se puede encontrar el marcado m' mediante una ecuación de marcados descrito en el siguiente lema.

Lema 2.1 ***Lema de ecuación de marcado***

Para cualquier secuencia finita de disparos $m \xrightarrow{\sigma} m'$ de una red N, la siguiente ecuación de marcado se cumple:

$$m' = m + C\overrightarrow{\sigma} \tag{2.12}$$

Por inducción sobre la longitud de σ.

Sea $\sigma = \epsilon$. Por lo tanto $\overrightarrow{\sigma}$ y se cumple que $m = m'$.

Ahora, supongamos que σ es una secuencia no vacía. Entonces αt para una secuencia α y una transición t.

Sea $m \xrightarrow{\alpha} m_\alpha \xrightarrow{t} m'$. Se tiene que

$$\begin{aligned} m' &= m_\alpha + C\vec{t} \\ &= m + C\vec{\alpha} + C\vec{t} \\ &= m + C\vec{\alpha t} \\ &= m + C\vec{\sigma} \end{aligned} \tag{2.13}$$

■

2.1.2. Propiedades estructurales de Redes de Petri

Definición 2.10 ***Sistemas de Red. Marcado inicial y marcados alcanzables***

*Un **sistema** de red (o sólo un sistema) es un para (N, m_0) donde N es una red conexa que tiene al menos un lugar y una transición y m_0 es un marcado de N, llamado **marcado inicial**. Un **marcado** m' se dice **alcanzable** en un sistema con m_0 sis existe una secuencia de transiciones σ tal que $m \xrightarrow{\sigma} m'$.*

Ejemplo 2.3 De la Fig. 2.1 las matrices que representan al sistema son:

$$Post = C^+ = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad Pre = C^- = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & -1 \end{bmatrix}$$

Observe que la única transición habilitada es t_1, por lo que sólo esta transición se puede disparar. Para determinar el marcado después del disparo de la transición t_1 se utiliza el lema 2.1 donde $\vec{\sigma} = [1\ 0\ 0\ 0\ 0]^T$ representa la transición t_1 y $m_0 = [3\ 0\ 0\ 0\ 0]^T$ es el marcado inicial.

$$m' = m_0 + C\vec{\sigma}$$

$$m' = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Definición 2.11 ***Vivacidad y propiedades relacionadas***

Un sistema es ***vivo*** *si para cada marcado alcanzable y cada transición $t \in T$, existe un marcado $m' \in RS(N,m)$ que habilita la transición t. Si (N,m_0) es un sistema vivo, entonces se dice que m_0 es un marcado vivo de N.*

Un sistema es vivo en lugares si para cada marcado alcanzable m y cada lugar p existe un marcado $m' \in RS(N,m)$ que marca al lugar p.

Un sistema es ***libre de bloqueo*** *si cada marcado alcanzable habilita al menos una transición, es decir, si no se puede alcanzar ningún marcado muerto (que no habilita transiciones) desde el marcado inicial.*

Hablando en términos generales, un sistema es vivo si toda transición siempre puede volver a ocurrir en algún momento.

Definición 2.12 ***Sistemas acotados. Cotas de un lugar***

Un sistema es acotado si para cualuier lugar p existe un número natural b tal que $m(p) \leq b$ para cualquier marcado alcanzable. Si $N(N,m_0)$ es un sistema acotado, entonces se dice que m_0 es un marcado acotado para N.

La cota de un lugar p en un sistema acotado (N,m_0) se define como:

$$max\{m(p)|m \in RS(N,m_0)\}. \tag{2.14}$$

Un sistema es llamado $b-$ Acotado si ningún lugar tiene una cota que sea mayor a b.

Definición 2.13 ***P-invariantes (P-Semiflujos)***

Un ***P-invariante*** *de una red N es una solución racional de la ecuación:*

$$Y^T \cdot C = 0 \tag{2.15}$$

Proposición 2.1 ***Propiedad fundamental de los P-invariantes***

Sea (N,m_0) un sistema y sea I un P-invariante de N. Si $m_0 \overrightarrow{*} m'$, entonces

$$I^T \cdot m = I^T \cdot m' \tag{2.16}$$

La demostración se encuentra en [19]

Esta propiedad fundamental denota que existen conjuntos de lugares en una red de Petri en donde la suma de sus marcas permanece constante ante el disparo de transiciones.

Definición 2.14 ***T-invariantes (T-Semiflujos)***

Un ***T-Semiflujos*** *de una red N es una solución racional de la ecuación:*

$$C \quad X = 0 \tag{2.17}$$

Proposición 2.2 ***Propiedad fundamental de los T-invariantes***

Sea σ una secuencia finita de transiciones de una red N que está habilitada en el marcado m. Entonces el vector de parikh $\vec{\sigma}$ es un ***T-invariante*** *si y sólo si $m_0 \xrightarrow{\sigma} m$ (es decir, si solo si la ocurrencia de σ reproduce el marcado m).*

Los P-sistemas (Maquinas de Estados) son aquellos en los que todas sus transiciones tienen exactamente un lugar de entrada y un lugar de salida.

Definición 2.15 ***P-sistemas (Maquinas de esttados), P-Redes***

Una red es $P-red$ si $|\bullet t| = 1 = |t \bullet| \forall t \in T$. Un sistema (N, m_0) es un P-sistema si N es una P-red.

La propiedad fundamental de las Maquinas de Estado es que todos los marcados alcanzables contienen exactamente el mismo número de marcas. Dicho de otra forma, el número de marcas del sistema permanece invariante ante el disparo de transiciones. Lo anterior, significa que todos los lugares de una máquina de estados formas un P-semiflujo. Los grafos Marcados o T-sistemas son aquellos en los que todos los lugares tienen exactamente una transición de entrada y una salida.

Definición 2.16 ***T-Sistemas (Grafos Marcados), T-redes***

Una red es $T-red$ si $|\bullet p| = 1 = |p \bullet| \forall p \in P$ Un sistema (N, m_0) es un $T-Sistema$ si N es una $T-Red$.

La propiedad fundamental de los grafos marcados es que la cantidad de marcas de los circuitos permanece invariante ante el disparo de transiciones.

Otro tipo de redes muy estudiado son las de Libre Elección (Free-Choice)

Definición 2.17 ***Redes de Libre Elección (Free-Choice)***

Una red (P, T, F) es de Libre Elección si $(p, t) \in F$ implica $\bullet t \times p \bullet \subseteq F$ para cada lugar p y cada transición t.

Un sistema (N, m_0) es de libre elección si N es de libre elección.

La propiedad fundamental de las redes de libre elección es que si un marcado habilita alguna transición de $p\bullet$, donde p es un lugar de la red, entonces habilita a todas las transiciones en $p\bullet$.

Definición 2.18 ***Sifones y Sifones propios***

Un conjunto R de lugares de una red son un ***sifón*** *si $\circ R \subseteq R\bullet$. Un sifón es llamado* ***Sifón propio*** *si no es el conjunto vacío.*

Dos importantes hechos de los sifones son que: Sifones que no tienen marcas permanecen sin marcas y que los sistemas vivos no tienen sifones sin marcas.

Definición 2.19 ***Trampas y Trampas propias***

Un conjunto R de lugares de una red son una ***trampa*** *si $R\bullet \subseteq \bullet R$. Una trampa es llamada* ***trampa propia*** *si éste no es el conjunto vacío.*

2.2. Redes de Petri Temporizadas

Las Redes de Petri Temporizadas (TPN, por sus siglas en inglés) evolucionan de forma similar a las Redes de Petri, pero asignan un tiempo de retardo entre el disparo de una transición y el cambio de marcado. Existen dos modelos importantes, uno de ellos temporizado en lugares [19], pero se refiere el modelo temporizado en transiciones por ser mas natural, puesto que la representación de actividades de un sistema es através de las transiciones [21]. A continuación se presenta una breve descripción de estas.

Definición 2.20 ***Redes de Petri Temporizadas en Transiciones [22]***

Una ***Red de Petri Temporizada en transiciones*** *se define como un par (N, D) donde:*

N es una Red de Petri. Sea $D : T \rightarrow (R^+ \cup 0)^{|T|}$ una función de retardos que asigna el tiempo que una transición t_i toma para realizar su disparo.

En una Red de Petri Temporizada en Transiciones, las marcas pueden tener cualesquiera de los dos atributos: **disponible** y **no disponible**. En el marcado inicial, todas las marcas se encuentran disponibles y pueden cambiar su estado cuando una transición t_i está habilitada.

Una transición t_j se encuentra habilitada cuando $m_{disp}(p_i) \geq Pre(p_i, t_j) \forall p_i \in \bullet t_j$ y $m_{disp}(p_i)$ son las marcas disponibles en el lugar p_i.

Cuando una transición t_j es disparada, $Pre(p_i, t_j)$ marcas cambian su atributo a no *disponibles*, de modo que no son removidas del lugar p_i (pero permanecen congeladas y no pueden habilitar a ninguna otra transición) hasta $d_j \in D$ unidades de tiempo después, cuando se colocan $Post(p_k, t_j)$ marcas etiquetadas como *disponibles* en los lugares $p_k \in t_j\bullet$.

De hecho, la evolución de marcado de las TPN también cumple con la ecuación (2.1), este tipo de redes son ampliamente utilizadas para la programación de tareas (en inglés scheduling.), puesto que se puede minimizar el tiempo para llegar de un marcado m a otro m' cuando existen caminos paralelos [22].

2.3. Redes de Petri Continuas

Las Redes de Petri Continuas (CPN, por sus siglas en inglés) [23] surgen a partir de un proceso de fluidificación, en el que cada una de las marcas es dividida en k partes y en el proceso al límite, cuando $k \to \infty$ se pueden hacer disparos de partes reales de marcas.

Las CPN surgen como una opción para disminuir en lo posible uno de los principales problemas de los sistemas de eventos discretos (SED): la explosión de estados [24].

Una Red de Petri está limitada a disparos de marcas completas, es decir, la cantidad de disparo está limitada a números naturales, mientras que en el caso de las CPN, se puede disparar cualquier cantidad real de marcas.

Existen trabajos de aplicación de las CPN, como el que se muestra en [25] y otros referentes a propiedades de éstas, como se presentan en [26, 27, 28].

Definición 2.21 *Una **Red de Petri Continua** es el par $\langle N, m_0 \rangle$ donde $N = \langle P, T, Pre, Post \rangle$ es una Red de Petri y $m : P \to \mathbb{R}_{\geq 0}^{|P|}$ asigna un número real de marcas a cada lugar. El marcado inicial es $m_0 \in \mathbb{R}_{\geq 0}^{|P|}$.*

A diferencia del caso discreto, en una CPN una transición t_i se encuentra habilitada en un marcado m si y sólo si $\forall p \in \bullet t_i,\ m(p) > 0$.

Definición 2.22 ***Grado de habilitación de una transición***

*El **grado de habilitación** de una transición t se define como:*

$$enab(t, m) = \min_{p \in \bullet t} \left\{ \frac{m(p)}{Pre[p, t]} \right\} \tag{2.18}$$

La transición t puede dispararse cualquier cantidad $\alpha \in R,\ 0 < \alpha < enab(t, m)$ y llevando el sistema a un nuevo marcado $m' = m + \alpha C(P, t)$, donde C es la matriz de incidencia previamente mencionada.

Si un marcado m' puede ser alcanzado desde m_0 con una secuencia de disparos σ, la ecuación fundamental de marcado puede ser reescrita como:

$$m = m_0 + C\sigma \tag{2.19}$$

donde $\sigma \in (\mathbb{R} \cup 0)^{|T|}$ es el vector de conteos de disparos.

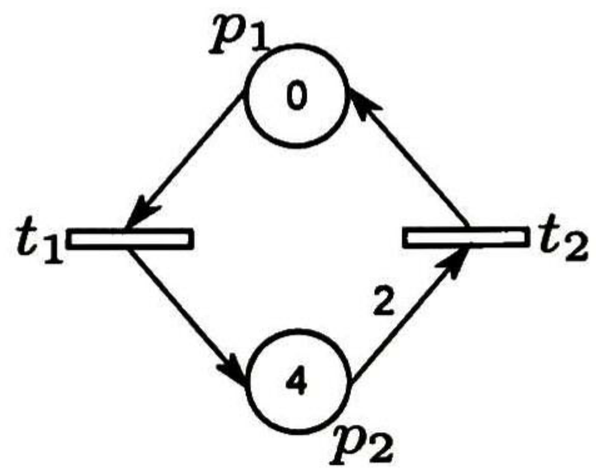


Figura 2.3: Ejemplo de una red de Petri Continua, sólo t_2 está habilitada.

Ejemplo 2.4 Considere la Red de Petri continua de la Fig. 2.3, el grado de habilitación de la transición t_2 es $enab(t_2, m_0) = 2$ y el grado de habilitación de t_1 es $enab(t_1, m_0) = 0$, por lo que t_1 no puede ser disparada. Entonces t_2 puede dispararse con una cantidad $\alpha_2 = 1.5 < enab(t_2, m_0)$, por lo que de la ecuación (2.19) el marcado alcanzable será $m = [1.5\ 1]^T$.

Definición 2.23 ***Acotamiento, vivacidad y lim-vivacidad para CPN.***

Una CPN N_c se dice que es:

Acotada *si todos sus lugares son acotados* $\forall p_i \in P,\ \exists b_i \in \mathbb{R} | b_i \geq m(p_i)\ \forall m(p) \in RS(N_c, m_0)$

Viva *si toda transición es viva (cualquier transición puede volver a ser disparada desde cualquier marcado alcanzable).*

La propiedad de vivacidad puede ser extendida a ***lim-vivacidad*** *suponiendo que una secuencia de longitud infinita puede ser disparada. Una transición t no es lim-viva si sólo si existe un marcado alcanzable tal que ninguno de sus sucesores habilite a la transición t.*

Definición 2.24 ***Acotamiento estructural y vivacidad estructural.***

Una red es:

Estructuralmente acotada *cuando* (N, m_0) *es acotada para cualquier marcado inicial* m_0.

Estructuralmente viva *cuando existe un marcado* m_0 *tal que la red* (N, m_0) *es viva.*

2.4. Redes de Petri Continuas Temporizadas(TCPN)

Las redes de Petric Continuas Temporizadas o redes fluidificadas (TCPN por sus siglas en inglés) son una relajación de las redes de Petri discretas que consiste en aproximar el comportamiento de una TPN por un conjunto de ecuaciones diferenciales lineales y continuas. Tal relajación es útil para tratar sistemas que presentan el problema de la explosión de estados, problema que en el contexto de las redes de Petri se presenta en sistemas altamente marcados.

Definición 2.25 *Una* ***Red de Petri Continua Temporizada*** *ó TCPN es un sistema definido por* (N, λ, m_0)*, donde* N *es una CPN, y* $\lambda : T \to \{\mathbb{R} \cup 0\}^{|P|}$ *es el marcado inicial de la red* N.

Cuando el tiempo es incluido en una red CPN la ecuación fundamental (2.19) depende explicitamente del tiempo:$m(\tau) = m_0 + C\sigma(\tau)$, es posible derivar con respecto al tiempo para obtener una ecuación dinámica: $\dot{m}(\tau) = C \cdot \dot{\sigma}(\tau)$. La derivada del vector de disparo $f(\tau) = \dot{\sigma}(\tau)$ es llamada **flujo**, y nos lleva a la ecuación de la dinámica de las TCPN temporizadas:

$$\dot{m}(\tau) = Cf(\tau) \tag{2.20}$$

El comportamiento del flujo de las transiciones está definido por alguna **semántica**. En estas semánticas se hace uso de un vector constante llamado velocidad de marcado $\lambda : T \to \mathbb{R}^+$. el cual asigna a cada transición t_i una constante positiva λ_i.

Definición 2.26 ***Semántica de servidores finitos (SF)***

Cada transición t *tiene asociado un valor real* $\lambda(t) > 0$ *el cual es la tasa máxima de flujo de la transición y es constante. Entonces el flujo queda definido como:*

$$f_j = \begin{cases} \lambda_j, \ \ if \ \forall p_i \in {\bullet t_j}, \ \ m_i > 0 \\ min \left\{ \underset{p_i \in {\bullet t_j} | m_i = 0}{\text{mín}} \left\{ \sum_{t_q \in {\bullet p_i}} \frac{f_q \cdot Post(t_q, p_i)}{Pre[p_i, t_j]} \right\}, \lambda_j \right\} \ \ de\ otra\ forma. \end{cases} \tag{2.21}$$

Para la semántica de **servidores finitos**, si el marcado de los lugares de entrada de la transición t_i es estrictamente mayor que cero, su flujo será constante, igual a λ_i, si no es estrictamente mayor, el flujo es el mínimo entre la velocidad máxima de disparo y el flujo total de entrada a los lugares vacíos. Esta semántica corresponde a la de *velocidad constante* de [23].

Definición 2.27 ***Semántica de servidores infinitos (SI)***

Cada transición t tiene asociado un valor real $\lambda > 0$ el cual es la tasa máxima de flujo de la transición y es constante. En esta semántica, la velocidad de ocurrencia es sensible al grado de habilitación de la transición, entonces el flujo queda definido:

$$f_i = \lambda_i \cdot enab(t_i, \boldsymbol{m}) = \lambda_i \cdot \min_{p \in \bullet t_i} \left\{ \frac{\boldsymbol{m}(p)}{\boldsymbol{Pre}[p, t_i]} \right\} \tag{2.22}$$

donde f_i es el flujo de la transición t_i y $p \in \bullet t_i$ es el lugar cuyo marcado $m(p)$ limita a t_i. Llamamos flujo efectivo a $f(m)$, el vector de flujos evaluado al marcado m.

Para la semántica de **servidores infinitos** el flujo a través de una transición es proporcional a su grado de habilitación.

Definición 2.28 ***semántica de producto*** *La **semántica de producto** el flujo de una función está dado por el producto de los marcados de sus lugares de entrada, definido como:*

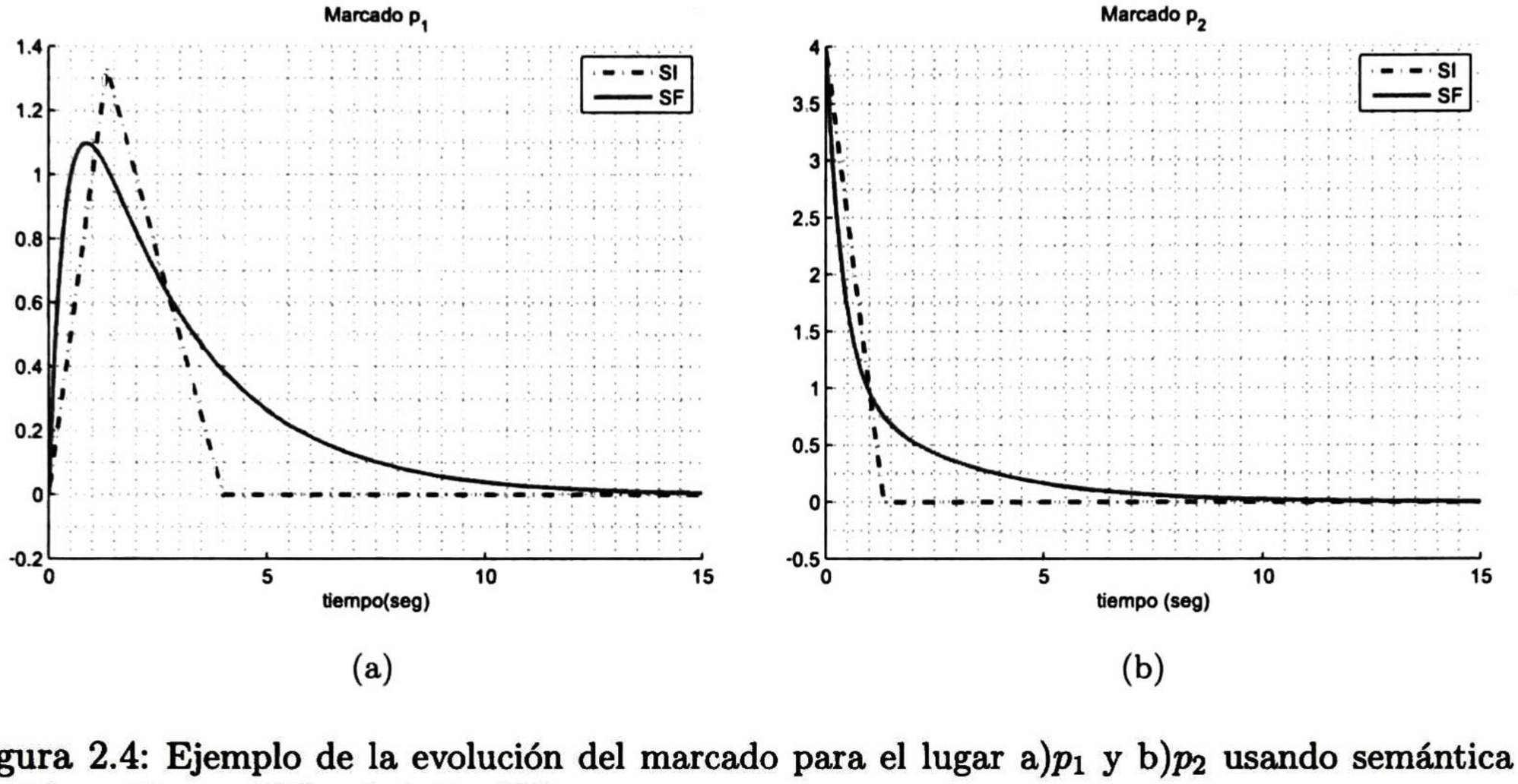
$$f_i = \lambda_i \cdot \prod_{p \in \bullet t_i} \left\{ \frac{\boldsymbol{m}(p)}{\boldsymbol{Pre}[p, t_i]} \right\} \tag{2.23}$$

Esta semántica puede ser obtenida con la semántica de servidores infinitos a partir de la decoloración de redes de Petri coloreadas [29].

Interpretando las transiciones como punto de encuentro de clientes y servidores, parece que la semántica más adecuada depende del número relativo de clientes y servidores que haya en el modelo discreto.

Ejemplo 2.5 Sea el sistema de la Fig. 2.3 una CPN con tasas máximas de flujo $\lambda(t_1) = 1$ y $\lambda(t_2) = 2$. En las Fig. 2.4(a) y Fig. 2.4(b) se muestra el comportamiento del lugar p_1 y p_2 respectivamente para las semánticas de servidores finitos e infinitos. Para analizar la evolución de la red bajo las semánticas definidas anteriormente se utiliza el simulador propuesto en [28].

Las transiciones fuente (transiciones sin lugares de entrada) no están definidas en estas semánticas puesto que ningún lugar restringe su flujo. Una manera de definir transiciones con flujo constante es modelar una transición t_i junto con un lugar p_j tal que $\bullet t_i = t_i \bullet = p_j$ y $\mathbf{Pre}(p_j, t_i) = \mathbf{Post}(p_j, t_i) = 1$ como se muestra en la Fig. **??**.



(a) (b)

Figura 2.4: Ejemplo de la evolución del marcado para el lugar a)p_1 y b)p_2 usando semántica de servidores Finitos(SF) e Infinitos(SI)

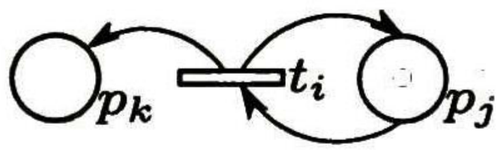


Figura 2.5: Ejemplo de una transición fuente bien definida.

Definición 2.29 *En una TCPN una transición $t_i \in T$ es llamada **bien definida** si $|\bullet t_i| \geq 1$, i.e. si tiene almenos un lugar de entrada. Una TCPN es llamada **bien definida** si $\forall t_i \in T$, t_i es bien definida.*

En este trabajo las TCPN son consideradas bien definidas.

Definición 2.30 ***Restricción en el flujo que pasa por una transición***

*Sea $\Sigma = (N_c, \lambda, m_0)$ una TCPN y sea m un marcado alcanzable. Se dice que el arco (p,t) **restringe el flujo que pasa por una transición t** en el marcado m ssi*

$$f(t) = \lambda(t)\frac{m(p)}{Pre(p,t)} \tag{2.24}$$

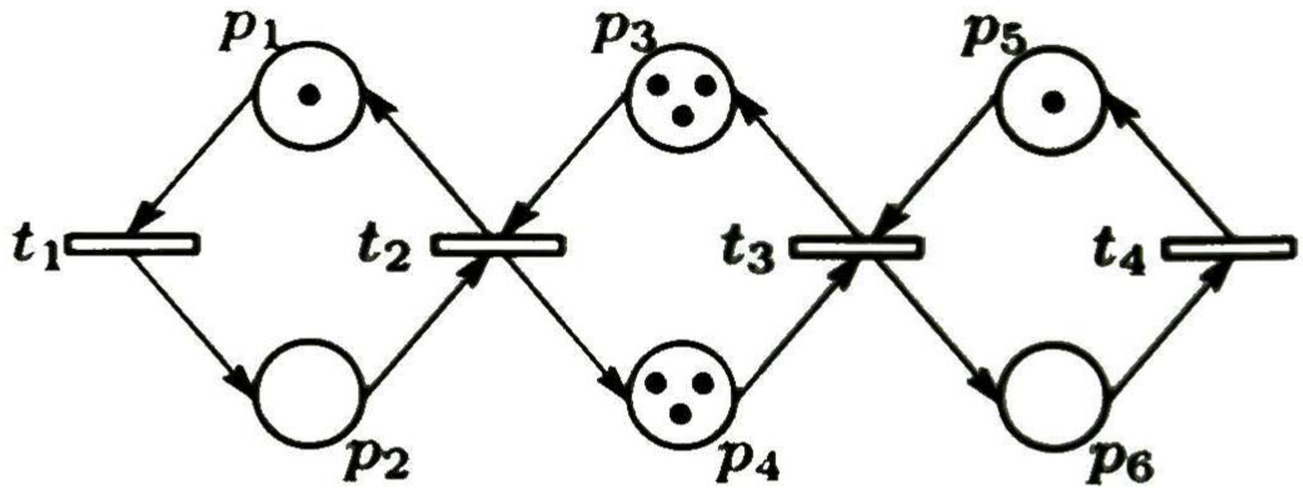


Figura 2.6: Ejemplo de una red de Petri Continua Temporizada

Ejemplo 2.6 Considere la Fig. 2.6 una red TCPN. Haciendo uso de la ecuación de flujos (2.22) tenemos:

$$\begin{aligned} f_1 &= \lambda(t_1) \cdot m(p_1) \\ f_2 &= \lambda(t_2) \cdot min\{m(p_2), m(p_3)\} \\ f_3 &= \lambda(t_3) \cdot min\{m(p_4), m(p_5)\} \\ f_4 &= \lambda(t_4) \cdot m(p_6) \end{aligned} \tag{2.25}$$

Si el vector de tasa de disparo es $\lambda = [1\ 1\ 1\ 1]^T$, entonces de la ecuación (2.20) tenemos:

$$\dot{m} = C \cdot f$$

$$\dot{m} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \lambda_1 m_1 \\ \lambda_2 min\{m_2, m_3\} \\ \lambda_3 min\{m_4, m_5\} \\ \lambda_4 m_6 \end{bmatrix}$$

$$\dot{m} = \begin{bmatrix} \dot{m}_1 \\ \dot{m}_2 \\ \dot{m}_3 \\ \dot{m}_4 \\ \dot{m}_5 \\ \dot{m}_6 \end{bmatrix} = \begin{bmatrix} f_2 - f_1 \\ f_1 - f_2 \\ f_3 - f_2 \\ f_2 - f_3 \\ f_4 - f_3 \\ f_3 - f_4 \end{bmatrix} = \begin{bmatrix} min\{m_2, m3\} - m_1 \\ m_1 - min\{m_2, m_3\} \\ min\{m_4, m_5\} - min\{m_2, m_3\} \\ min\{m_2, m_3\} - min\{m_4, m_5\} \\ m_6 - min\{m_4, m_5\} \\ min\{m_4, m_5\} - m_6 \end{bmatrix}$$

Una *TCPN* bajo la semántica de servidores infinitos es un sistema lineal conmutado, dado por el operador *min* que aparece en la ecuación de flujo (2.22). A lo largo de este trabajo se utilizará la semántica de servidores infinitos para el modelado de sistemas que presentan la característica de explosión de estados.

Definición 2.31 ***Configuración***

Sea $\Sigma = (N_c, \lambda, m_0)$ una TCPN. Una ***configuración*** *de Σ en el marcado m es un conjunto de (p,t) arcos que describen el flujo real que pasa por todas las transiciones.*

El número total de configuraciones en un sistema está dado por:

$$NumConfig = \prod_{i=1}^{|T|} |\bullet t_i| \tag{2.26}$$

Entonces, una configuración es una cobertura de T por sus arcos de entrada. Una posible representación de una configuración dada es a través de una matriz de configuración $\prod$. La construcción de esta matriz se explica a continuación.

Definición 2.32 ***Vector característico de Restricción del flujo que pasa por una transición.***

Sea el arco (p_i, t_j) el que restringe el flujo que pasa por la transición t_j en el marcado m, entonces el ***vector característico de restricción de flujo que pasa por la transición*** *$\prod_j(m)$ es un vector con sus elementos igual a cero y un uno en la posición i.*

$$\pi_j(m) = \begin{bmatrix} 0 \\ \dots \\ 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} \begin{matrix} \} \\ \\ \\ \\ \\ \\ \} \end{matrix} \begin{matrix} 1 \\ \dots \\ i-1 \\ i \\ i+1 \\ \dots \\ n \end{matrix} \tag{2.27}$$

Es claro que $\pi_j(m) \in \{0,1\}^{|P|}$ y que depende del marcado. En caso de que existan dos arcos (p_i, t) y (p_j, t) cumplan con la restricción de flujo (2.30), cualesquiera de los dos vectores característicos puede ser utilizado para representar al arco en la configuración, pero siempre se deberá elegir sólo uno.

Definición 2.33 ***Matriz de configuración*** $\Pi_(m)$

Dado un marcado m, la ***matriz de configuración*** *se construye con el transpuesto de los vectores característicos de restricción en la dinámica de una transición.*

$$\pi_j(m) = \begin{bmatrix} \pi_1^T \\ \pi_2^T \\ \dots \\ \pi_T^T \end{bmatrix} \tag{2.28}$$

Ejemplo 2.7 En la red TCPN de la Fig. 2.6, se tienen 4 configuraciones, estás dependen de la relación que existe en los lugares de entrada de cada transición. Las matrices de configuración $\Pi(m)$ son:

$$\Pi(m) = \begin{cases} \Pi_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & si\ m_2 \leq m_3\ y\ m_4 \leq m_5 \\ \Pi_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & si\ m_3 \leq m_2\ y\ m_4 \leq m_5 \\ \Pi_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & si\ m_2 \leq m_3\ y\ m_5 \leq m_4 \\ \Pi_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} & si\ m_3 \leq m_2\ y\ m_5 \leq m_4 \end{cases}$$

Definición 2.34 ***Matriz de máxima tasa de disparo***

La matriz de máxima tasa de disparos está definida como:

$$\Lambda = diag(\lambda_1, \dots, \lambda_{|T|}) \tag{2.29}$$

De acuerdo a las definiciones anteriores y su notación podemos re definir el vector de flujo (2.30) y la ecuación fundamental (2.20) como:

$$f = \Lambda \cdot \Pi(m) \cdot m \tag{2.30}$$

$$\dot{m} = C \cdot \Lambda \ \ \Pi(m) \cdot m \tag{2.31}$$

2.5. Acción de Control en TCPN

De acuerdo a la semántica de los modelos, se observa que la única acción de control que se puede aplicar es **reducir** el flujo de disparo de transiciones [9]. Pensando por ejemplo en sistemas de producción, resulta bastante lógico pensar en frenar las máquinas, pero resulta imposible hacer que vayan más rápido que su velocidad máxima. Así, si el flujo de una transición t puede ser reducida o incluso detenida, se dirá que t es una transición controlable, de otra manera la transición es incontrolable. El flujo forzado de una transición controlable t_i es $f_i - u_i$, donde f_i es el flujo del sistema no forzado y u_i es la acción de control y verifica que $0 \leq u_i \leq f_i$. Observe que esto significa que el valor máximo de la acción de control es variable y depende del marcado. El vector de flujo controlado es:

$$\begin{aligned} &f = \Lambda\Pi(m) \cdot m - u \\ &0 \leq u_i \leq [\Lambda\Pi(m) \cdot m]_i. \end{aligned} \tag{2.32}$$

Con el fin de obtener una versión simplificada de la ecuación de estados, el vector de entradas u se reescribe como:

$$u = I_u \Lambda\Pi(m) \cdot m \tag{2.33}$$

donde $I_u = diag(I_{u_1}, \ldots, I_{u_{|T|}})$ y $0 \leq I_{u_i} \leq 1$. Sustituyendo (2.33) en (2.32) y definiendo la matriz $I_c = I - I_u$ la ecuación de estados (2.20) se puede reescribir como:

$$\begin{aligned} &\dot{m} = C I_c \Lambda\Pi(m) \cdot m \\ &0 \leq I_{ci} \leq 1 \end{aligned} \tag{2.34}$$

2.6. Termodinámica y Transferencia de Calor

En esta sección se detallaran algunos conceptos importantes en el ámbito de transferencia de calor. Para obtener información más detallada el lector puede consultar [2], [30].

La transferencia de calor es un fenómeno termodinámico que se lleva acabo por la transferencia de energía de un medio caliente hacia uno frío. La transferencia de energía siempre

se produce del medio que tiene mayor temperatura hacia el medio de temperatura más baja y esa transferencia se detiene cuando se alcanza un equilibrio térmico entre los medios actuantes.

El requisito básico para la transferencia de calor es la presencia de una diferencia de temperatura. No puede haber transferencia neta de calor entre dos medios que están a la misma temperatura. La velocidad de la transferencia de calor en cierta dirección depende de la magnitud del gradiente de temperatura (la diferencia de temperatura por unidad de longitud o la razón de cambio de temperatura en esa dirección). A mayor gradiente de temperatura, mayor es la razón de transferencia de calor.

Para comprender el funcionamiento de la transferencia de calor es necesario definir algunos conceptos importantes que involucra este fenómeno.

El calor y temperatura son conceptos diferentes. La temperatura se refiere a la medida del calor. En cambio el calor es una transferencia de energía entre dos cuerpos, producida por una diferencia de calor.

Definición 2.35 *El **calor** se define como la energía cinética total de todos los átomos o moléculas en una sustancia.*

Definición 2.36 *La **Temperatura** es una medida de la energía cinética promedio de los átomos y moléculas individuales de una sustancia.*

Cuando se agrega calor a una sustancia, sus átomos o moléculas se mueven más rápido y su temperatura se eleva, o viceversa.

Definición 2.37 ***calor específico*** *El calor específico se define como la energía requerida para elevar un grado la temperatura de una unidad de masa de una sustancia.*

Suele tenerse interés en dos tipos de calores específicos: el calor específico cuando el volumen permanece constante c_v y el calor específico cuando la presión permanece constante c_p. Una sustancia cuyo volumen específico no cambia con la temperatura o la presión se le conoce como **sustancia incomprensible** y para ellos los calores específicos a volumen constante y presión constante son idénticos. Es decir $c_p \cong c_v \cong c$. Los sólidos y líquidos mantienen constantes sus volúmenes por que son considerados como sustancias incomprensibles.

Definición 2.38 *La cantidad de **calor transferido** durante un proceso se denota como Q. La cantidad de calor transferido por unidad de tiempo se define como **razón de transferencia de calor** y se denota como $\dot{Q}$.*

La tasa de transferencia de calor de calor $\dot{Q}$ tiene como unidades J/s, lo cual es equivalente a W.

Definición 2.39 *La razón de calor por unidad de área perpendicular a la dirección de Q se llama **flujo de calor** y el flujo promedio de calor se expresa:*

$$\dot{q} = \frac{\dot{Q}}{A} \tag{2.35}$$

en donde A es el área de transferencia de calor.

Un análisis termodinámico se interesa en la cantidad de transferencia de calor que existe en un sistema cuando pasa de un estado de equilibrio a otro. Por lo tanto es necesario conocer los mecanismos de transferencia de calor.

2.6.1. Mecanismos de Transferencia de Calor

El calor puede transferirse en tres modos diferentes: **conducción, convección y radiación**. Estas tres formas de transferir calor de un cuerpo a otro requieren de la existencia de una diferencia de temperatura y todos ellos ocurren del medio de mayor temperatura al medio que posee una temperatura más baja.

Conducción

La conducción es el mecanismo de transferencia de calor en donde la energía se transfiere de las partículas más energéticas de una sustancia hacia las adyacentes menos energéticas, como resultado de la interacción entre partículas. La interacción produce un flujo de calor desde la sustancia de temperatura más alta hasta la de menor temperatura.

La rapidez de la conducción de calor a través de un medio depende de la geometría del mismo, el espesor y el material de que esté hecho así como la diferencia de temperatura que exista en él.

Definición 2.40 ***Ley de Fourier de la conducción de calor.***

Para un volumen de espesor Δx, con un área de sección transversal A y cuyas caras opuestas se encuentren a temperaturas T_1 y T_2, se encuentra que la razón de la conducción de calor a través de una capa plana es proporcional a la diferencia de temperatura $\Delta T = T_2 - T_1$ a través de ésta y el área de transferencia de calor, e inversamente proporcional al espesor de esa capa es decir:

$$\dot{Q}_{cond} = kA\frac{T_1 - T_2}{\Delta x} = -kA\frac{\Delta T}{\Delta x} \tag{2.36}$$

en donde la constante de proporcionalidad k es la ***conductividad térmica*** *del material. En el caso de que* $\Delta x \to 0$ *la ecuación anterior se reduce a su forma diferencial.*

$$\dot{Q}_{cond} = -kA\frac{dT}{dx} \tag{2.37}$$

la cual se hace llamar ***ley de Fourier de la conducción del calor****. Aquí* $\frac{dT}{dx}$ *es el gradiente de Temperatura.*

El signo menos indica que la conducción de calor es en la dirección decreciente de la temperatura.

Definición 2.41 ***Conductividad térmica*** *La conductividad térmica* $\mathbf{k}$ (W/mK) *es una medida de la capacidad de un material para conducir el calor. La conductividad térmica de un material se puede definir como la razón de transferencia de calor a través de un espesor unitario del material por unidad de área por unidad de diferencia de temperatura.*

Un elevado valor de la conductividad térmica en un material indica que el material es un buen conductor de calor y un valor bajo indica que es un mal conductor o que es un aislante.

Convección

La **convección** es el mecanismo de transferencia de calor entre una superficie sólida y líquido o gas adyacentes que están en movimiento. Puede ser producida de forma natural, por la diferencia de densidades debidas a las variación de temperatura en el fluido ó puede ser producida de manera forzada cuando el fluido es forzado a fluir sobre la superficie mediante medios externos.

La rapidez de la transferencia de calor por convección es proporcional a la diferencia de temperatura y se expresa en forma matemática por medio de la **ley del enfriamiento de newton**. La ley de enfriamiento de Newton enuncia que cuando la diferencia de temperaturas entre un cuerpo y su medio ambiente no es demasiado grande, el calor transferido por unidad de tiempo hacia el cuerpo o desde el cuerpo por conducción, convección y radiación, es aproximadamente proporcional a la diferencia de temperaturas entre el cuerpo y dicho medio externo, siempre y cuando este último mantenga constante su temperatura durante el proceso de enfriamiento.

Definición 2.42 ***Ley del enfriamiento de Newton.***

Esta ley describe que la razón de pérdida de calor de un cuerpo es proporcional a la diferencia entre la temperatura del cuerpo y el medio ambiente que lo circunda. Se expresa de la siguiente forma:

$$\dot{Q}_{conv} = hA_s(T_s - T_\infty) \tag{2.38}$$

donde h es el coeficiente de transferencia de calor por convección en (W/m^2K), A_s *es el área superficial a través de la cual tiene lugar la transferencia de calor por convección,* T_s *es la temperatura de la superficie y* T_∞ *es la temperatura del fluido.*

El coeficiente de transferencia de calor h no es una propiedad del fluido. Es un parámetro que se calcula de forma experimental y cuyo valor depende de todas las variables que influyen sobre la convección como es la geometría de la superficie, las propiedades del fluido y su naturaleza de movimiento.

Radiación

La radiación es la energía emitida por la materia en forma de ondas electromagnéticas. Se produce desde una fuente hacia todas las direcciones y es producida por los cambios en las configuraciones electrónicas de los átomos o moléculas. La radiación térmica es la forma de radiación emitida por los cuerpos debido a su temperatura y a diferencia de la conducción y la convección, la radiación térmica no necesita un medio para su propagación.

La razón de la radiación que se puede emitir desde una superficie a una temperatura termodinámica T_s es expresada por la **ley de Stefan-Boltzman**

$$\dot{Q}_{emitida} = \epsilon\sigma A_s T_s^4 \tag{2.39}$$

donde $\sigma = 5.67 \times 10^{-8} W/m^2 \cdot K^4$ es la constante de Stefan-Boltzman y $0 \leq \epsilon \leq 1$ es la emisividad de la superficie, para el cual $\epsilon = 1$ corresponde a la emisividad de un cuerpo negro.

2.7 Generación de calor

Muchas sistemas transfieren calor por medio de energía eléctrica, mecánica, nuclear o química. Por lo que es conveniente que el análisis en estos procesos puedan ser caracterizados como procesos con generación de calor.

La generación de calor es un fenómeno volumétrico. Es decir, ocurre en todo el medio. Por lo tanto, la velocidad de generación de calor en un medio suele especificarse por unidad de volumen y es denotado como $\dot{e}_g$ en W/m^3

La velocidad de generación de calor en un medio puede variar con el tiempo y con la posición.

Definición 2.43 *La velocidad total de la generación de calor en un medio de volumen V se puede determinar como:*

$$\dot{Q}_{gen} = \int_V \dot{e}_g dV \tag{2.40}$$

Si la generación de calor en el medio es uniforme, la relación se reduce a:

$$\dot{Q}_{gen} = \dot{e}_g V \tag{2.41}$$

donde $\dot{e}_g$ es la velocidad constate de generación de calor por unidad de volumen.

2.8. Sistemas de tiempo real

Actualmente podemos encontrar sistemas de tiempo real en cualquier tipo de dispositivo. Un sistema en tiempo real, consiste en un sistema de procesamiento de información el cual tiene que responder a estímulos de entrada generados externamente en un periodo finito y específico. Para que un sistema de tiempo real sea correcto depende no solamente de los resultados lógicos obtenidos, sino que también existe una dependencia del tiempo de entrega de los resultados producidos [31].

Las aplicaciones de sistemas en tiempo real son muy diversas y están presentes en diferentes campos de la ingeniería. Comúnmente los sistemas embebidos son ejemplos de sistemas en tiempo real, los cuales van desde aplicaciones en procesos de producción, sistemas de mando y control, telefonía móvil, realidad virtual, entre otras. En comparación con los sistemas de propósito general, los cuales sólo buscan encontrar un resultado a cualquier costo, los sistemas embebidos en tiempo real requieren hallar un resultado en un tiempo fijo. Si éstos resultados llegasen a encontrarse después del tiempo definido, el funcionamiento de alguna aplicación resultaría en fallos o errores catastróficos en un sistema mayor.

Definitivamente en los sistemas de tiempo real las restricciones temporales son un aspecto determinante, dependiendo de la aceptación de éstas restricciones, los sistemas de tiempo real pueden dividirse en dos categorías: sistemas en *tiempo real crítico* y sistemas de *tiempo real acrítico*. En los sistemas de tiempo real críticos (hard real time systems), la ejecución de

las tareas debe cumplir con su restricciones de tiempo, es decir cada tarea debe completar su ejecución antes de su respectivo límite de entrega. En cambio, en los sistemas de tiempo real acríticos (soft real time systems), las tareas también tienen restricciones de tiempo, sin embargo, pueden completar su ejecución en un mayor tiempo que lo especificado, dicho de otra manera, es posible tolerar el incumplimiento ocasional de la restricción temporal de una tarea. Otra categoría que podemos encontrar son los sistemas en tiempo real de tipo firme (Firm), éstos admiten una determinada pérdida ocacional de especificaciones temporales.

En general, un sistema de tiempo real se compone de un conjunto finito de tareas $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$, donde cada tarea τ_i contiene un conjunto de acciones similares que pudieran repetirse a lo largo del tiempo.

Otras características de tiempo se refieren a la regularización de su activación. En consecuencia, las tareas en tiempo real pueden ser *periódicas* dado a que su ejecución se realiza cada intervalo de tiempo fijo, *aperiódicas* las cuales no tienen un tiempo fijo entre llegadas, es decir se ejecutan en intervalos irregulares de tiempo, y las *esporádicas*, que corresponden a las tareas que ocurren repetitivamente pero con una separación en intervalos de tiempos arbitrarios [32] [33].

2.9. El problema de scheduling.

Un problema de scheduling es un problema de optimización, que tiene que ver con la asignación de recursos a tareas o actividades durante periodos de tiempo determinados.

En sistemas de fabricación y producción en la manufactura flexible, así como en la mayor parte de procesamiento de información, configuración de transporte y distribución, existen dos etapas de toma de decisiones que juegan un papel muy importante en el proceso: la planificación y el scheduling (calendarización).

La *planificacion* [34] es la etapa donde se formula que se va a hacer, cuántas u cuáles tareas se necesitan, que tiempo necesita cada una y qué recursos, obteniendo como resultado un plan a realizar en el proceso.

scheduling es el proceso de asignar recursos o actividades a tareas a lo largo del tiempo, es la etapa donde se selecciona la ordenación de tareas, la asignación de recursos y el calendario para su realización que optimiza cierto criterio, es decir, se obtiene un "schedule"óptimo de entre los que satisfacen el plan obtenido en la etapa previa.

La planificación es la etapa previa al scheduling. La planificación se encarga de obtener un plan de todos los posibles para que el scheduling lo ejecute de tal forma que se cumplan los objetivos de optimización que se hayan planteado.

En general un problema de scheduling consiste en la asignación de recursos a tareas y

al mismo tiempo el establecimiento de los tiempos de inicio de cada tarea. Las tareas en la mayoría de los casos, deben de cumplir un cierto tiempo para ser realizadas, es decir cada tarea tiene asignado un tiempo de duración. Cuando la tarea termina, se dice que ha sido ejecutada y los recursos que se le habían sido asignados se liberan. El que se encarga de estas decisiones se conoce como *schedule.*

Un schedule Factible o admisible, es aquel que es físicamente realizable, es decir satisface el conjunto de restricciones preestablecidos en un determinado plan, realizable por que se encuentra dentro de la capacidad de los recursos y requerimientos de uso de los recursos.

Los sistemas de tiempo real utilizan algoritmos de scheduling en tiempo de ejecución, en donde la asignación se resuelve al instante, eliminando la necesidad de un plan de ejecución previo. En éste trabajo se considera que la asignación se resuelve en tiempo de ejecución, por lo tanto nos enfocamos al problema de scheduling.

2.9.1. Complejidad de un problema de scheduling

Debido a la combinatoria de asignar un conjunto de tareas a un conjunto de recursos, la complejidad computacional del problema se convierte en un problema de tipo NP-Completo [35]. Esto significa que el problema es tan duro como otros problemas que han sido ampliamente estudiados y reconocidos como difíciles por los expertos en algoritmos. Como consecuencia, el tiempo necesario para encontrar un schedule óptimo no puede ser representado de forma polinomial, y por ello, se sabe que con las técnicas disponibles no es posible encontrar soluciones óptimas. Por lo tanto, en el estudio para este tipo de problema se ha optado en utilizar técnicas heurísticas que tienen una menor complejidad de cálculo. Dichas técnicas normalmente tienen la forma de reglas de "despacho"(dispaching rules), y no garantizan la optimalidad de la solución [36].

A fin de reducir la complejidad de la construcción de un problema de scheduling, una de las formas es simplificar la arquitectura del sistema, por ejemplo restringiendo el estudio al caso de sistemas uniprocesadores, teniendo precedencias en tareas, considerando tareas homogeneas (periódicas o aperiódicas), etc.

2.9.2. Clasificación de problemas de scheduling

Los tipos de problemas existentes se pueden clasificar en dos familias de scheduling, ésto según a la demanda de recursos en cierto tiempo y el suministro de ellos:

Problemas de scheduling "puros"

Problemas de asignación de recursos.

En los problemas de *scheduling "puros"*, cada procesador tiene una capacidad definida para un tiempo determinado. Aquí el problema de scheduling cubre la demanda de recursos necesarios para realizar las tareas a lo largo del tiempo, sin exceder la capacidad disponible. En los problemas de *asignación de recursos*, se disponen de un conjunto de tareas y para cada una de ellas se tienen un conjunto de procesadores idénticos (realizan el mismo tipo de operación, aunque no necesariamente requieran los mismos tiempos o costes de procesamiento). En este caso no se conoce a priori el procesador que será asignado a una determinada tarea, sino que el problema consiste en asignar los recursos a tiempo, cumpliendo las demandas establecidas.

Si se considera que una tarea puede interrumpirse y ser continuada más tarde, estamos hablando de un scheduling con reemplazo (preemptive scheduling). Si la ejecución de cada tarea se realiza completamente antes de abandonar el recurso asignado, se habla de un scheduling sin remplazo (non-preemptive scheduling).

Los algoritmos comunes de scheduling utilizan una asignación de prioridades a las tareas, y en función de ellas, se resuelven los posibles conflictos de utilización de los procesadores. Dependiendo de si la prioridad de las tareas es fija o cambia en función del estado del sistema, se dice que están basados en prioridades *estáticas* o *dinámicas*.

Una amplia variedad de estrategias de heurísticas para problemas de scheduling se pueden encontrar en [35]. Algunas de las técnicas más representativas para el problema de scheduling son las siguientes:

Reglas de prioridad de despacho (priority dispatch rules), es una de las estrategias más comunes para resolver problemas de scheduling y consisten en definir prioridades entre las tareas que se desean realizar. Se trata en general de algoritmos voraces que toman desiciones basadas en algún criterio de optimización local.

a FIFO (First In First Out). Posiblemente esta técnica sea la más sencilla para problemas de scheduling. Crea una cola de ejecución donde la primer tarea a entrar es la que se ejecuta. La principal ventaja de ésta técnica es la simplicidad en su implementación, además de ser intuitivamente razonable. Un inconveniente es que una tarea de larga duración retrasa a todas las tareas detrás de ella.

b SJF (Shortest Job First). Es una de las técnicas más utilizadas para sistemas que tengan características de tiempo de entraga máximo. Consiste en ordenar la cola de ejecución de las tareas de menor a mayor tiempo de procesamiento y cuando un recurso está disponible se ejecuta la tarea mas rápida.

c EDF (Earliest DeadLine First). Es muy utilizada en sistemas de tiempo real y consiste en ordenar la cola de ejecución dependiendo de la tarea que este más cercana a su fecha límite (deadline). Este algoritmo es óptimo para procesos con un sólo recurso, es decir si se tiene una cola de ejecución de tareas que estén caracterizadas por un tiempo de

ejecución y una fecha de límite el algoritmo asegura que todas las tareas se asignen al recurso cumpliendo con sus fechas límites.

d RM (Rate Monotic) Es un algoritmo para sistemas de tiempo real. El algoritmo RM asigna las prioridades a las tareas de acuerdo a sus períodos, mientras más corto sea el periodo de la tarea mayor es su prioridad.

Capítulo 3

Modelo Térmico

En este capítulo se presenta una metodología para el modelado térmico de un sistema de microprocesadores en chip mediante Redes de Petri Continuas Temporizadas (TCPN). Se asume que el sistema está sujeto a conducción, convección térmica y generación de calor que surge de la energía eléctrica consumida por los procesadores. En esta metodología se modelan por separado los mecanismos de transferencia de calor y después se integran en un sólo modelo global que captura el comportamiento térmico de todo el sistema.

3.1. Modelo de conducción Térmica

La energía calorífica que se transfiere en un sistema de microprocesadores en chip es en gran medida por la conducción. La conducción sucede cuando existe contacto directo entre elementos a diferentes temperaturas, transfiriendo el calor en dirección normal a la superficie de contacto.

Para el análisis de transferencia de calor por medio de conducción considere dos componentes con geometrías prismáticas y propiedades isotrópicas como se muestra en la Fig. 3.1. Ambos componentes se encuentran en contacto con temperaturas constantes T_1 y T_2 respectivamente, donde $T1 \neq T_2$. Entonces el calor fluye del componente con mayor temperatura al que posee menos temperatura en dirección normal a la superficie de contacto A. Denotando a k_1, k_2 como los coeficientes de conductividad térmica, V_1,V_2 los volúmenes de control, ρ_1,ρ_2,cp_1,cp_2 las densidades y el calor específico de los componentes 1 y 2 respectivamente, con Δx_1,Δx_2 como la distancia ortogonal entre el centro de cada cara al punto de contacto de las componentes, entonces la conducción unidimensional del calor en estado estacionario a través de los elementos se formula mediante la ley de Fourier de la conducción de calor de la ecuación (2.40).

$$\begin{aligned} \dot{Q}_1 &= k_1 A \frac{T_1 - T_2}{\Delta x} \\ \dot{Q}_2 &= k_2 A \frac{T_2 - T_1}{\Delta x}. \end{aligned} \tag{3.1}$$

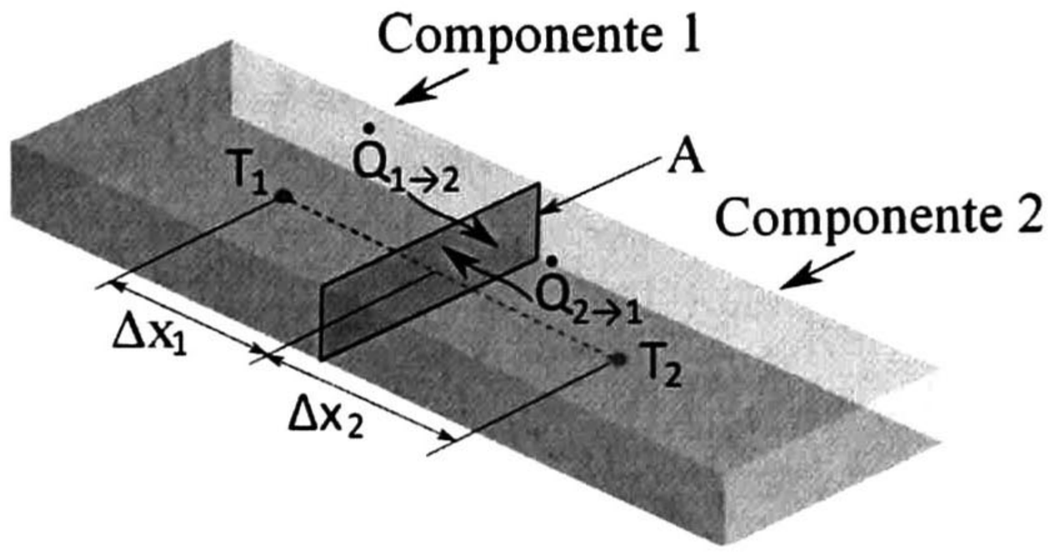


Figura 3.1: Dos componentes sujetos a transferencia de calor por conducción.

Por lo tanto, usando la aproximación en diferencias finitas aplicado a la ley de Fourier para conducción de calor, el calor transferido entre el componente 1 hacia el componente 2 y viceversa se obtiene mediante:

$$\begin{aligned} \dot{Q}_{1\to2} &= \frac{k_1 k_2 A}{k_2\Delta x_1 + k_1 \Delta x_2} \cdot (T_1 - T_2) \\ \dot{Q}_{2\to1} &= \frac{k_1 k_2 A}{k_2\Delta x_1 + k_1 \Delta x_2} \cdot (T_2 - T_1). \end{aligned} \tag{3.2}$$

Por otra parte, si se tiene un elemento de volumen de control con temperatura constante a lo largo de cada componente, entonces el incremento de energía térmica interna del volumen se puede expresar como:

$$\begin{aligned} \dot{Q}_1 &= V_1 \rho_1 cp_1 \dot{T}_1 \\ \dot{Q}_2 &= V_2 \rho_2 cp_2 \dot{T}_2 \end{aligned} \tag{3.3}$$

Asumiendo que ningún componente genera calor ni transfiere calor por convección, entonces la dinámica de la temperatura puede ser aproximada combinando las ecuaciones (3.2) y (3.3) como sigue:

$$\begin{aligned} V_1 \rho_1 cp_1 \dot{T}_1 = \dot{Q}_1 = -\dot{Q}_{1\to2} = -\frac{k_1 k_2 A}{k_2\Delta x_1 + k_1 \Delta x_2} \cdot (T_1 - T_2) \\ V_2 \rho_2 cp_2 \dot{T}_2 = \dot{Q}_2 = -\dot{Q}_{2\to1} = -\frac{k_1 k_2 A}{k_2\Delta x_1 + k_1 \Delta x_2} \cdot (T_2 - T_1). \end{aligned} \tag{3.4}$$

Así, la dinámica de temperatura de los componentes bajo transferencia de calor por conducción se puede expresar como:

$$\begin{aligned}\dot{T}_1 &= -\frac{1}{V_1\rho_1 cp_1}\cdot\frac{k_1k_2A}{k_2\Delta x_1+k_1\Delta x_2}\cdot(T_1-T_2)\\ \dot{T}_2 &= -\frac{1}{V_2\rho_2 cp_2}\cdot\frac{k_1k_2A}{k_2\Delta x_1+k_1\Delta x_2}\cdot(T_2-T_1).\end{aligned} \tag{3.5}$$

Las dos ecuaciones diferenciales anteriores representan una aproximación del comportamiento dinámico de temperatura de ambos componentes sujetos a conducción. La aproximación que se obtiene mediante estas ecuaciones es válida mientras la temperatura sea casi constante a lo largo de cada componente, lo cual sucede cuando se consideran componentes lo suficientemente pequeños como en el análisis de diferencias finitas.

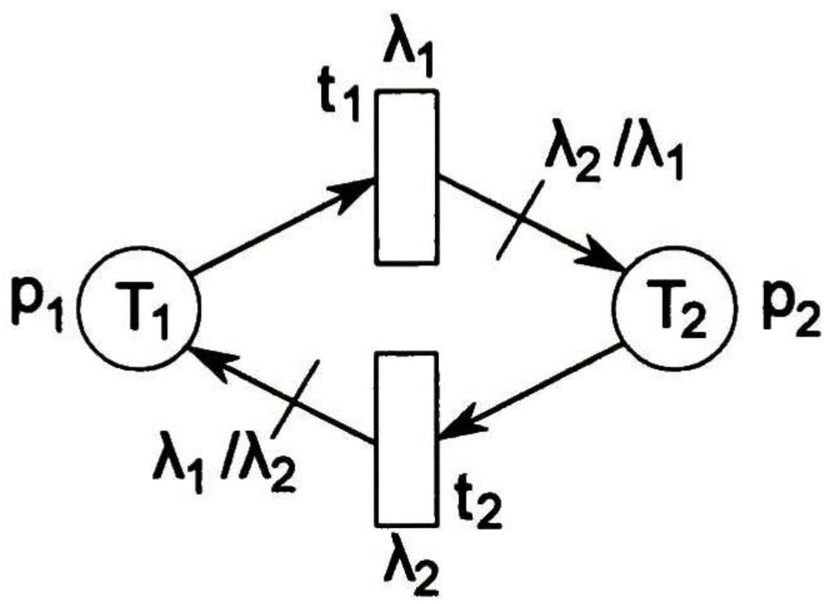


Figura 3.2: Modelo en TCPN de transferencia de calor por conducción.

Las ecuaciones en (3.5) pueden ser modeladas mediante una red $TCPN$ con semántica de servidores infinitos como se muestra en la Fig. 3.2. Esta red está compuesta por dos lugares que representan la temperatura de cada componente y dos transiciones que capturan el comportamiento del flujo de calor que existe entre ellos. Por lo tanto, la dinámica de la temperatura en cada elemento está representada mediante el marcado de cada lugar el cual puede ser expresado por la ecuación fundamental de las TCPN ecuación (2.20).

El comportamiento del flujo de las transiciones en la semántica de servidores infinitos necesita del vector de tasa de disparos λ el cual queda definido como:

$$\lambda = \begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix} = \begin{bmatrix}\frac{1}{V_1\rho_1 cp_1}\cdot\frac{k_1k_2A}{k_2\Delta x_1+k_1\Delta x_2}\\ \frac{1}{V_2\rho_2 cp_2}\cdot\frac{k_1k_2A}{k_2\Delta x_1+k_1\Delta x_2}\end{bmatrix} \tag{3.6}$$

La ecuación de la dinámica de esta red está en función de las matrices de $Post$ y Pre incidencia:

$$Pre = \begin{bmatrix}1\,0\\ 0\,1\end{bmatrix} \qquad Post = \begin{bmatrix}0 & \frac{\lambda_1}{\lambda_2}\\ \frac{\lambda_2}{\lambda_1} & 0\end{bmatrix} \tag{3.7}$$

entonces, la dinámica de la red de Petri se puede expresar como:

$$m_{cond}^{\cdot} = \begin{bmatrix} \dot{p_1} \\ \dot{p_2} \end{bmatrix} = \begin{bmatrix} -1 & \frac{\lambda_1}{\lambda_2} \\ \frac{\lambda_2}{\lambda_1} & -1 \end{bmatrix} \begin{bmatrix} \lambda_1 p_1 \\ \lambda_2 p_2 \end{bmatrix} \tag{3.8}$$

donde p_1, p_2 representan las temperaturas de T_1 y T_2 que coinciden con la solución de la ecuación diferencial de las ecuaciones (3.5).

3.2. Modelo de convección Térmica

El mecanismo de transferencia de calor por convección en un MPSoC's se produce por medio de la transferencia de energía que existe entre la superficie sólida y el aire circundante. La convección puede ser natural o forzada. Para el caso de estudio y modelado, la convección en este trabajo se considera natural.

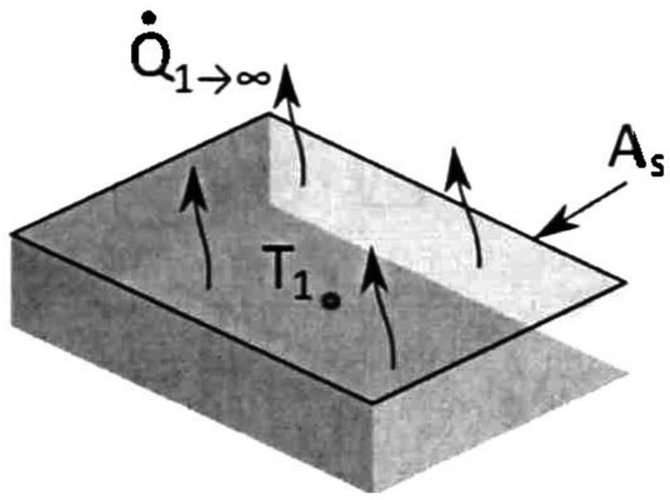


Figura 3.3: Transferenia de calor de un componente por convección de una cara hacia el aire.

Consideremos un componente prismático con temperatura T_1 a lo largo del componente como se muestra en la Fig. 3.4. Asumiendo que existe transferencia de calor en una cara con área A_S hacia el aire circundante con temperatura T_∞ y asumiendo que el coeficiente de convección h es conocido, se tiene que la rapidez de la transferencia de calor por convección es proporcional a la diferencia de temperatura entre la superficie caliente hacia el aire y se expresa por medio de la ley de enfriamiento de Newton como:

$$\dot{Q}_{1\to\infty} = hA_s(T_1 - T_\infty) \tag{3.9}$$

Considerando que el componente no genera calor ni transfiere calor por conducción hacia otros componentes y la transferencia de calor por convección solo prevalece en una cara del componente, la dinámica de la temperatura en el componente puede ser aproximada combinando las ecuaciones (3.3) y (3.9) como:

$$V_1\rho_1 cp_1\dot{T}_1 = -\dot{Q}_{1\to\infty} = -hA_s(T_1 - T_\infty) \tag{3.10}$$

Por lo tanto, el comportamiento de la dinámica de temperatura del componente resulta:

$$\dot{T}_1 = -\frac{hA_s}{V_1\rho_1 cp_1} \cdot (T_1 - T_\infty) \tag{3.11}$$

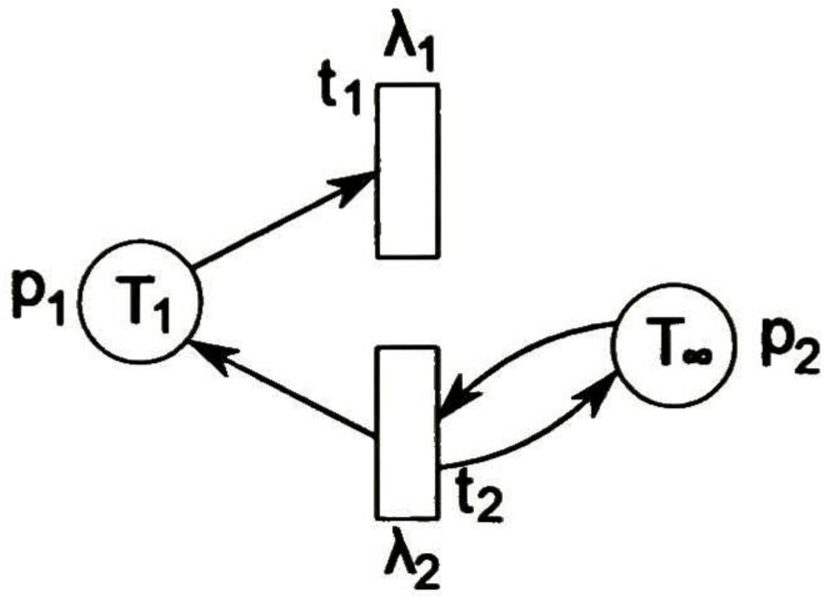


Figura 3.4: Modelo en TCPN de transferencia de calor por convección.

La ecuación diferencial anterior representa la dinámica de la temperatura sujeto a transferencia de calor por convección de una cara del componente hacia el aire circundante. Esta ecuación puede modelarse por medio de una red TCPN con semántica de servidores infinitos como se muestra en la Fig.3.4. La red está compuesta por dos lugares que representan la temperatura del componente T_1 y la temperatura del aire T_∞ respectivamente. La transición t_2 es una transición fuente bien definida que inyecta marcas a p_1 a una razón λ_2, la transición t_1 quita marcas de p_1 a razón de λ_1. El flujo en las transiciones está definida por el vector λ como:

$$\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \frac{hA_s}{V_1\rho_1 cp_1} \\ \frac{hA_s}{V_1\rho_1 cp_1} \end{bmatrix}$$

Las matrices de Post y Pre incidencia de esta red son:

$$Pre = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad Post = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \tag{3.12}$$

Por lo tanto, la dinámica de la red de Petri se escribe como:

$$\dot{m}_conv = \begin{bmatrix} \dot{p_1} \\ \dot{p_2} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 p_1 \\ \lambda_2 p_2 \end{bmatrix} \tag{3.13}$$

El marcado de los lugares p_1 y p_2 representan la temperatura (promedio) del componente 1 y la temperatura del aire circundante, respectivamente. La evolución del marcado en p_1 coincide con la solución de la ecuación diferencial obtenida en (3.11).

3.3. Modelo de generación de calor

La generación de calor en un sistema de multiprocesadores en chip surge de la conversión de energía eléctrica a energía térmica en los procesadores. Esta conversión sucede debido al procesamiento de datos en los procesadores lo cual resulta en una generación de calor internamente, ésto se manifiesta en todo el medio mediante una elevación de temperatura. A medida que incrementa la temperatura en el medio, también aumenta la transferencia de calor de ese medio a sus alrededores. Esto continua hasta alcanzar las condiciones de operación estacionarias donde la temperatura en el medio no cambia y la velocidad de generación de calor es igual a la razón de transferencia de calor hacia los alrededores. La velocidad de generación de calor puede variar con el tiempo o puede considerarse uniforme.

La generación de calor es un fenómeno que depende estrictamente del volumen del cuerpo. Por lo que se debe de determinar en función de unidades de volumen. Considere un componente prismático cuya temperatura T_1 es casi constante a lo largo del componente. Entonces, el componente genera energía térmica uniformemente a una velocidad g_1 indicando la potencia que se genera por unidades de volumen. Por lo tanto, la energía térmica que se genera en todo el componente está dada por la ecuación (2.41).

$$\dot{Q}_{\to 1} = V_1 g_1 \tag{3.14}$$

Asumiendo que no existe transferencia de calor por conducción de ó hacia otros componentes ó convección hacia el aire, entonces el comportamiento de la dinámica de temperatura para este fenomeno puede ser expresada combinando las ecuaciones (3.10) y (3.14) como:

$$V_1 \rho_1 cp_1 \dot{T}_1 = -\dot{Q}_{\to 1} = V_1 g_1 \tag{3.15}$$

despejando de la ecuación anterior para $\dot{T}_1$ se obtiene:

$$\dot{T}_1 = \frac{g_1}{\rho_1 cp_1} \tag{3.16}$$

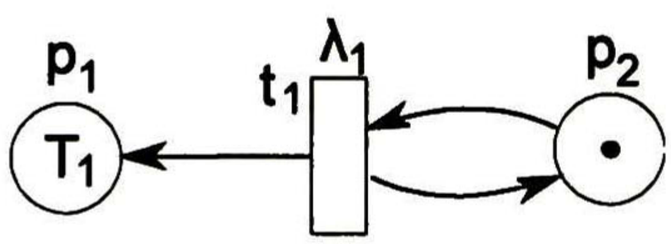


Figura 3.5: Modelo en TCPN de la Generación de calor.

La ecuación diferencial anterior puede modelarse con una TCPN bajo la semántica de servidores infinitos con una transición fuente bien definida como se muestra en la Fig. 3.5. El lugar p_2 inyecta

marcas con una rapidez λ_1 al lugar p_1 que representa el componente. Entonces la ecuación puede representarse por una TCPN como:

$$\lambda_1 = \frac{g_1}{\rho_1 cp_1} \tag{3.17}$$

las matrices de Post y Pre incidencia son:

$$Pre = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad Post = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.18}$$

la dinámica de la red de Petri queda expresada como:

$$m_{gen}^{\cdot} = \begin{bmatrix} \dot{p_1} \\ \dot{p_2} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \lambda_1 p_1. \tag{3.19}$$

Note que el marcado del lugar p_1 representa la temperatura (promedio) del componente. En cambio el lugar p_2 permanece constante con marcado inicial $m_{1_0} = 1$. La evolución del marcado en p_1 es exactamente la solución de la ecuación diferencial (3.16).

3.4. Integración de Conducción, Convección y Generación de calor

Para el análisis térmico de un gran número de dispositivos en ingeniería como lo es un Sistema de multiprocesadores en Chip, el sistema se descompone en volúmenes de control como en el análisis de elemento finito. El enfoque principal de éste análisis es dividir la geometría en la que se quiere resolver una ecuación diferencial en pequeños elementos, teniendo en cuenta que cada elemento genera una ecuación diferencial.

Como se menciono al inicio del capítulo, el calor se transfiere por los mecanismos de conducción, convección y radiación, este último no es considerado en el modelado del sistema. La energía que se tiene en un volumen de control por medio de estos mecanismos se obtiene mediante el principio de conservación de la energía. Donde La energía final menos la energía inicial es igual a la energía neta transferida.

Considere el componente prismático 1 de la Fig. 3.7, con temperatura T_1 casi constante a lo largo del componente. Sean $T_2, T_3, \ldots, T_k$ las temperaturas (promedio) de los componentes vecinos con propiedades: $V_1, V_2, \ldots, V_k$, $cp_1, cp_2, \ldots, cp_k$, y $\rho_1, \rho_2, \ldots, \rho_k$, los volumenes, capacidad específica y densidades, respectivamente. El área entre el componente 1 y cualquier componente vecino j se denota por $A_{1,j}$. La distancia ortogonal con la cara de contacto que existe entre el centro del componente 1 hacia el punto de contacto con algún componente vecino j es $\Delta x_{1,j}$ de manera similar la distancia del centro del componente j hacia el punto de contacto se denota como $\Delta x_{j,1}$.

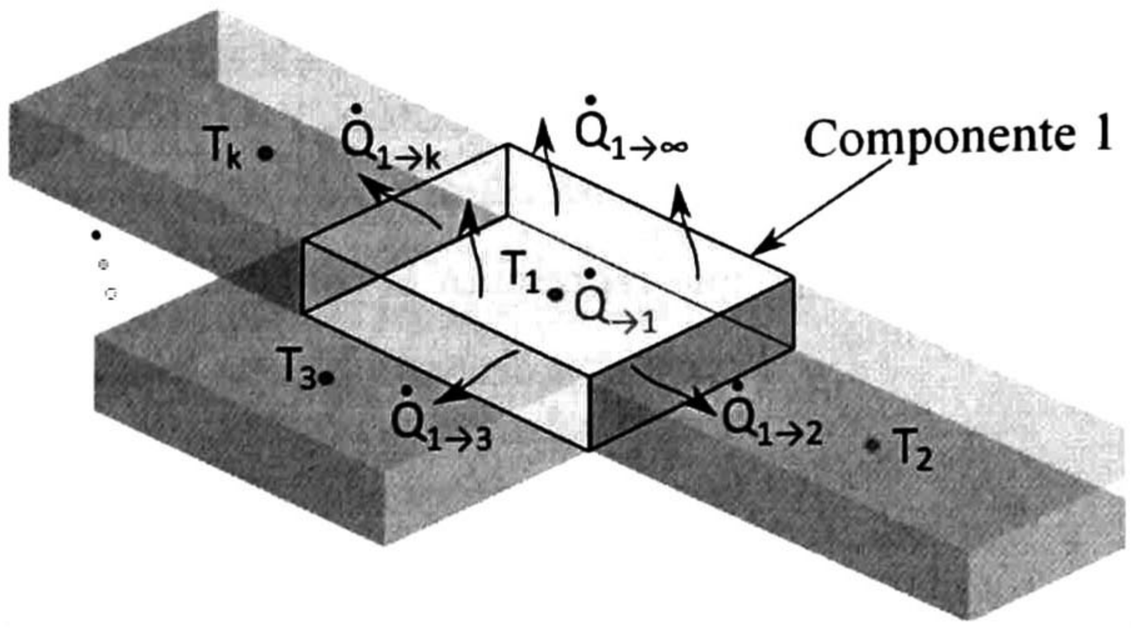


Figura 3.6: Componente que genera calor a g_1 y transfiere a sus vecinos por conducción y hacia el aire por convección.

Considerando también que el componente 1 genera calor a una tasa uniforme g_1 y se transfiere calor en todo el medio mediante conducción con $k_1, k_2, \ldots, k_k$ como los coeficientes de conductividad térmica de los componentes vecinos y por convección con coeficiente de convección h sobre la superficie A_S hacia el aire circundante con temperatura T_∞, descartando la radiación, entonces la razón de transferencia de calor del componente 1 hacia sus componentes vecinos puede expresarse haciendo un balance de energía como sigue

$$\dot{Q}_1 = -\sum_{j=2}^{k} \dot{Q}_{1\to k} - \dot{Q}_{1\to\infty} + \dot{Q}_{\to 1} \tag{3.20}$$

Usando las ecuaciones (3.2), (3.3), (3.9), (3.14), el incremento de energía térmica en el componente 1 se escribe como

$$\begin{aligned} \dot{Q}_1 = V_1\rho_1 cp_1 \dot{T}_1 = &-\textstyle\sum_{j=2}^{k} \frac{k_1 k_j A_{1,j}}{k_j \Delta x_{1,j} + k_1 \Delta x_{j,1}} (T_1 - T_j) \\ &-hA_s(T_1 - T_\infty) + V_1 g_1 \end{aligned} \tag{3.21}$$

por lo tanto, la evolución de la dinámica de temperatura (promedio) en el componente 1 puede expresarse por medio de la siguiente ecuación

$$\dot{T}_1 = -\frac{1}{V_1\rho_1 cp_1} \sum_{j=2}^{k} \frac{k_1 k_j A_{1,j}}{k_j \Delta x_{1,j} + k_1 \Delta x_{j,1}} (T_1 - T_j) - \frac{1}{V_1\rho_1 cp_1} hA_s(T_1 - T_\infty) + \frac{1}{\rho_1 cp_1} g_1 \tag{3.22}$$

La ecuación anterior representa la dinámica de temperatura de un componente que genera calor y lo transfiere a sus vecinos y hacia el aire. Mediante esta ecuación se puede crear un modelo que capture el comportamiento térmico en una TCPN.

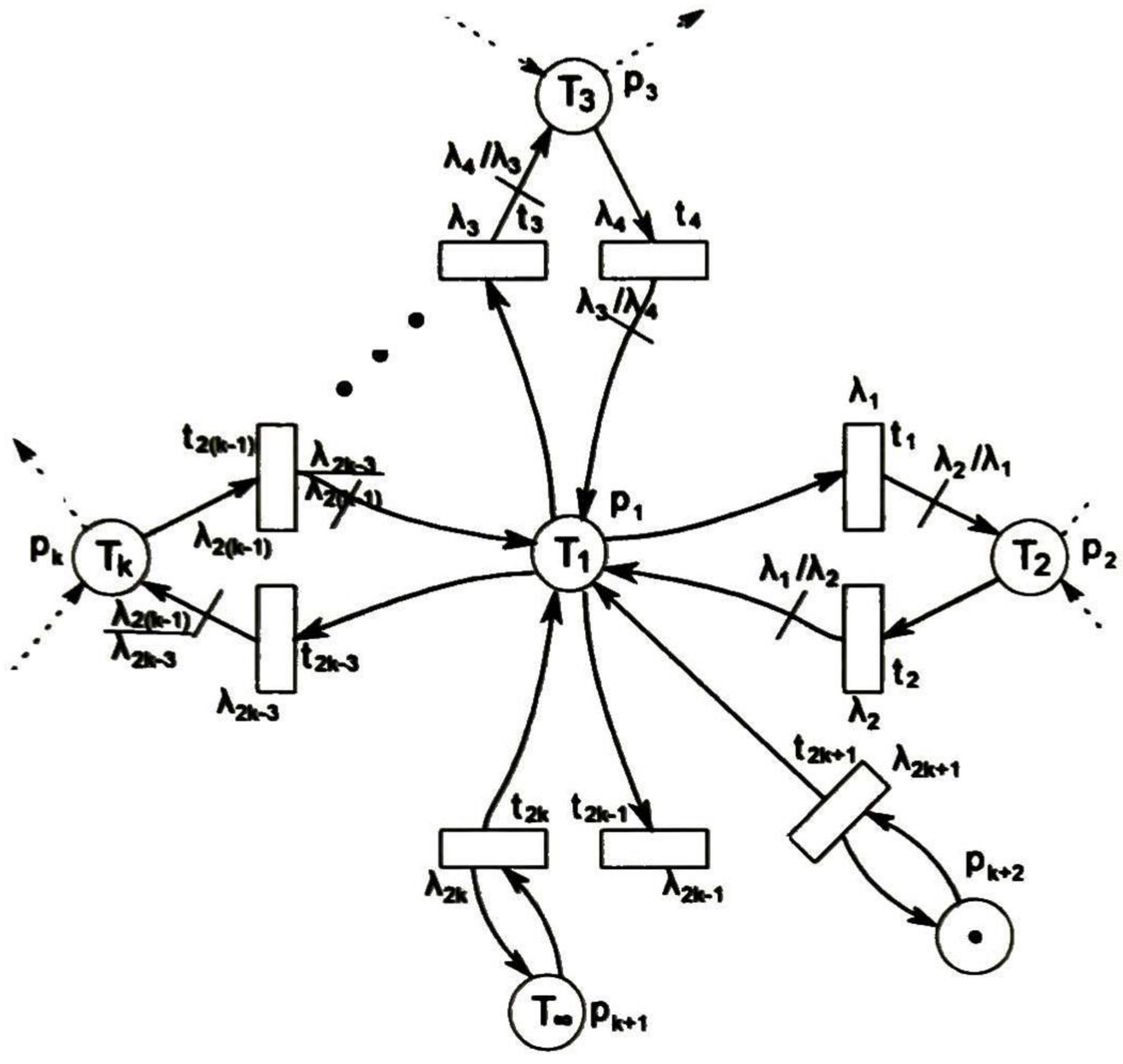


Figura 3.7: Modelo en TCPN de un componente que genera calor y transfiere a sus vecinos por conducción y hacia el aire por convección

El comportamiento de la TCPN de la Fig. 3.7 es igual a la solución de la ecuación diferencial (3.22). Aquí el lugar p_j donde $j = \{1, \ldots, k\}$ representa la temperatura promedio de la $j - th$ componente del sistema. El marcado en p_{k+1} representa la temperatura del aire circundante mientras que el marcado en p_{k+2} es constante e igual a 1. Los arcos de entrada y salida del lugar p_1, representan la dinámica de transferencia de calor por conducción entre los componentes vecinos $p_2, p_3, \ldots, p_k$, la convección hacia el aire y la generación de calor en el mismo componente. En este sentido, el modelo en TCPN de la Fig. 3.7 representa la dinámica de temperatura del componente 1 con respecto a las temperaturas de los componentes vecinos.

3.5. Ejemplo Ilustrativo.

En esta sección se ilustra la metodología antes descrita mediante un ejemplo que se presenta a continuación.

Ejemplo 3.1 El calor generado en la circuitería sobre la superficie de un chip de silicio se conduce hasta la placa de baquelita de $3cm \times 3cm$ con un espesor de $2mm$ al cual está sujeto como se

Tabla 3.1: Propiedades de los Materiales

	Silicio	Baquelita	Cobre
cp	712 J/Kg^oK	820 J/Kg^oK	385 J/Kg^oK
ρ	2330 Kg/m^3	1300 Kg/m^3	8933 Kg/m^3
k	148 W/m^oC	50 W/m^oC	400 W/m^oC

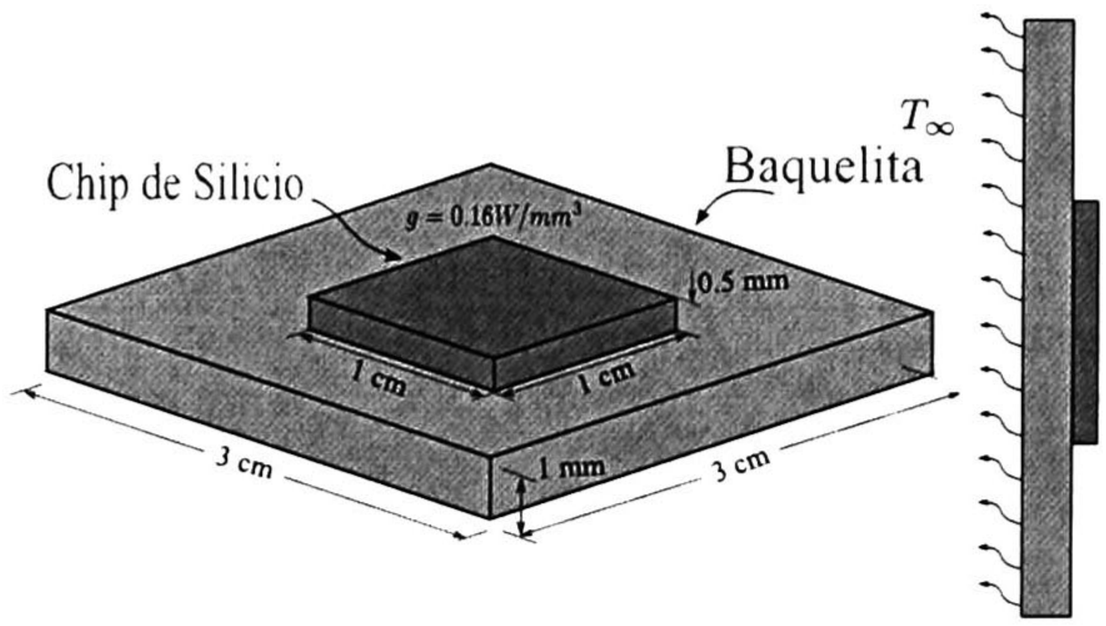


Figura 3.8: Chip de silicio sobre una placa de baquelita

muestra en la Fig. 3.8. El chip tiene un volumen de $1cm \times 1cm \times 1mm$ y genera calor uniformemente a $g = 0.16W/mm^3$. Considerando que la baquelita transfiere calor hacia el aire con temperatura $T_\infty = 40^oC$ con un coeficiente de convección $h = 0.001W/mm^{2o}C$, se desea obtener un modelo térmico del sistema. Las propiedades isotropicas de los materiales se encuentran en la Tabla 3.1.

Conociendo la geometría del sistema y sus propiedades isotrópicas proporcionadas en la Tabla 3.1 se procede a construir el modelo en TCPN, para ello es necesario dividir el sistema en elementos prismáticos sencillos. Para ilustrar la metodología de modelado en este ejemplo, se considera una resolución de mallado para el sistema de $1cm \times 1cm \times 1mm$ para el chip y $1cm \times 1cm \times 2mm$ para la placa. Con esta resolución se obtienen nueve elementos en la placa de baquelita y un elemento para el chip de silicio.

El modelo en TCPN se construye con los diez elementos que surgen de la partición, donde cada elemento es modelado por un lugar en la red que captura el incremento de temperatura mediante la dinámica del marcado. La Fig. 3.9 muestra la partición del sistema con la resolución antes mencionada y a su vez muestra los elementos volumétricos con su respectivo modelo en TCPN. De aquí se obtienen 12 lugares donde 10 de ellos representan la temperatura de cada elemento $(p_1, \ldots, p_{10})$ y los restantes representan la temperatura ambiente del medio p_{11} y una transición fuente bien definida p_{12} para la generación de calor. Las transiciones $t_1, \ldots, t_{26}$ representan la

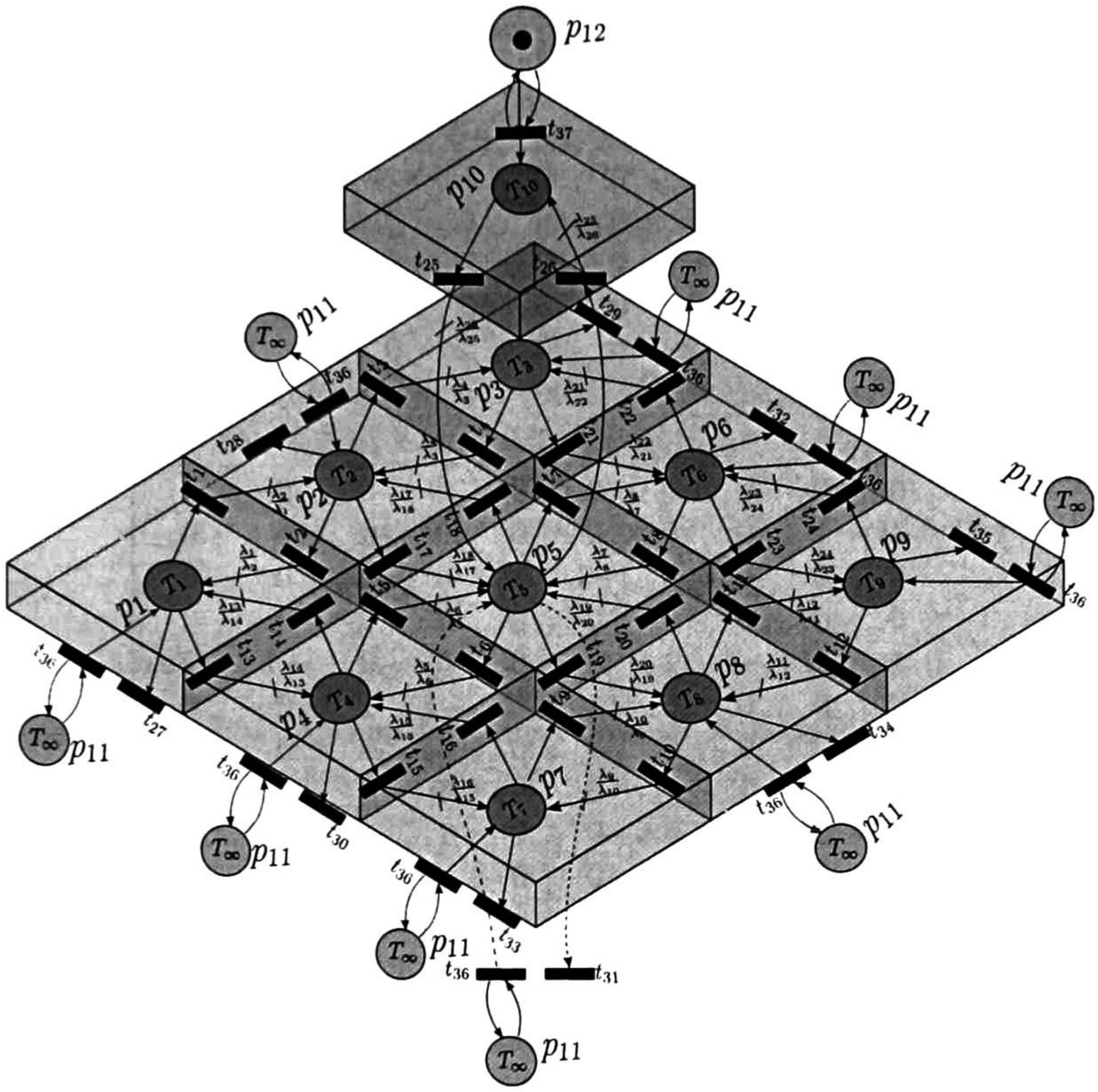


Figura 3.9: Modelo en TCPN del ejemplo 3.1

transferencia de calor mediante conducción de los elementos prismáticos, las transiciones $t_{27}, \ldots, t_{36}$ representan la transferencia de calor de la placa al medio ambiente por convección y la generación de calor en el chip es representado por la transición t_{37}. Por otra parte, note que el modelo considera la transferencia de calor por conducción entre elementos adyacentes.

Una representación matemática del modelo en TCPN se puede obtener mediante las ecuaciones de estados para los diferentes mecanismos de transferencia de calor. Las matrices de incidencia que rigen el comportamiento del cambio de temperatura entre elementos adyacentes ocasionado por el contacto entre ellos por conducción, convección o en su caso por generación de calor son C_{cond}, C_{conv} y C_{gen} respectivamente.

Columnas de 1-16.

$$C_{cond} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdots$$

Columnas de 17-26.

$$\cdots \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0.79 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.29 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_{conv} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad C_{gen} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.23}$$

Por lo tanto, para representar el comportamiento térmico de todo el sistema se obtiene una matriz de incidencia global compuesta por las matrices anteriores.

$$C = [C_{cond}\, C_{conv}\, C_{gen}] \tag{3.24}$$

Las tasas de disparos λ_{cond}, λ_{conv} y λ_{gen} para las transiciones en este modelo son calculadas mediante las ecuaciones descritas en la Sección 3.4 y mediante los parámetros de la tabla 3.1.

$$\lambda_{cond} = \begin{matrix}[\,0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047 \\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 0.0047\ 20.06\ 25.79\,]^T\end{matrix}$$

$$\lambda_{conv} = [\ 0.47\ 0.47\ 0.47\ 0.47\ 0.47\ 0.47\ 0.47\ 0.47\ 0.47\ 0.47\ ]^T$$

$$\lambda_{gen} = 78.36 \tag{3.25}$$

La tasa total de disparos de todas las transiciones en el modelo global se puede escribir como:

$$\lambda = \begin{bmatrix} \lambda_{cond} \\ \lambda_{conv} \\ \lambda_{gen} \end{bmatrix} \tag{3.26}$$

Dado que la red TCPN de esta metodología pertenece a la clase de red llamada máquina de estados, las cuales no tienen sincronizaciones en las transiciones, entonces se deduce que solo existe una configuración en la red. Por lo tanto, se puede considerar a la matriz de configuraciones como única y descrita como $\Pi(m) = Pre^T$, consiguientemente el flujo en las transiciones se puede escribir en forma vectorial como $f(m) = \Lambda\Pi(m)m$, donde Λ es la matriz diagonal cuyos elementos son los mismos que λ, así el comportamiento dinámico del sistema en ecuaciones de estados se puede expresar como

$$\dot{m} = C\Lambda\Pi(m)m. \tag{3.27}$$

Note que la ecuación anterior de la red de Petri del modelo es una ecuación de estados lineal, por lo que podemos reescribirla de la siguiente manera

$$\begin{aligned} \dot{m} &= C\Lambda\Pi m - Cu = Am + Bu \\ y &= Sm. \end{aligned} \tag{3.28}$$

donde y es el estado medido que corresponde a las temperaturas en los procesadores y $u = \frac{q}{v}$ Watts la potencia consumida por el procesador y representa la entrada al sistema, la matriz B es la matriz que mapea el sistema de entrada a las transiciones correspondientes.

3.6. Validación del modelo mediante ANSYS

ANSYS es un software de simulación ingenieril que fue desarrollada bajo la teoría de elementos finitos para la solución de problemas mecánicos, acústica, electromagnetismo y dinámica de fluidos. En problemas mecánicos incluye el análisis de transferencia de calor donde la solución que otorga es exacta y confiable en procesos industriales.

Debido a la exactitud de los resultados que se obtienen con un modelo creado en ANSYS, se opta usar este software como base de comparación y validación para la metodología propuesta en la sección 3.4. Para validar un modelo creado por medio de ésta metodología considere el siguiente ejemplo.

Ejemplo 3.2 Considere un sistema de microprocesadores en chip con cuatro procesadores de silicio de $1cm \times 1cm$ montados en una placa difusora de calor de $5cm \times 5cm$ hecha de baquelita. Cada procesador tiene $0.5mm$ de grosor y generan calor a una tasa uniforme de $g_1 = 0.13W/mm^3, g_2 = 0.15W/mm^3, g_3 = 0.18W/mm^3, g_4 = 0.20W/mm^3$. La placa difusora de cobre de $1mm$ de grosor transfiere calor por convección en una de sus caras (cara opuesta al la cara donde están montados los microprocesadores) hacia el aire circundante de temperatura fija a 45^oC (el cual es el valor que se asume generalmente en los diseños térmicos [6]). Las propiedades isotropicas de los materiales se encuentran en la Tabla 3.1 y el coeficiente de convección se asume que es $h = 0.001W/mm^{2o}C$. Se desea obtener un modelo que capture la dinámica de temperatura en los procesadores.

La metodología propuesta basa su operación en la capacidad de dividir el sistema en un número finito de elementos, posteriormente modela los mecanismos de transferencia de calor como una TCPN y finalmente crea un modelo global mediante la unión de los modelos que capturan el comportamiento térmico total del sistema.

Usando la metodología descrita en la Sección 3.4 se obtiene el modelo global en TCPN donde un elemento prismático queda representado por un lugar en la red y la dinámica del marcado corresponde a la temperatura de dicho elemento. Para este ejemplo el sistema se modela y simula (con SimHPN [28]) a diferentes resoluciones (cantidad de elementos) con la finalidad de obtener soluciones del sistema para su posterior comparación.

Las resoluciones del mallado que se consideran en este ejemplo son: $1cm \times 1cm$, $0.5cm \times 0.5cm$, $0.25cm \times 0.25cm$ y $0.20cm \times 0.20cm$, donde en todos los casos, el espesor del elemento es igual al espesor de la placa o al espesor de los chips ($1mm$ ó $0.5mm$).

Para el primer caso el sistema se divide en elementos prismáticos de $1cm \times 1cm$, de esta partición se tienen 25 componentes de dimensión $1cm \times 1cm \times 0.1cm$ en la placa de baquelita, mientras que cada procesador conforma un elemento en esta división. La Fig. 3.11 muestra la resolución del mallado y a su vez muestra la distribución de temperatura de cada elemento.

Si se considera una resolución de $0.5cm \times 0.5cm$, la placa se parte en 100 elementos prismáticos, mientras los procesadores se dividen en 4 elementos de dimensión $0.5cm \times 0.5cm \times 0.05cm$. Note que para esta resolución la solución que se obtiene difiere a la solución anterior (ver en la Fig. 3.12),

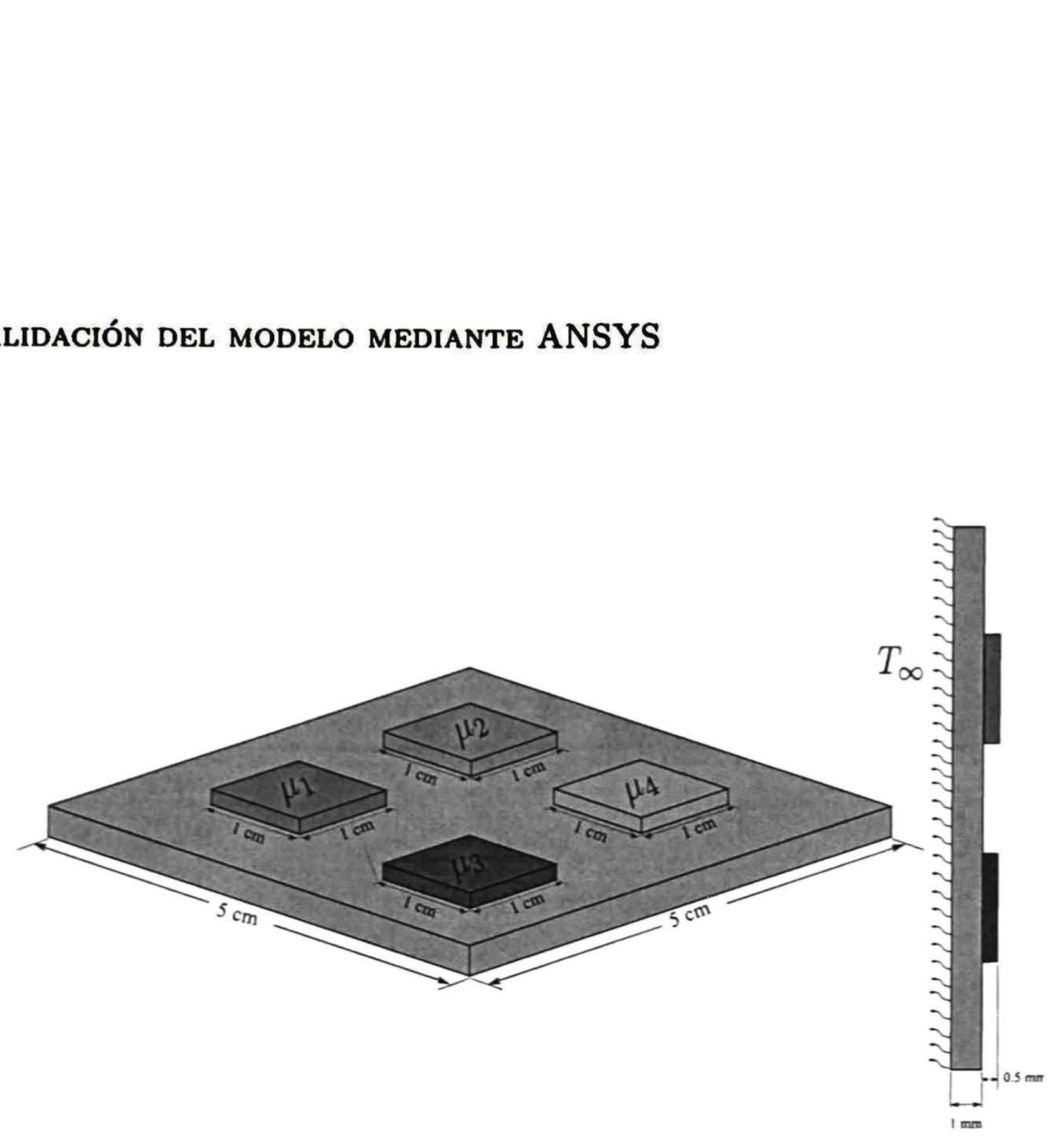


Figura 3.10: 4 procesadores de silicio montados sobre una placa de baquelita

esto es debido a que el calor se transfiere a más elementos adyacentes a los procesadores, es decir existe una mayor distribución de temperatura en todo el sistema.

Las figuras Fig. 3.13 y Fig. 3.14 corresponden a la distribución de temperaturas para los casos de resoluciones $0.25cm \times 0.25cm$ y $0.20cm \times 0.20cm$. Comparando la solución del modelo obtenido con estas resoluciones se observa que las temperaturas tanto en los procesadores como en la placa en las figuras Fig. 3.13(b) y Fig. 3.14(b) son similares. Una resolución mayor por consiguiente, resulta en una solución similar a las anteriores, entonces una resolución de $0.20cm \times 0.20cm$ es suficiente para este problema.

No se tiene una metodología para elegir una resolución adecuada, no obstante esta elección depende de la precisión que se requiera en la solución. Crear un modelo mas complejo (con un mayor número de componentes) resulta en un incremento de la cantidad de ecuaciones diferenciales a resolver, ésto a su vez conlleva a ralentizar la simulación del sistema. Sin embargo si se desea obtener la temperatura precisa de un punto en particular es conveniente considerar una resolución mayor. Así, se puede deducir que la complejidad de implementar un modelo térmico depende de la geometría del sistema, su homogeneidad y la resolución requerida en su descomposición.

Para la validación del modelo, se considera que un sólo procesador trabaja generando calor a una tasa de $g_1 = 0.13W/mm^3$, los procesadores restantes no trabajan y por tanto no generan calor $g_2 = g_3 = g_4 = 0W/mm^3$. Bajo estas consideraciones, el modelo es simulado en SimHPN [28] donde la evolución de temperaturas en los procesadores que se obtiene en la simulación es comparado

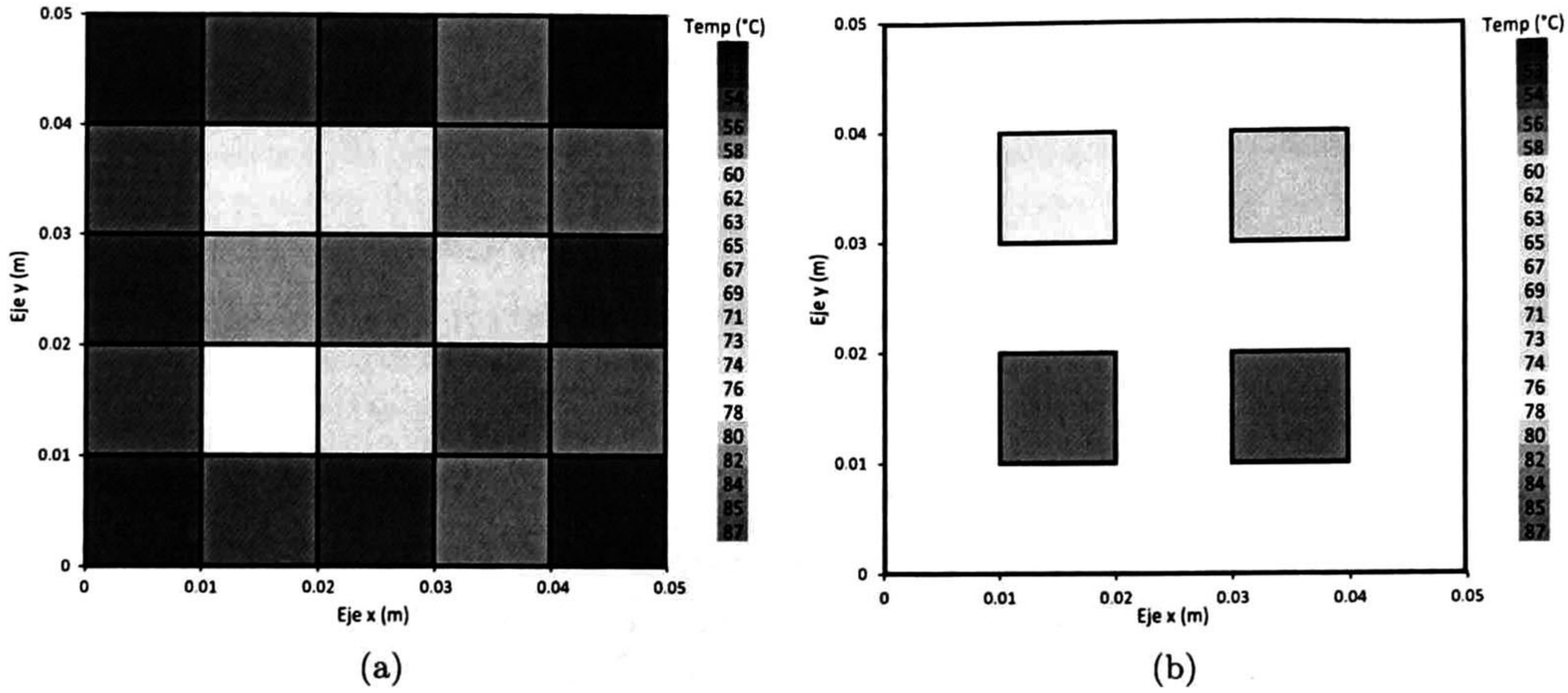


Figura 3.11: Distribución de Temperaturas en a)Placa y b)Chip con resolución de $1cm \times 1cm$

con la solución del sistema modelado en ANSYS como se muestra en las figuras Fig. 3.15 y Fig. 3.16. Note que la solución que se consigue de este modelo captura el comportamiento del retardo de transporte de calor hacia los otros procesadores, por lo que podemos considerar que este modelo es una muy buena aproximación del comportamiento térmico del sistema, más aún se puede apreciar en las figuras Fig. 3.17 y Fig. 3.18, que el error en estado estable es menor a 0.3^oC.

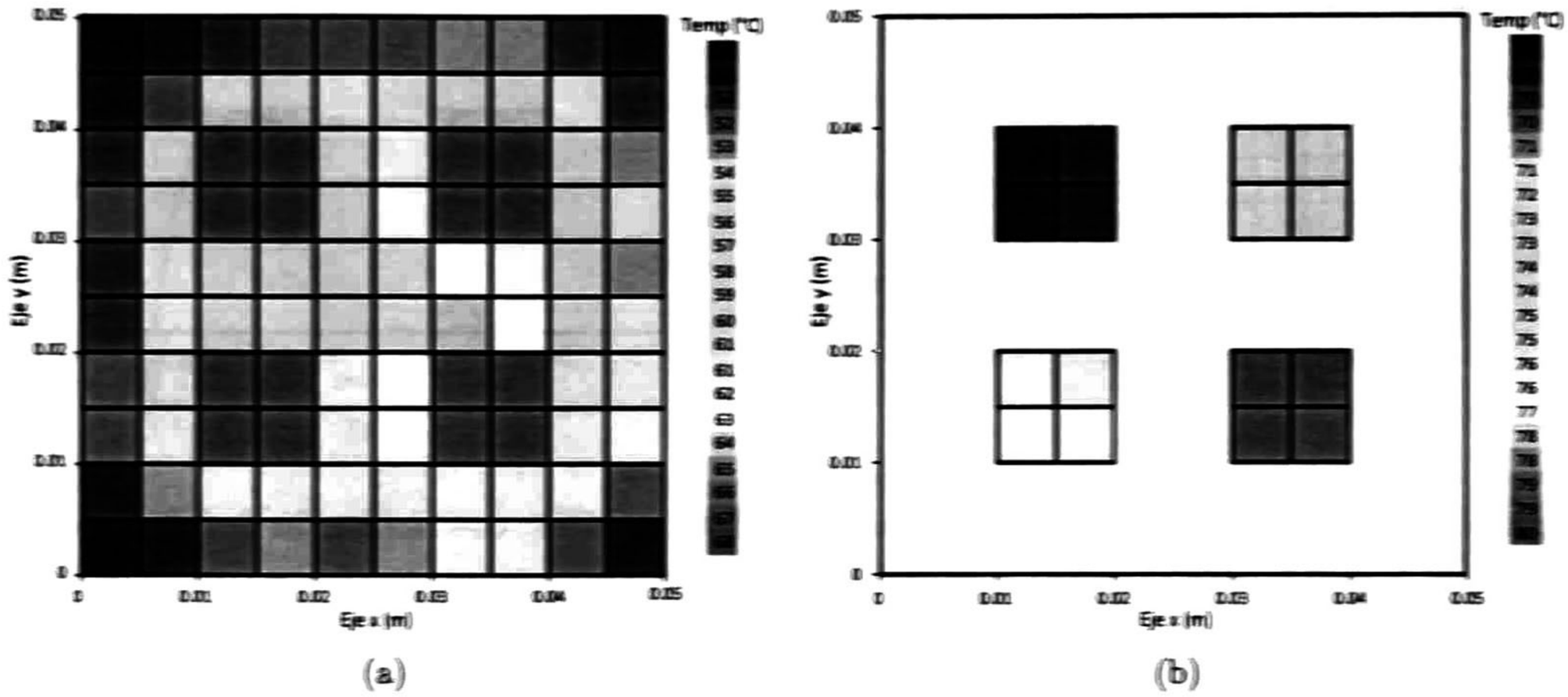


Figura 3.12: Distribución de Temperaturas en a)Placa y b)Chip con resolución de $0.5cm \times 0.5cm$

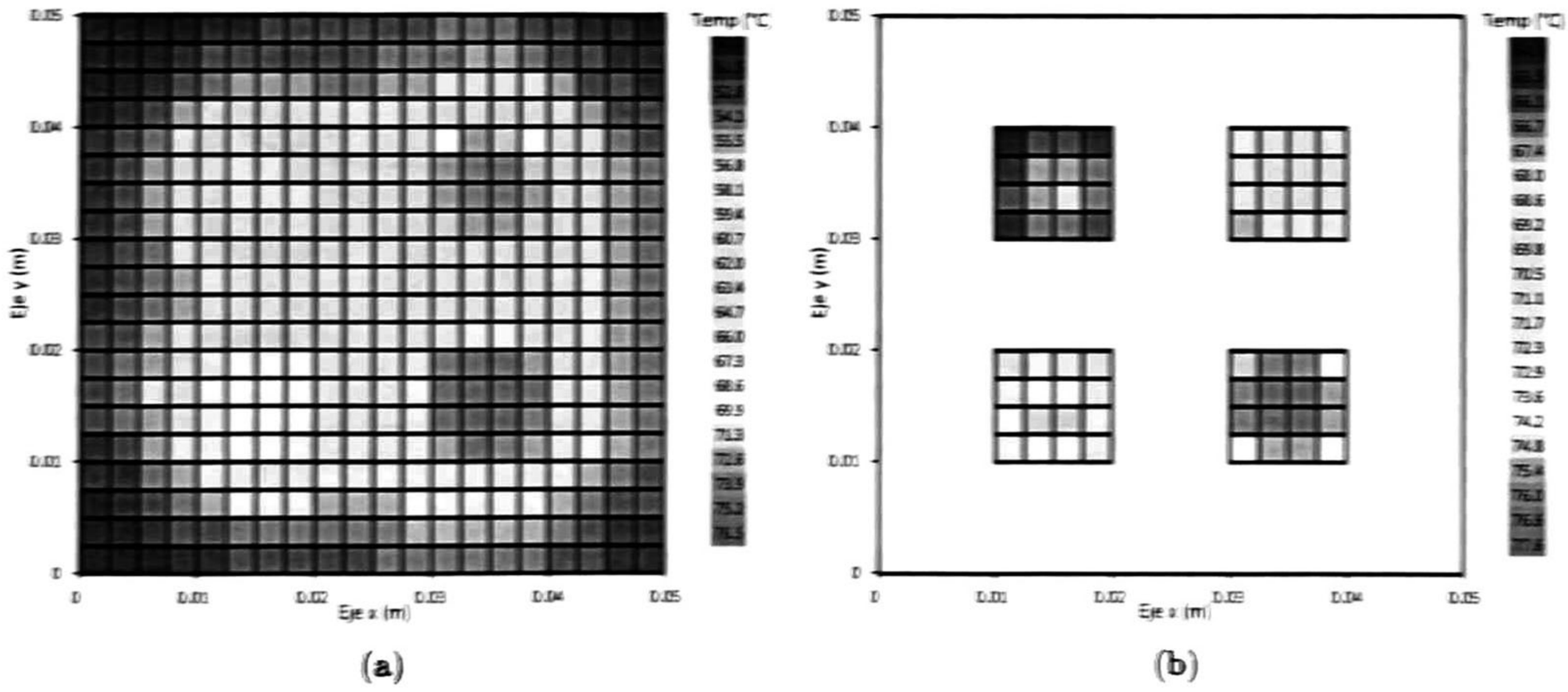
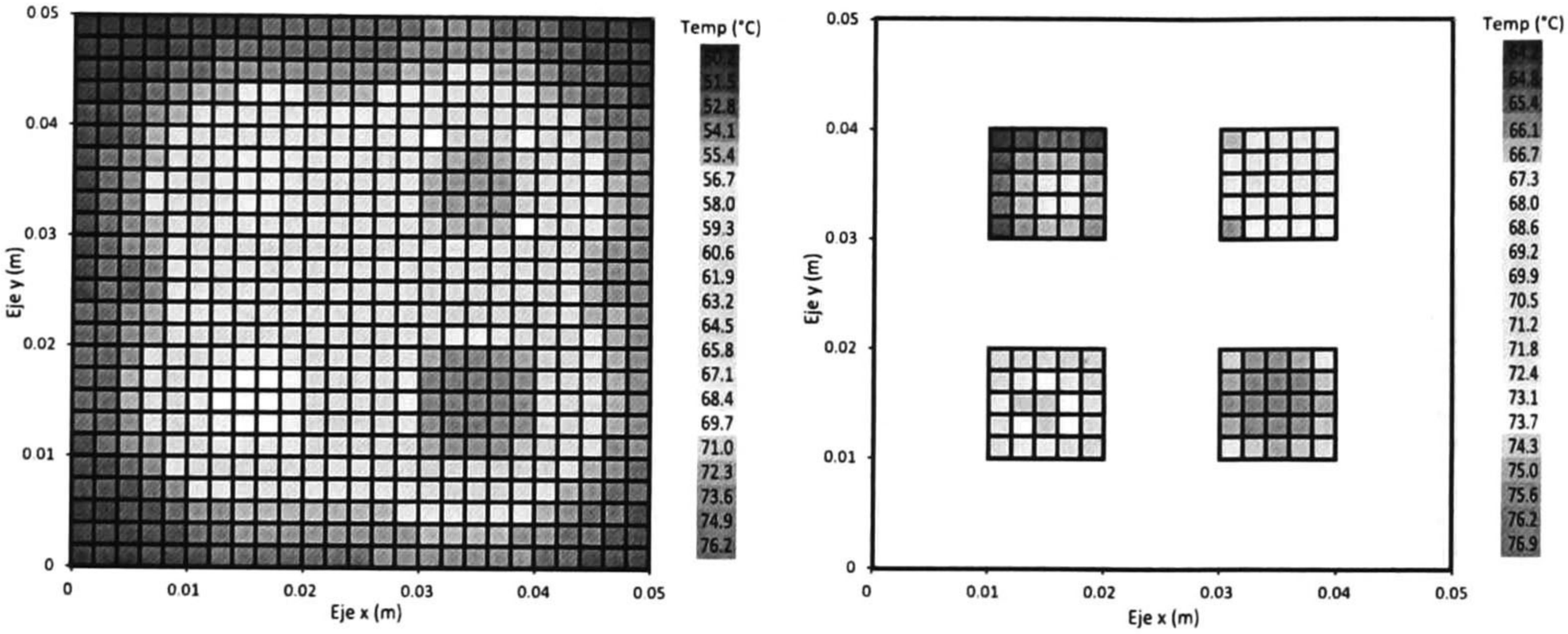


Figura 3.13: Distribución de Temperaturas en a)Placa y b)Chip con resolución de $0.25cm \times 0.5cm$



(a) (b)

Figura 3.14: Distribución de Temperaturas en a)Placa y b)Chip con resolución de $0.20cm \times 0.20cm$

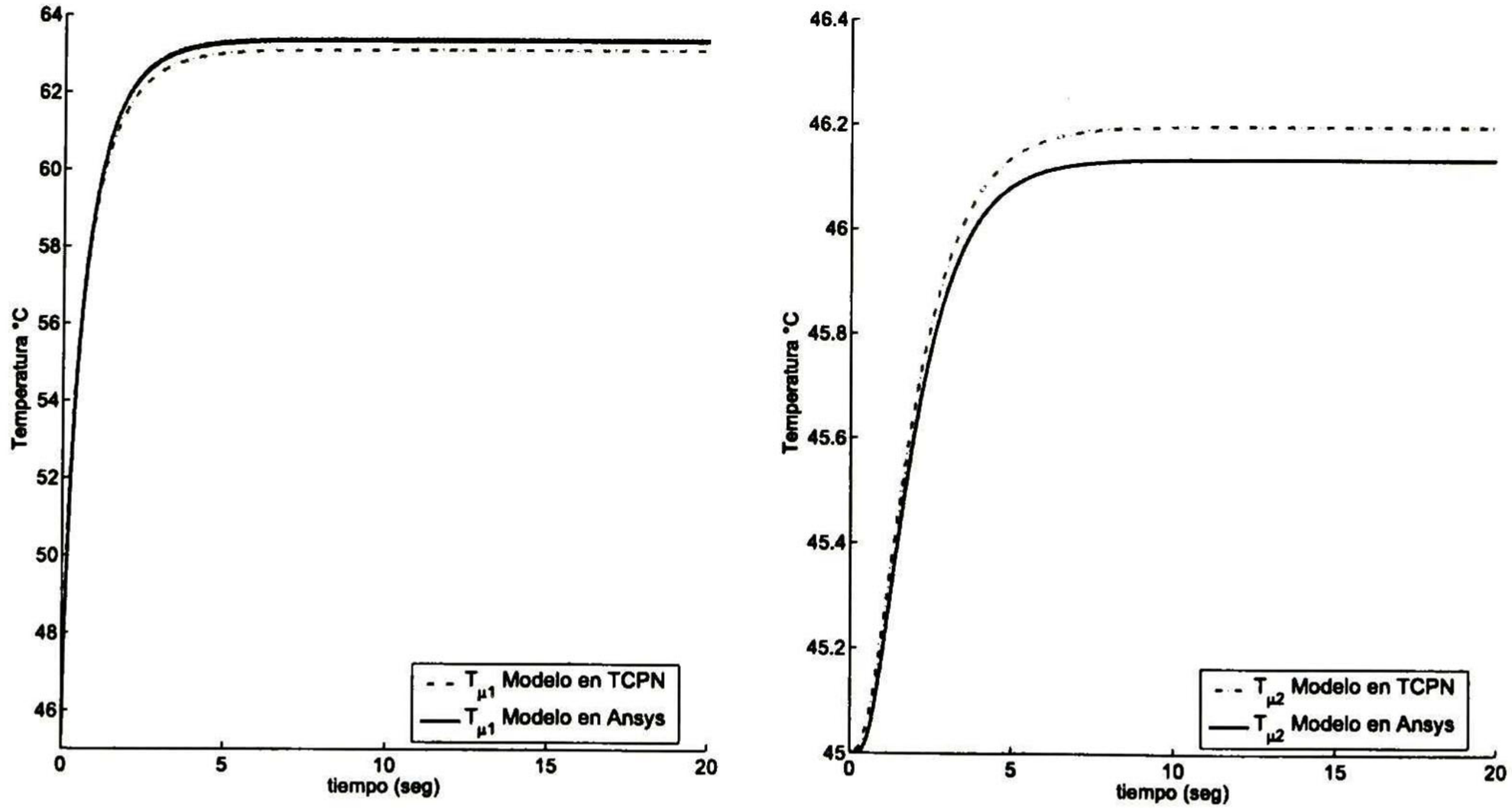


Figura 3.15: Comparación del modelo en TCPN y el modelo en Ansys

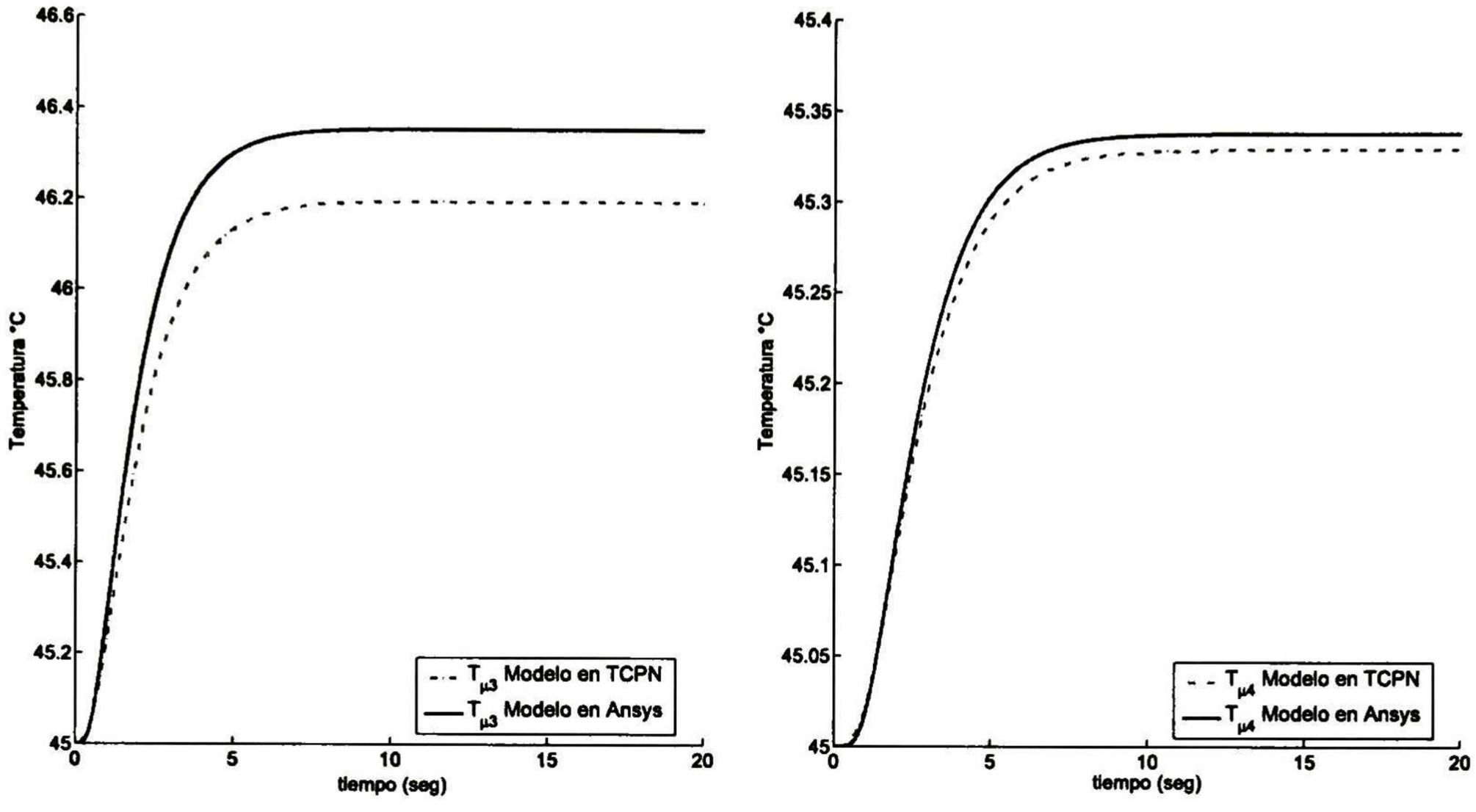


Figura 3.16: Comparación del modelo en TCPN y el modelo en Ansys

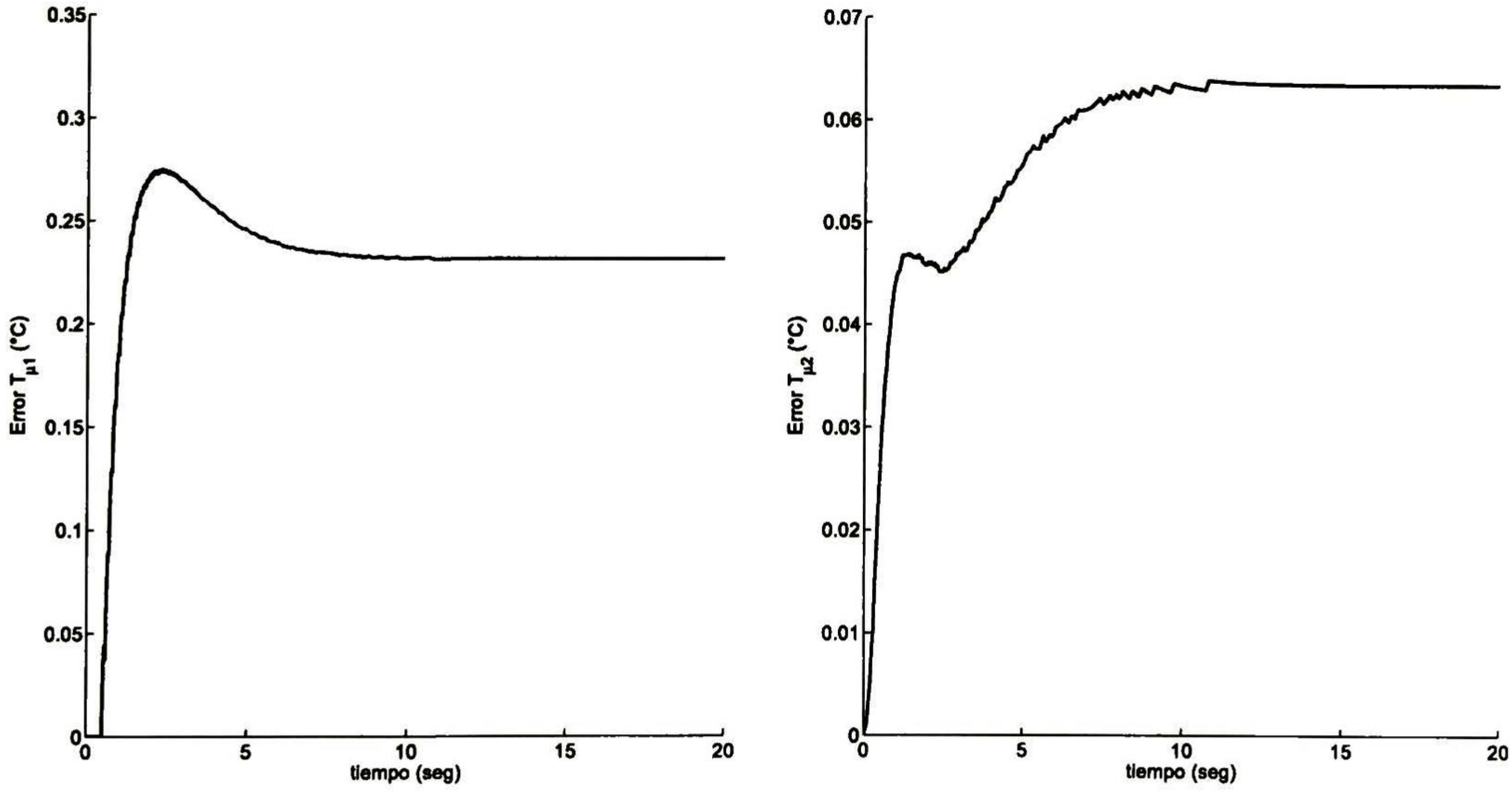


Figura 3.17: Error del modelo en TCPN y el modelo en Ansys procesador 1 y 2

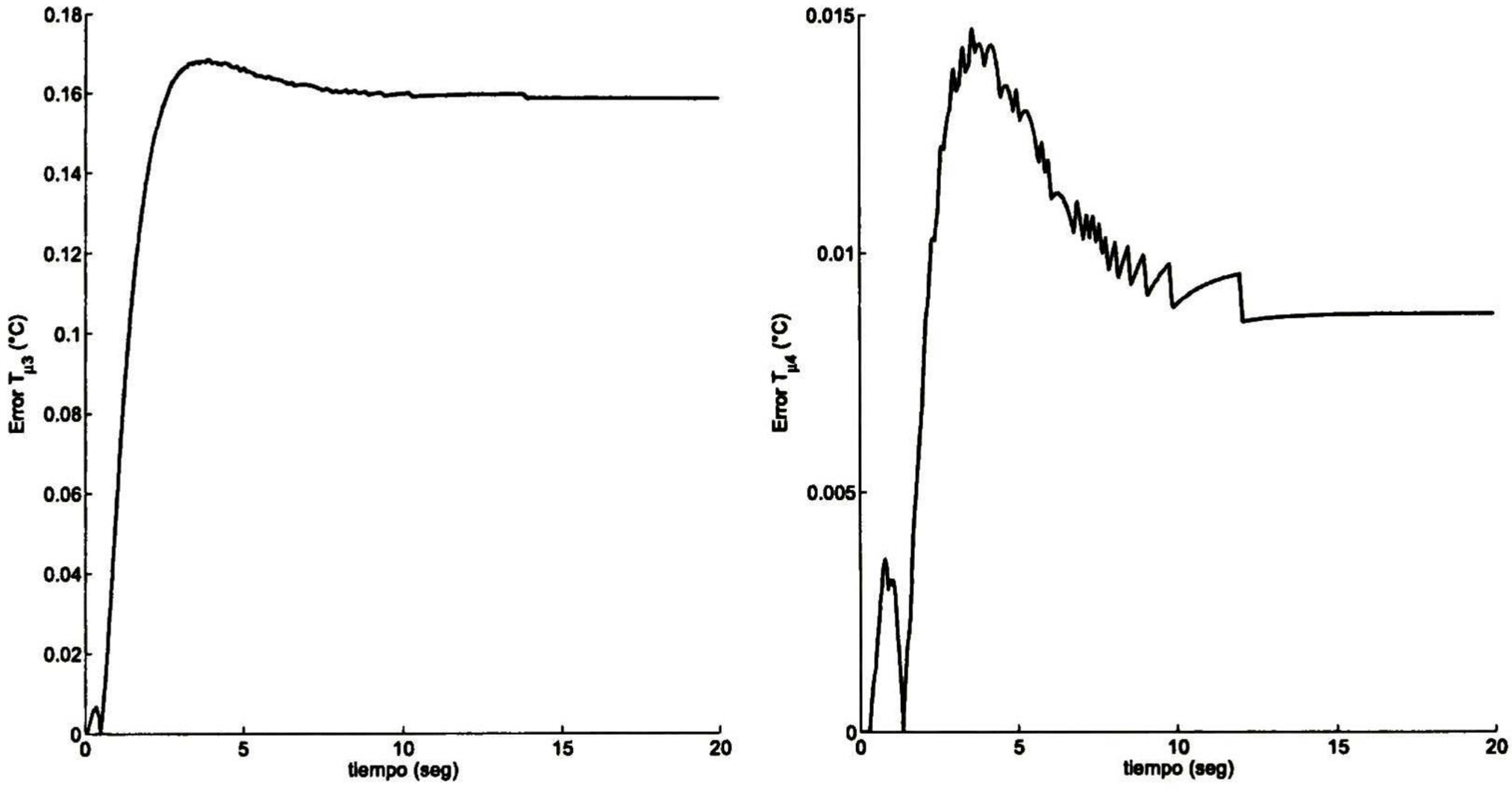


Figura 3.18: Error del modelo en TCPN y el modelo en Ansys procesador 3 y 4

Capítulo 4

Modelo de Tareas y Procesador

En este capítulo se presenta una metodología de modelado usando redes de Petri Continuas Temporizadas para un conjunto de tareas periódicas independientes con especificaciones temporales tales como fecha límite de entrega, tiempo de ejecución y periodo, que deben de ser asignadas y ejecutadas en un conjunto de procesadores idénticos.

4.1. Introducción

Los dispositivos modernos, como los dispositivos móviles, los juegos de video, telefonía celular, etc., actualmente están integrados por $MPSoC$'s, los cuales satisfacen las necesidades de procesamiento en estos sistemas. Debido a la gran cantidad de tareas que deben ejecutar estos dispositivos en tiempo real (por ejemplo: reproducción de música, vídeo, comunicaciones, localización GPS entre otros) se obtiene un incremento de temperatura en el $MPSoC$ y por consecuencia en los dispositivos que los contienen. Por lo tanto, la temperatura es una nueva restricción que también deben tomar en cuenta los schedulers modernos, por lo general la temperatura debe estar por debajo de un límite permitido para evitar problemas ergonómicos o problemas que afecten el rendimiento y la vida de uso del dispositivo. Además del problema térmico, el consumo de energía juega otro papel importante, ya que la carga de la batería debe ser administrada para durar el mayor tiempo posible mientras se ejecutan las de tareas.

Desafortunadamente, el consumo de energía y la temperatura en los $MPSoC$'s aumentan a medida que se ejecutan un mayor número de tareas, existiendo un compromiso entre el número de tareas ejecutadas, la temperatura y el tiempo de duración de la carga de la batería. Con el fin de garantizar que las tareas cumplan con sus limitaciones temporales y el $MPSoC$ cumpla con las limitaciones térmicas, actualmente se diseñan schedulers que hacen frente a estas las limitantes.

Una manera de conciliar el consumo de energía, temperatura y la cantidad de tareas que fluyen en un $MPSoC$ (*throughput*), consiste en encontrar una secuencia adecuada de ejecución de tareas (schedule). Sin embargo, el primer paso para lograr este objetivo es la construcción de un modelo adecuado para las tareas. Para la construcción de dicho modelo, proponemos la siguiente

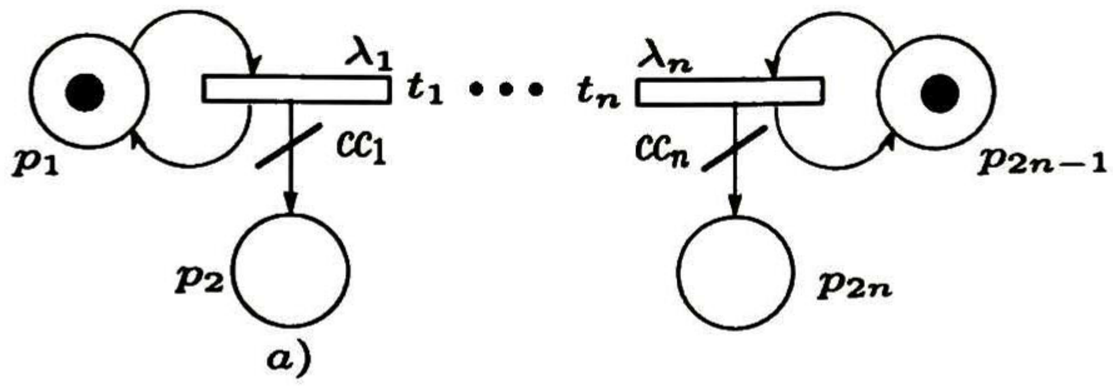


Figura 4.1: Módulo de red de Petri para n tareas periódicas e independientes.

metodología de modelado.

4.2. Modelo de Tareas

Considere un sistema de tiempo real donde la carga de trabajo es modelado como un conjunto de tareas periódicas e independientes $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ con restricciones de tiempo. Cada tarea tiene los siguiente parámetros:

- ω_j, representa el periodo de la tarea τ_j, se considera fijo y conocido e indica que la tarea debe ejecutarse cada ω_j unidades de tiempo.
- d_j, es la fecha límite de entrega (deadline) de la tarea τ_j. La $k-th$ ejecución de la tarea τ_j debe finalizar antes de $k \cdot \omega_j + d_j$.
- cc_j, representa la cantidad de ciclos de CPU (ciclos de CPU) que consume una tarea τ_j al ser ejecutada en algún procesador.

4.2.1. Metodología de modelado de Tareas

Una tarea independiente τ_j arriba al sistema cada ω_j y consume cc_j ciclos de CPU durante su ejecución. El modelado del arribo de la tarea τ_j cada ω_j unidades de tiempo se obtiene mediante el módulo en red de Petri que se muestra en a) en la Fig. 4.1. En esta representación la tasa de disparo $\lambda_j = \frac{1}{\omega_j}$ representa la generación de una nueva copia de la tarea τ_j cada ω_j unidades de tiempo. Añadiendo un peso cc_j al arco que va de la transición t_1 al lugar p_2 se tienen los ciclos de CPU que consume la tarea en cierto procesador. Para el caso de representar un conjunto finito de tareas τ, el modelo resulta en la creación de un módulo a) de la Fig. 4.1 por cada tarea.

Modelo de fecha límite de entrega (deadline)

Cada tarea τ_j está asociada a su fecha límite de entrega (deadline) d_j, éste parámetro indica que el $k-th$ disparo de τ_j debe finalizar en un tiempo ε_j, donde $k \cdot \omega_j \leq \varepsilon_j \leq k \cdot \omega_j + d_j$.

Para el modelo en $TCPN$, el número de tareas τ_j que arriban al sistema en el tiempo η_j esta dado por

$$\alpha_j = \int_0^{\eta_j} \frac{1}{\omega_j} dt, \tag{4.1}$$

el numero de instancias de tipo de tarea τ_j que fueron atendidas por m procesadores en el tiempo η_j es

$$\beta_j = \int_0^{\eta_j} \frac{1}{\lambda_{1,j} + \ldots + \lambda_{m,j}} dt, \tag{4.2}$$

entonces la diferencia de tareas que arriban y las tareas que son atendidas

$$\gamma_j(\eta_j) = \alpha_j(\eta_j) - \beta_j(\eta_j), \tag{4.3}$$

representa las instancias de tareas de tipo τ_j que no fueron atendidas. Por lo tanto, con el fin de cumplir con los tiempos límites de entrega, el $MPSoC$ debería ejecutar estas instancias de tarea que no son atendidas a una tasa de $\gamma_j(\eta_j)/d_j$, i.e

$$f_{1,j}(\eta_j) + \ldots + f_{q,j}(\eta_j) \geq \frac{\gamma_j(\eta_j)}{d_j} \tag{4.4}$$

donde $f_{i,j}(\eta_j) = \lambda_{1,j} min\{m(p_3), m(p_1)\}$ representan los flujos que pasan a través de las transiciones $t_{i,j}$ de la Fig. 4.2.

La ecuación (4.4) no puede ser representada mediante un módulo en $TCPN$. Sin embargo, esta fórmula puede usarse en algún scheduler ó controlador para asegurar que las tareas cumplan con sus respectivos deadlines.

4.3. Modelo del Procesador

Cuando se requiere obtener un modelo para un procesador es necesario establecer un criterio de medida que permita cuantificar el rendimiento de los procesadores. La unidad de medida por excelencia es el tiempo, es decir un procesador que ejecuta una tarea más rápidamente que otra es mejor si medimos el tiempo de respuesta de una tarea. Sin embargo, se puede considerar que para muchas tareas la medida de rendimiento es el *throughput*, este parámetro representa el número de tareas que se ejecutaron por unidad de tiempo. Por lo tanto, para modelar al procesador mediante el formalismo $TCPN$ por su carácter continuo, es ventajoso utilizar al *throughput* como medida de rendimiento.

Considere un MPSoC Φ que contiene un conjunto de procesadores $\varphi = \{q_1, q_2, q_3, \ldots, q_n\}$ con características idénticas de procesamiento. Un procesador $q_i \in \varphi$ puede ser visto como un servidor de

ciclos de CPU disponibles, el cual puede variar de acuerdo a la frecuencia de reloj Fc. La idea estriba en que estos ciclos contenidos en algún procesador q_i se reparten a algún conjunto de tareas finitas τ. La cantidad de ciclos en q_i puede ser calculado dependiendo de la arquitectura del procesador, para esta metodología de modelado es conveniente denotar como M_i a los ciclos de CPU disponibles cuando q_i trabaja a su máxima frecuencia, i.e M_i es el máximo *throughput* en ciclos de CPU de un procesador q_i. Mientras la frecuencia varía al ejecutar una tarea, también lo hace el número de ciclos de CPU disponibles por unidad de tiempo en el procesador, lo que implica también que el tiempo de ejecución de una tarea también varíe.

Un modelo obtenido con esta propuesta esta restringido por los siguientes puntos:

- Cada procesador $q_i \in \varphi$ puede cambiar dinámicamente su frecuencia y por consiguiente su voltaje. Se considera que se tienen κ frecuencias disponibles $S_1 < S_2 < S_3 < \ldots < S_\kappa$ para los procesadores en el $MPSoC$.
- Cuando el procesador q_i está apagado (i.e, no ejecuta ninguna tarea) la potencia consumida es cero. Cuando el procesador está encendido y ejecuta alguna tarea a una frecuencia S_k con $k = 1, \ldots, \kappa$, el procesador consume $P_i(S_k)$ potencia.

4.3.1. Metodología de modelado del procesador

En esta sección se explica la metodología propuesta, además se muestra cómo el modelo puede controlar el número de tareas que el $MPSoC$ ejecuta por unidad de tiempo *throughput*. La metodología propuesta esta basada en $TCPN$.

Modelo de Potencia

Calcular el consumo de potencia de un dispositivo electrónico necesita de una descripción muy detallada del dispositivo en cuestión, se necesita conocer los parámetros de los semiconductores que lo componen, además se necesita también, un conocimiento electrónico detallado de la micro-arquitectura. Sin embargo, esta información no siempre se encuentra disponible. Por ello, se opta en utilizar un modelo para el consumo de potencia.

Particularmente, en los dispositivos modernos los procesadores utilizan lógica CMOS. Una aproximación para un modelo de potencia de un procesador tipo CMOS está relacionado con la frecuencia S_k con la cuál trabaja. La potencia que genera un procesador q_i se denota como $P_i(S_k)$ y puede dividirse en dos partes: $P_{i_d}(S_k)$ y $P_{i_{ind}}(S_k)$, donde $P_{i_d}(S_k)$ es la potencia dinámica que surge de la carga y descarga de compuertas en el procesador q_i, mientras que la potencia $P_{i_{ind}}(S_k)$ corresponde a la potencia estática y principalmente resulta de la corriente de fuga. La función $P_{i_d}(S_k)$ puede modelarse como una función creciente convexa de la frecuencia, mientras $P_{i_{ind}}(S_k)$ se puede considerar constante [10].

En este documento, se consideran procesadores tipo CMOS, en donde la potencia que genera un procesador q_i es dominada por la potencia dinámica $P_{i_d}(S_k)$ como una función polinomial de

S_k. Por lo que, la potencia de un procesador q_i con lógica CMOS puede modelarse por medio de la función siguiente:

$$P_i(S_k) = \alpha S_k^{\imath} + \Gamma \tag{4.5}$$

para cualquier $0 < \imath \leq 3$, con $\alpha, \Gamma \geq 0$ [37].

Esta ecuación no puede modelarse en una rede de petri, sin embargo se puede incluir como el flujo de la transición t_3 de la Fig. 4.2 el cual representa el calor generado por el procesador al ejecutar una tarea.

Modelo del throughput en el Procesador

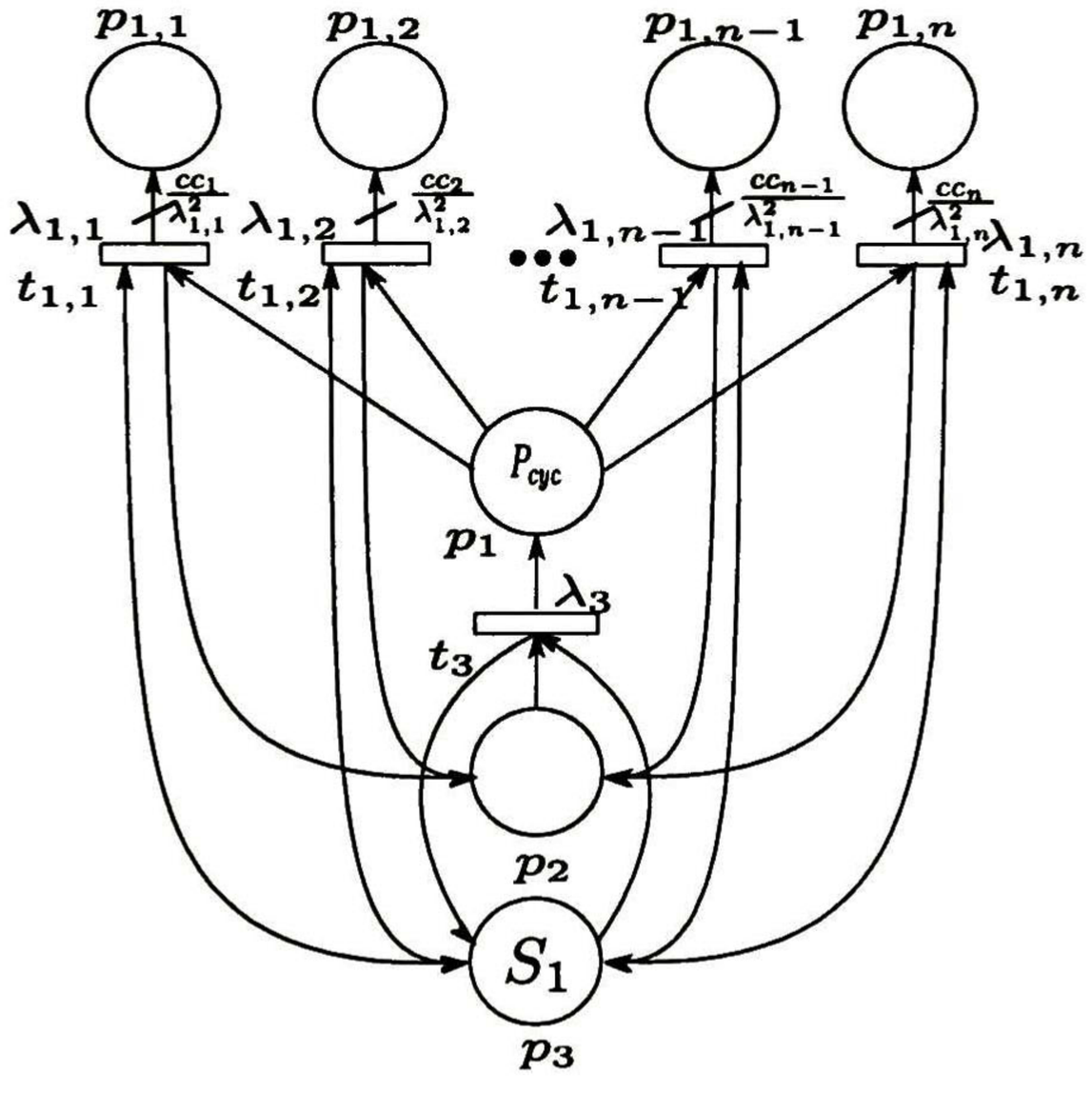


Figura 4.2: Módulo de red de Petri para la ejecución de un conjunto de tareas τ en un sólo procesador q_1.

La Fig. 4.3 muestra el modelo del *throughput* de un procesador. El disparo de la transición $t_{1,j}$ representa el hecho de $\lambda_{1,j}$ ciclos de CPU de la tarea τ_j por segundo se asignan al procesador q_1 con $\lambda_{1,j} \leq cc_j$. El marcado del lugar p_1 modela el número de ciclos de CPU disponibles por unidad de tiempo. El marcado inicial de p_1 es justamente el máximo *throughput* M_1. En la misma figura, el marcado en p_2 representa la carga de ciclos en el procesador de acuerdo a la asignación de una tarea al procesador q_1. La transición t_3 representa la ejecución de las tareas por q_1.

Cuando una tarea τ_j se asigna a un procesador q_1, la transición $t_{1,j}$ se dispara y los ciclos de CPU correspondientes a la tarea se mantienen en p_2 para que el procesador q_1 ejecute la tarea. Una vez que la tarea fue ejecutada, los ciclos de CPU se restauran en el lugar p_1, indicando que q_1 libera el número ciclos de CPU de la tarea ejecutada, por lo que ahora esos ciclos están disponibles para ejecutar una nueva tarea.

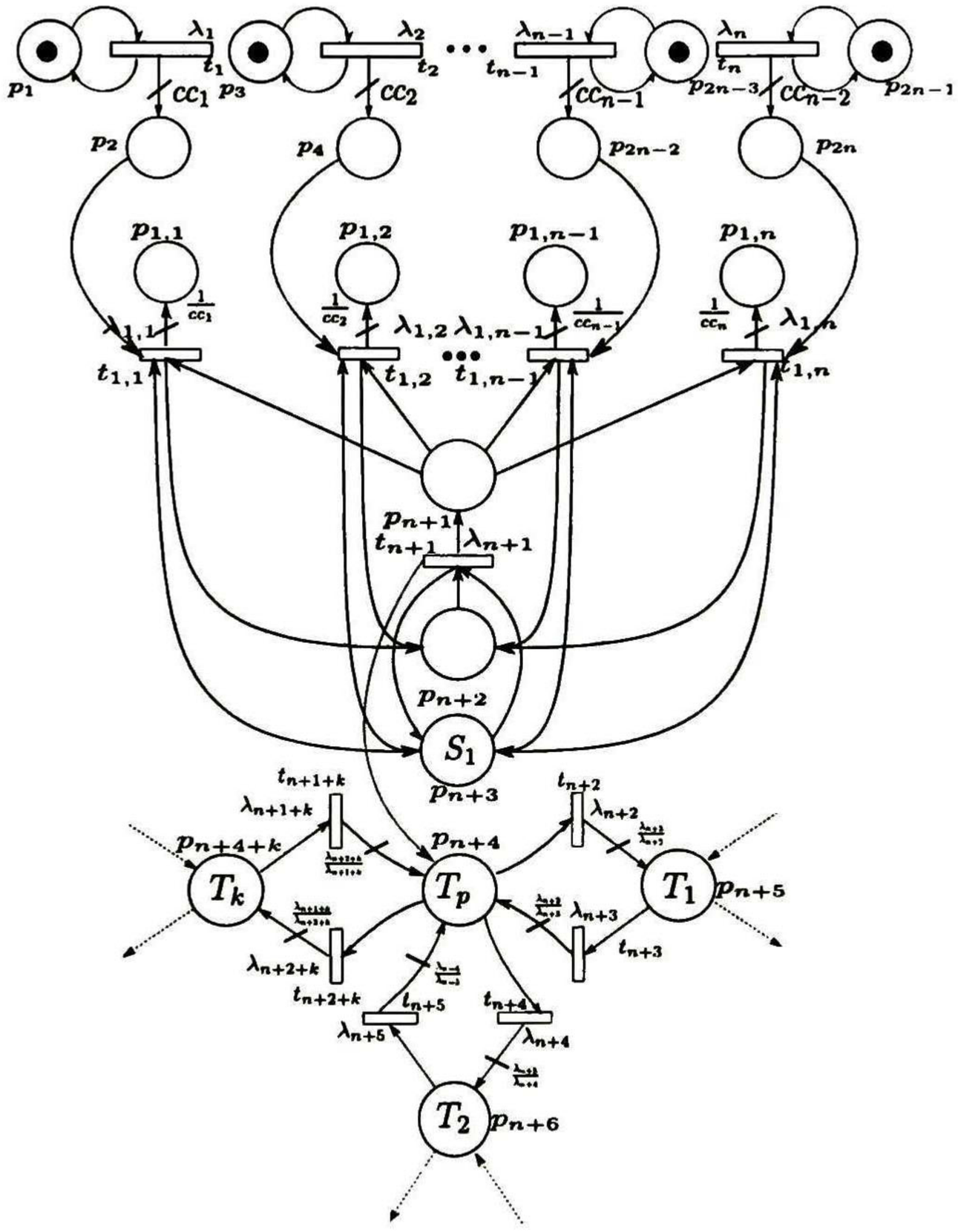


Figura 4.3: Modelo de tareas, procesador y térmico en $TCPN$.

Por lo tanto, al modelar al procesador como un servidor de ciclos de CPU el cual varía su frecuencia de reloj o voltaje, es equivalente a variar el *throughput* (asignar mayor o menor número de ciclos por unidad de tiempo de cada tarea) en el procesador. Este *throughput* (flujo de ciclos de CPU) es modelado por las transiciones $t_{1,j}$ en la Fig. 4.2.

Se añade un lugar p_3 de la misma figura, con el fin de modelar por separado la frecuencia del

procesador como una restricción adicional. Se calcula la tasa de disparo de las transiciones $t_{1,j}$ para cada posible frecuencia del procesador, y se establece el marcado inicial en el lugar p_3 de acuerdo a cada caso. Se prefiere incluir la frecuencia como el marcado en el lugar p_3 puesto que en un MPSoC real, el número de frecuencias posibles de trabajo es finito.

4.4. Integración del modelo térmico, tareas y procesador

Una representación de la asignación y ejecución de n tareas en un procesador se obtiene mediante la unión de los lugares $p_2, \ldots, p_{2n}$ del modelo de tareas (Fig. 4.1) hacia las transiciones $t_{1,1}, \ldots, t_{1,n}$ del modelo del procesador (Fig. 4.2). Este modelo puede usarse para sustituir el módulo de generación de calor (Fig. 3.5) utilizado anteriormente en el modelo térmico (Fig. 3.7), añadiendo el arco de la transición t_3 en la Fig. 4.2 al lugar p_1 de la Fig. 3.7. Este arco representa el calor generado cuando las tareas son ejecutadas por el procesador. La Fig. 4.3 representa la unión de dicho arco al modelo térmico.

La integración e interacción de los modelos en $TCPN$ (térmico, de tareas y procesador) para n tareas asignados a m procesadores se obtiene mediante el modelo global en $TCPN$ de la Fig. 4.4, la ecuación fundamental que rige la evolución dinámica del marcado está dada como

$$\begin{aligned} \dot{m} &= C\Lambda\Pi(m)m - \hat{C}u \\ 0 &\geq u \geq \Lambda\Pi(m)m \end{aligned} \tag{4.6}$$

donde $\hat{C}$ contiene las columnas de C correspondientes a las transiciones $t_{i,j}$ en la Fig. 4.4.

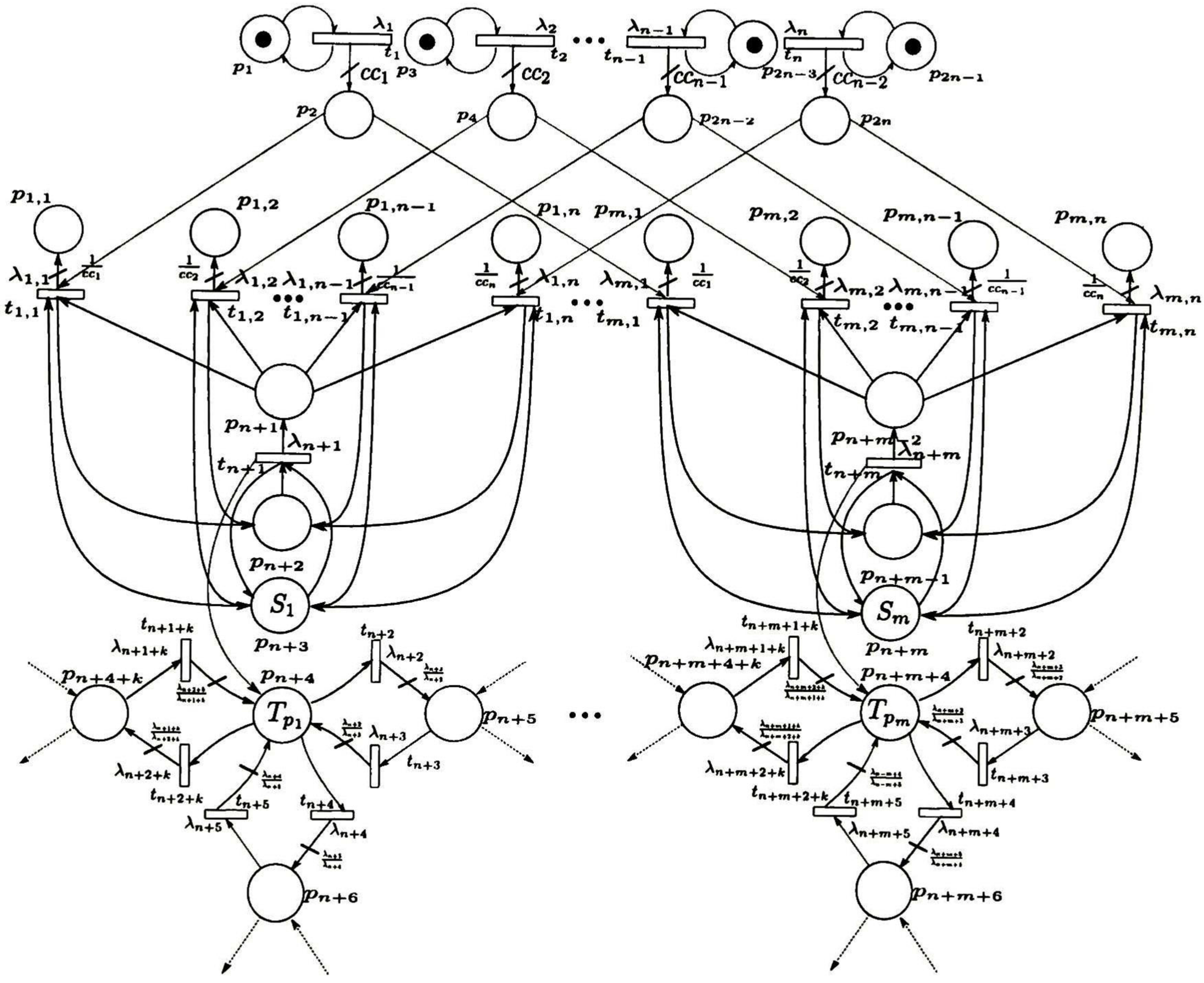


Figura 4.4: Modelo en $TCPN$ de asignación y ejecución de tareas en sistema Multiprocesador.

Capítulo 5

Schedulers Propuestos

Con el fin de mostrar la utilidad de los modelos propuestos anteriormente, en este capítulo se proponen dos tipos de schedulers. El primero, muestra el enfoque de control predictivo (MPC) usando el modelo térmico del capítulo 3. El segundo se basa en un scheduler continuo usando el modelo global en TCPN.

5.1. Definición del Problema

El problema a resolver es el siguiente:

Definición 5.1 *Sea Φ un MPSoC y τ un conjunto de tareas independientes y periódicas, en el que el tiempo de ejecución y tiempo límite de entrega son conocidos. Suponiendo que todas las tareas tienen la misma prioridad. El problema del scheduler térmico consiste en la asignación de tareas a procesadores de modo que el número de tareas desatendidas sea mínima y la temperatura en los procesadores se mantenga debajo de un cierto nivel conocido.*

En este trabajo se considera que el tiempo de ejecución, el deadline y el periodo son deterministas. Para los schedulers propuestos, se adopta el MPSoC del ejemplo 3.2 como caso de estudio. Aquí, la potencia consumida puede modelarse como una función cúbica de la frecuencia más una constante: $P(S_k) = (1.52S_k^3 + 0.08)\,Watt$ [37]. Las frecuencias disponibles (S_k) en GHz son $0.15, 0.4, 0.6, 0.8$, y 1. La potencia consumida y la frecuencia para este $MPSoC$ se encuentra en la Tabla 5.1.

Tabla 5.1: Frecuencia y Potencia consumida para un $MPSoC$

Frecuencia (GHz)	0.15	0.4	0.6	0.8	1
Potencia (Watt)	0.08	0.17	0.40	0.90	1.6

5.1.1. MPC scheduler

El algoritmo de control predictivo (MPC por sus siglas en inglés) es una estrategia de control que por cada paso de tiempo simula al sistema con diferentes acciones de control en un horizonte finito de tiempo. El MPC basa su funcionamiento en la predicción de cómo será el comportamiento del sistema en un futuro cercano ante varias entradas de control y escoge la entrada de control que minimice o en su caso maximice una función de costo.

Ecuación algebraica de Temperatura

El modelo térmico presentado en el Capítulo 3 es un sistema de ecuaciones diferenciales, el cual contiene tantas ecuaciones diferenciales como cantidad de componentes en el modelo. Por lo tanto al tener una alta granularidad (i.e, muchos componentes), encontrar la solución a estas ecuaciones consumiría un tiempo muy alto. Sin embargo, el modelo que se obtuvo no presenta sincronizaciones en sus transiciones, por lo que sólo se tiene una configuración en la red. Esto significa que tenemos ecuaciones lineales positivas las cuales pueden reescribirse como un sistema de ecuaciones lineales clásica, donde la solución de éstas es conocida.

Entonces, denotando a T_T, T_g y T_a como las matrices que representan la proyección natural de los lugares P de la red $TCPN$ del modelo térmico (Fig. 3.7) hacia los lugares que representa la temperatura $P_T(p_1, \ldots, p_k)$, los lugares de generación de calor P_g (como p_{k+1}) y la temperatura del aire P_a (como p_{k+2}), respectivamente. El modelo térmico en $TCPN$ puede reescribirse como:

$$\begin{aligned} \dot{m}_T &= T_T C\Lambda\Pi T_T^T m_T + T_T C\Lambda\Pi T_g^T m_g + T_T C\Lambda\Pi T_a^T m_a \\ &= Am_T + Bu \\ \dot{m}_g &= 0 \\ \dot{m}_a &= 0 \\ y &= \mathcal{S}m_T \end{aligned} \tag{5.1}$$

donde m_T, m_g y m_a representan los marcados en los lugares P_T, P_g, P_a, respectivamente; $A = T_T C\Lambda\Pi T_T^T$, $B = [T_T C\Lambda\Pi T_g^T, T_T C\Lambda\Pi T_a^T]$, $u = [m_g, m_a]^T$ pueden verse como las entradas al sistema y y es la temperatura de algunos componentes en el $MPSoC$ (Salidas del sistema).

Ahora, para este problema se considera que el scheduler ajusta la generación de calor cuando una nueva instancia de tarea arriba o cuando una instancia de tarea ha sido procesada, por lo tanto la generación de calor es constante a trozos. Denotando como δ_0, δ_1, ..., δ_k las instancias de tiempo en la cual la generación de calor se ajusta por el scheduler, la salida del sistema (5.1) puede obtenerse de la solución general de los sistemas lineales mediante dos referencias de tiempo δ_{k-1} and δ_k, como sigue:

$$y(\delta_k) = \mathcal{S}e^{A\delta_k} m_T(\delta_{k-1}) + \mathcal{S}\int_{\delta_{k-1}}^{\delta_k} e^{A(\delta_k - \eta)} Bu(\eta)d\eta \tag{5.2}$$

Mientras u es constante en $[\delta_{k-1}, \delta_k]$, entonces,

$$y(\delta_k) = \mathcal{S}e^{A\delta_k} m_T(\delta_{k-1}) + \mathcal{S}\int_{\delta_{k-1}}^{\delta_k} e^{A(\delta_k-\eta)} B d\eta \cdot u(\delta_k) \tag{5.3}$$

Por otra parte, si la temperatura alcanza el estado estable en el intervalo previo $[\delta_{k-2}, \delta_{k-1}]$, entonces $0 = Am_{k-1} + Bu(\delta_{k-1})$, por lo tanto $m_T(\delta_{k-1}) = A^{-1}Bu(\delta_{k-1})$, donde A^{-1} es la seudoinversa de A (A pierde rango por el auto lazo de los lugares del modelo térmico en $TCPN$ en la Fig. (3.7) que representan los anuladores derechos).

Finalmente, definiendo $G_0(\delta_k) = \mathcal{S}e^{A\delta_k}A^{-1}B$ y $G_u(\delta_k) = \mathcal{S}\int_{\delta_{k-1}}^{\delta_k} e^{A(\delta_k-\eta)}Bd\eta$, la ecuación (5.3) se reescribe como sigue:

$$y(\delta_k) = G_0(\delta_k)u(\delta_{k-1}) + G_u(\delta_{k-1})u(\delta_k) \tag{5.4}$$

$G_0(\delta_k)$ y $G_u(\delta_{k-1})$ son matrices pequeñas que son de dimensión de numero de salidas × numero de entradas en el sistema. Más aún, las entradas $G_0(\delta_k)$ y $G_u(\delta_{k-1})$ pueden aproximarse por funciones lineales en lapsos de tiempo que son calculadas fuera de línea.

Entonces, usando la ecuación lineal algebraica (5.4) y propiedades para sistemas lineales (i.e, superposición), La temperatura para un componente $y(\delta)$, que se obtiene por la ejecución de algún schedule $u(\delta)$, puede calcularse eficientemente ahorrando tiempo computacional en la ejecución del algoritmo MPC.

Criterio de Optimización

A fin de obtener la mejor asignación de tareas a los procesadores, diferentes acciones de control (schedules) son probados en el sistema. Por cada asignación de tareas $\hbar_k$ a los procesadores, la temperatura en éstos y el retraso de cada tarea τ_j (*lateness* L_j) son calculados. Después, se escoge el *mejor* schedule que minimiza la siguiente funcional:

$$J(\hbar_k, N) = J_T(\hbar_k, N) + J_L(\hbar_k, N) \tag{5.5}$$

donde $J_T(\hbar_k, N)$ representa la sumatoria del error de temperatura ($e_{T_{cpu}}$) de todos los procesadores, definido como la diferencia entre la temperatura predecida después de un horizonte de tiempo N y la temperatura máxima permitida para cada procesador; $J_L(\hbar_k, N)$ denota la suma de los *lateness* (L_j) de cada tarea τ_j que se asignan a los procesadores. El *lateness* está definido como el tiempo adicional que toma una tarea τ_j después de su fecha límite (d_j) para terminar su ejecución. El *lateness* es cero si la tarea no requiere de tiempo extra. Las sumatorias se ponderan para mejorar la importancia de alguna de las dos variables.

$$\begin{aligned} J_T(\hbar_k, N) &= \Sigma_{j=1}^{N} e_{T_{cpu}}^T \cdot Q \cdot e_{T_{cpu}} \\ J_L(\hbar_k, N) &= \Sigma_{j=1}^{N} L^T \quad R \cdot L \end{aligned} \tag{5.6}$$

donde Q y R son matrices definidas positivas.

Para ilustrar el comportamiento del scheduler mediante el algoritmo MPC, considere el siguiente ejemplo.

Ejemplo 5.1 Tomando como caso de estudio al modelo del $MPSoC$ del Ejemplo 3.2 y considerando que se tiene un conjunto de tareas $\tau = \{\tau_1, \tau_2\}$ periódicas e independientes con las siguientes propiedades:

τ	cc_j	d_j	ω_j
τ_1	10	2	1
τ_2	20	2	2

Se obtiene un schedule mediante el uso del algoritmo MPC propuesto. Las Fig. 5.1 y Fig. 5.2 respectivamente muestran la asignación de las tareas y la temperatura de los procesadores.

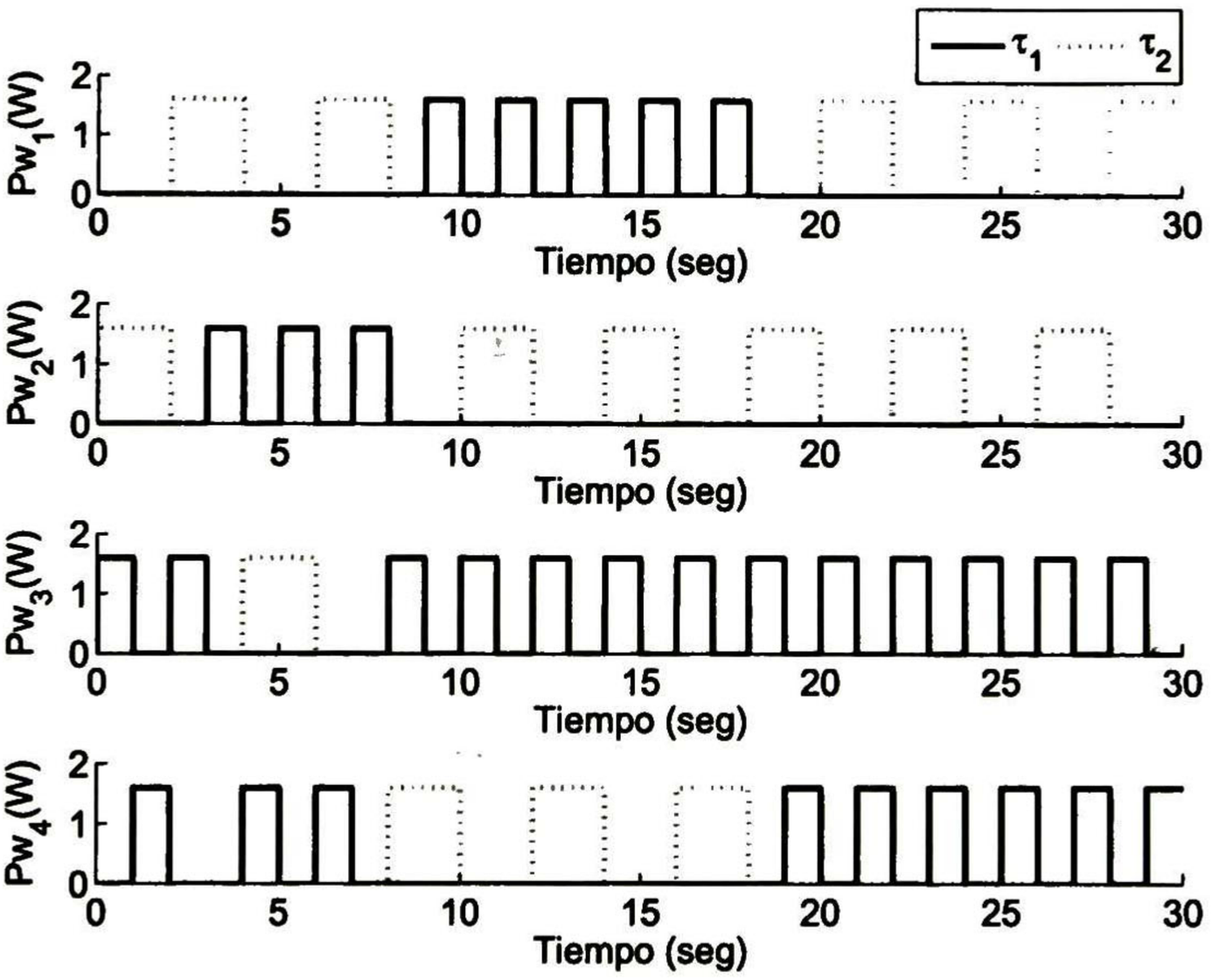


Figura 5.1: Asignación de tareas mediante MPC

En este ejemplo , el horizonte de predicción y de control son iguales al período de la tarea más corta calculada a partir del conjunto de tareas τ.

Durante estos horizontes todos los posibles schedules son generados y evaluados (con $Q = I$ y $R = 2 \cdot I$). El schedule $\hbar_k$ con el mínimo $J(\hbar_k, N)$ se selecciona y se ejecuta en el conjunto de procesadores φ. Después, el algoritmo se repite nuevamente para encontrar el nuevo schedule minimice la función de costo (ecuación (5.5)).

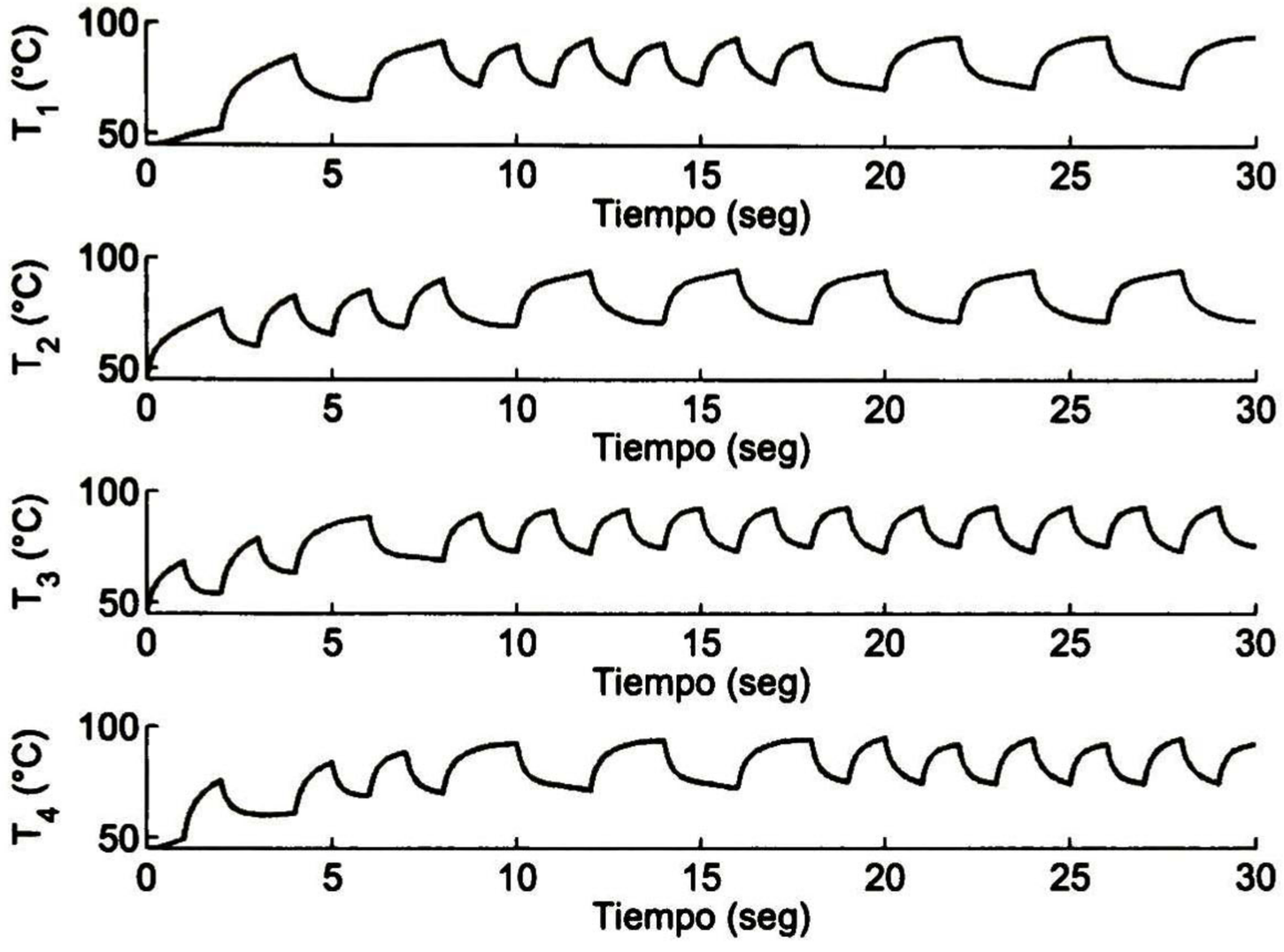


Figura 5.2: Temperatura de cuatro procesadores resultado de la asignación de tareas mediante MPC

5.1.2. Scheduling TCPN

En esta sección se presenta un scheduler continuo usando el modelo global en $TCPN$ del Capítulo 4. El scheduler que se presenta esta compuesto por dos partes.

Primeramente, se propone un scheduler de tiempo continuo que asigna un conjunto de tareas τ a un conjunto de procesadores φ. Este enfoque consiste en asignar tareas a los procesadores por medio del flujo en las transiciones $t_{i,j}$ (que permiten o bloquean el disparo de las transiciones $t_{i,j}$ en la Fig. 4.2, es decir aquellas transiciones que asignan alguna tarea τ_j al procesador q_i), la elección depende de una ecuación que modela qué tareas no cumplen con su fecha límite a las cuales se le asigna una mayor prioridad para ser ejecutadas.

En la segunda parte, se implementa un control proporcional para mantener la temperatura de los procesadores debajo de un límite requerido T_u. La temperatura es controlada seleccionando la adecuada frecuencia y reduciendo el *throughput* en los procesadores (i.e reduciendo flujo de las transiciones $t_{i,j}$ de la Fig. 4.2). Este control garantiza que la temperatura se mantenga debajo de un valor conocido, sin embargo no se garantiza que las restricciones temporales se cumplan.

La Fig. 5.3 muestra la el esquema de control para este enfoque.

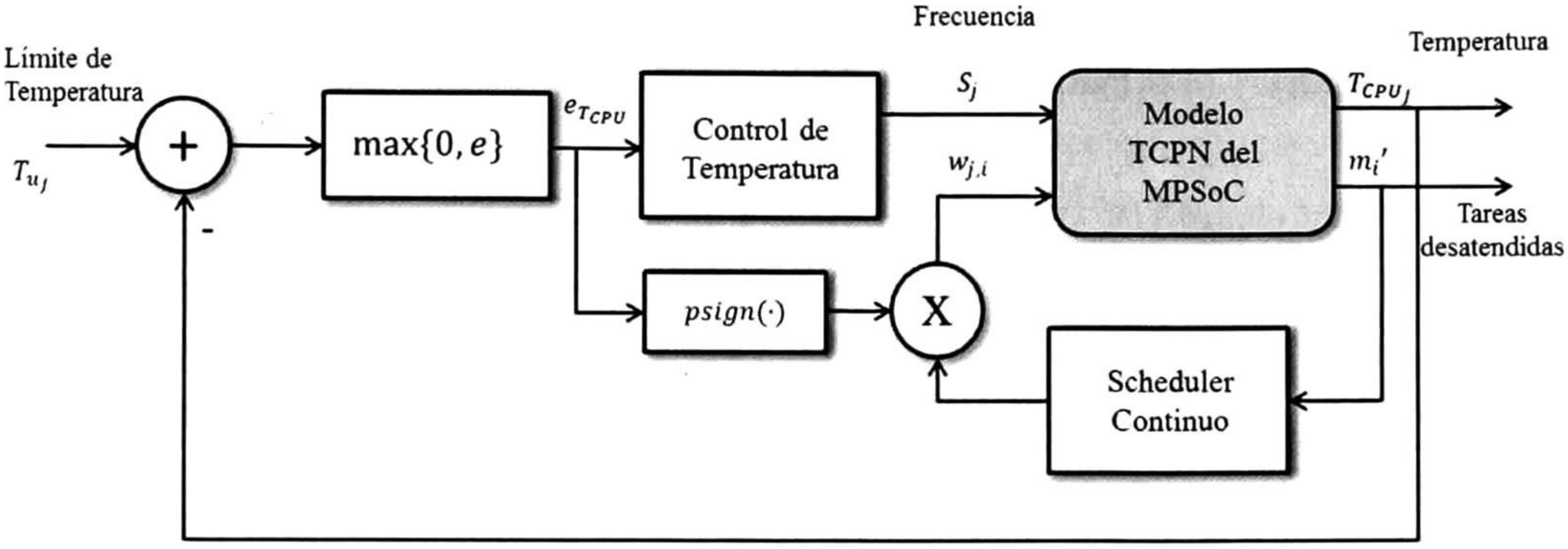


Figura 5.3: Esquema de control para el scheduler continuo.

Scheduler en tiempo continuo

Suponiendo que la temperatura y el *throughput* pueden medirse, podemos obtener un *scheduler continuo* para el control de temperatura, donde las variables a manipular son la frecuencia del procesador y los flujos en las transiciones $t_{i,j}$ (*throughput* del procesador q_i).

De acuerdo con la ecuación (5.7) se puede calcular la cantidad de tareas que no cumplen con sus restricciones de temporales. Por lo tanto, si $\frac{\gamma_j(\eta)}{d_j} - \sum_{i=1}^{m} f_{i,j}(\eta) > 0$ (m es el número de procesadores y j es el indice de cada tarea) entonces, en el tiempo η, mas tareas por unidad de tiempo son ejecutadas en comparación con la cantidad de tareas que arriban; de otra manera, si $\frac{\gamma_j(\eta)}{d_j} - \sum_{i=1}^{m} f_{i,j}(\eta) < 0$ entonces, en el tiempo η, un menor número de tareas por unidad de tiempo son ejecutadas en comparación con las tareas que arriban. Por lo tanto, las tareas que no son atendidas del tipo τ_j, denotadas como m'_j son calculadas resolviendo para cualquier tiempo η la siguiente ecuación:

$$\begin{aligned} \dot{m'}_j = max(0, \tfrac{\gamma_j(\eta)}{d_j} - \textstyle\sum_{i=1}^{m} f_{i,j}(\eta)) \\ -max(\textstyle\sum_{i=1}^{m} f_{i,j}(\eta) - \tfrac{\gamma_i(\eta)}{d_j}, 0). \end{aligned} \tag{5.7}$$

Dependiendo del valor anterior, se propone un control que reduce el flujo en las transiciones $t_{i,j}$ con la finalidad de que las tareas terminen antes de su fecha de entrega.

Para resolver el problema de asignación en este modelo, $e_{\tau_j} = m'_j$ se le nombra como el *error de tareas*, el cual indica el número de instancias de tarea de tipo τ_j que no cumplen con sus fechas límites de entrega.

Basado en el error de tareas (ecuación (5.7)), se propone una acción de control que es calculada y aplicada al modelo global en $TCPN$ para la asignación (Fig. 4.2) reduciendo la velocidad de disparo de las transiciones $t_{i,j}$ como sigue.

$$u_{i,j} = psign(e_{\tau j}) f_{i,j} \tag{5.8}$$

donde

$$psign(x) = \begin{cases} 0 & if \quad x > 0 \\ 1 & if \quad x \leq 0 \end{cases} \tag{5.9}$$

los subíndices $i \in \{1, \ldots, m\}$ se refieren a la $i - th$ procesador y el subíndice $j \in \{1, \ldots, n\}$ se refiere al $j - th$ tarea (en este caso no se usa la clásica función *sign*, en su lugar se usa una función denotada como *psign*, esto se debe a que una acción de control negativo no se permite en las $TCPN$s).

Un error positivo de tareas e_{τ_j} significa que tenemos tareas desatendidas, por lo que el control (5.8) activa las transiciones $t_{i,j}$ con el fin de asignar instancias de tareas a los procesadores; de lo contrario, si e_{τ_j} es negativo ó cero, significa que los procesadores están ejecutando instancias de tareas τ_j a tiempo y el control bloquea las transiciones $t_{i,j}$ (i.e., deja de asignar tareas para ahorrar energía).

Control de Temperatura

Con el propósito de obtener un buen manejo de la temperatura en el $MPSoC$, se propone un control para este propósito. Con este enfoque, la temperatura es controlada mediante una acción de control proporcional (dependiente del error de temperatura) que selecciona una frecuencia adecuada de trabajo para cada procesador. El objetivo de este control es mantener la temperatura por debajo de un cierto valor de temperatura predefinido, por lo que el error de temperatura se calcula como sigue:

$$e_{T_{CPU_i}} = \text{máx}\{0, T_{CPU_i} - T_{u_i}\} \tag{5.10}$$

donde T_{u_i} es la máxima temperatura permitida en el procesador q_i.

Este control ajusta la frecuencia de los procesadores asignando marcas en los lugares de frecuencia del modelo del procesador (lugar p_3 de la Fig. 4.2). Este control funciona de la siguiente manera.

Inicialmente, una frecuencia preliminar se calcula basado en una ley básica de control proporcional de la siguiente forma:

$$\begin{gathered} S_i = K_p \cdot e_{T_{CPU_i}} \\ S_{i_{min}} \leq S_i \leq S_{i_{max}} \\ i \in \{1, \ldots, m\}. \end{gathered} \tag{5.11}$$

Después, se selecciona de un conjunto de frecuencias disponibles ya conocidas el valor más próximo al calculado en (5.11). Se aplica la frecuencia seleccionada, colocando las marcas correspondientes al nivel de frecuencia en p_{n+3}, ..., p_{n+m} para los procesadores $q_1, \ldots, q_m$ respectivamente (véase

la Fig. 4.4). A medida que el error de temperatura se acerca al valor predefinido T_u, la frecuencia del procesador disminuye. Adicionalmente, si la temperatura medida en los procesadores excede la temperatura crítica, el control asigna la mínima frecuencia disponible y bloquea las transiciones $t_{i,j}$ y t_{n+j}, es decir que las tareas ya no se ejecutan puesto que hemos excedido la máxima temperatura permitida (ver Fig. 4.4). En tal caso, el flujo controlado de tales transiciones viene dado por:

$$\begin{aligned} w_{i,j} &= (f_{i,j} - u_{i,j}) \cdot psign(e_{T_{CPU_i}}) \\ w_{n+i} &= f_{n+i} - psign(e_{T_{CPU_i}}) \cdot f_{n+i}. \end{aligned} \tag{5.12}$$

Ejemplo 5.2 Considere un sistema de multiprocesadores en chip que consta de 4 procesadores de silicio montados sobre una placa difusora de cobre como en el Ejemplo 3.2. Inicialmente los procesadores se encuentran a una temperatura de 45^oC y cada uno de ellos trabaja con un rango de frecuencias como se muestra en la Tabla 5.1. Se desea mantener la temperatura de los cuatro procesadores en un valor menor o igual a $T_{u_1} = 95^oC$, $T_{u_2} = 85^oC$, $T_{u_3} = 80^oC$ y $T_{u_4} = 75^oC$ durante la ejecución de un conjunto de tareas periódicas e independientes $\tau = \{\tau_1, \tau_2\}$, donde cada tarea está representada por los siguientes parámetros temporales.

τ	cc_j	d_j	ω_j
τ_1	50	2	1
τ_2	100	2	2

De la misma manera que en el ejemplo 3.2 obtenemos el modelo térmico en variables de estado teniendo en cuenta que ahora se utiliza una placa difusora de cobre. Entonces, mediante el conocimiento del número de procesadores y la cantidad de tareas a ejecutar, se procede a modelar el sistema del ejemplo en cuestión obteniendo la estructura global en $TCPN$ como se muestra en la Fig. 5.4.

Con los parámetros proporcionados anteriormente y asumiendo que los procesadores consumen 500 ciclos de CPU por segundo, tenemos que el vector de tasa de disparo λ, el vector de flujos f y la matriz de incidencia C para este sistema se calculan de la siguiente manera:

$$\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_{1,1} \\ \lambda_{1,2} \\ \lambda_{2,1} \\ \lambda_{2,2} \\ \lambda_{3,1} \\ \lambda_{3,2} \\ \lambda_{4,1} \\ \lambda_{4,2} \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \end{bmatrix} = \begin{bmatrix} \frac{1}{\omega_1} \\ \frac{1}{\omega_2} \\ 500 \\ 500 \\ 500 \\ 500 \\ 500 \\ 500 \\ 500 \\ 500 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ cc_1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & cc_2 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{cc_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{cc_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{cc_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{cc_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{cc_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{cc_2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{cc_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{cc_2} & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} f_1 \\ f_2 \\ f_{1,1} \\ f_{1,2} \\ f_{2,1} \\ f_{2,2} \\ f_{3,1} \\ f_{3,2} \\ f_{4,1} \\ f_{4,2} \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{bmatrix} = \begin{bmatrix} \lambda_1 m_1 \\ \lambda_2 m_2 \\ \lambda_{1,1} min\{m_2, m_5, m_{13}\} \\ \lambda_{1,2} min\{m_4, m_5, m_{13}\} \\ \lambda_{2,1} min\{m_2, m_6, m_{14}\} \\ \lambda_{2,2} min\{m_4, m_6, m_{14}\} \\ \lambda_{3,1} min\{m_2, m_7, m_{15}\} \\ \lambda_{3,2} min\{m_4, m_7, m_{15}\} \\ \lambda_{4,1} min\{m_2, m_8, m_{16}\} \\ \lambda_{4,2} min\{m_4, m_8, m_{16}\} \\ \lambda_5 min\{m_9, m_{13}\} \\ \lambda_6 min\{m_{10}, m_{14}\} \\ \lambda_7 min\{m_{11}, m_{15}\} \\ \lambda_8 min\{m_{12}, m_{16}\} \end{bmatrix} \tag{5.13}$$

El número de tareas que finalizan después de su deadline se calcula de la siguiente manera:

$$\begin{aligned} e_{\tau_1} = m'_1 = max(0, \tfrac{m_2}{d_1} - \textstyle\sum_{j=1}^{4} f_{j,1}) - max(\textstyle\sum_{i=1}^{4} f_{i,1} - \tfrac{m2}{d_1}, 0) \\ e_{\tau_2} = m'_2 = max(0, \tfrac{m_4}{d_2} - \textstyle\sum_{j=1}^{4} f_{j,2}) - max(\textstyle\sum_{i=1}^{4} f_{i,2} - \tfrac{m4}{d_2}, 0), \end{aligned} \tag{5.14}$$

aplicando el control continuo para la asignación de tareas, obtenemos:

$$\begin{aligned} u_{1,1} &= psign(e_{\tau_1}) \cdot f_{1,1} \\ u_{1,2} &= psign(e_{\tau_2}) \cdot f_{1,2} \\ u_{2,1} &= psign(e_{\tau_1}) \cdot f_{2,1} \\ u_{2,2} &= psign(e_{\tau_2}) \cdot f_{2,2} \\ u_{3,1} &= psign(e_{\tau_1}) \cdot f_{3,1} \\ u_{3,2} &= psign(e_{\tau_2}) \cdot f_{3,2} \\ u_{4,1} &= psign(e_{\tau_1}) \cdot f_{4,1} \\ u_{4,2} &= psign(e_{\tau_2}) \cdot f_{4,2}, \end{aligned} \tag{5.15}$$

si la temperatura excede el valor predefinido, entonces los procesadores no deberían ejecutar ninguna tarea, es decir t_5, t_6, t_7 y t_8 no deben dispararse. Para establecer estas restricciones es necesario calcular lo siguiente:

$$\begin{aligned} u_5 &= psign(e_{T_1}) \\ u_6 &= psign(e_{T_2}) \\ u_7 &= psign(e_{T_3}) \\ u_8 &= psign(e_{T_4}). \end{aligned} \tag{5.16}$$

Para el control de temperatura, se calcula el error de temperatura como en la ecuación (5.10). Entonces, el control de temperatura se obtiene mediante el cálculo de la ecuación (5.8) y se es-

coge una frecuencia de la Tabla 5.1 más próxima al valor calculando con (5.10). Las frecuencias seleccionadas se asignan como marcas a los lugares p_{13},p_{14},p_{15},p_{16}.

Por último, se calculan las entradas de control $w_{i,j}$, w_5, w_6, w_7 y w_8 como sigue:

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_{1,1} \\ w_{1,2} \\ w_{2,1} \\ w_{2,2} \\ w_{3,1} \\ w_{3,2} \\ w_{4,1} \\ w_{4,2} \\ w_5 \\ w_6 \\ w_7 \\ w_8 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ (f_{1,1} - u_{1,1}) \cdot psign(e_{T_1}) \\ (f_{1,2} - u_{1,2}) \cdot psign(e_{T_1}) \\ (f_{2,1} - u_{2,1}) \cdot psign(e_{T_2}) \\ (f_{2,2} - u_{2,2}) \cdot psign(e_{T_2}) \\ (f_{3,1} - u_{3,1}) \cdot psign(e_{T_3}) \\ (f_{3,2} - u_{3,2}) \cdot psign(e_{T_3}) \\ (f_{4,1} - u_{4,1}) \cdot psign(e_{T_4}) \\ (f_{4,2} - u_{4,2}) \cdot psign(e_{T_4}) \\ f_5 - u_5 \\ f_6 - u_6 \\ f_7 - u_7 \\ f_8 - u_8 \end{bmatrix} \tag{5.17}$$

Finalmente mediante los cálculos anteriores, el modelo global en $TCPN$ que representa el comportamiento térmico del $MPSoC$ al ejecutar un conjunto de tareas periódicas e independientes es simulado en SimHPN [28].

La Fig. 5.5 muestra la temperatura de los procesadores producida por la ejecución de las tareas a diferentes niveles de frecuencia. El control de temperatura asigna los niveles de frecuencia adecuados para mantener la temperatura del procesador por debajo del umbral de temperatura permitido como se muestra en la Fig. 5.6.

La asignación de las tareas a los procesadores lo realiza el control de asignación de flujos (ecuación (5.16)) que se muestra en la Fig. 5.7, aquí se observa que las transiciones se activan o bloquean a medida que se tengan tareas que estén cerca de no cumplir con su fecha límite de ejecución, por lo que el *throughput* en los procesadores depende de la prioridad que se asigne con los errores e_{τ_1} y e_{τ_2}.

La Fig. 5.8 es el resultado del control de asignación cuando se tienen $cc_1 = 1000$ y $cc_2 = 1500$ ciclos de cómputo. En este caso el sistema esta sobrecargado, ya que por cada periodo de las tareas se tienen 2500 ciclos de cómputo los cuales pueden ser asignados a los 4 procesadores para su ejecución. Por consiguiente, el control de flujos en las transiciones (ecuación (5.16)) para la asignación de las tareas conmuta rápidamente para asegurar que se cumplan los plazos de tiempo. Sin embargo, el cálculo de las frecuencias de trabajo en los procesadores y la temperatura que se obtienen mediante estos ciclos de cómputo son los mismos que con los ciclos de cómputo considerados anteriormente por lo que se omiten las figuras respectivas.

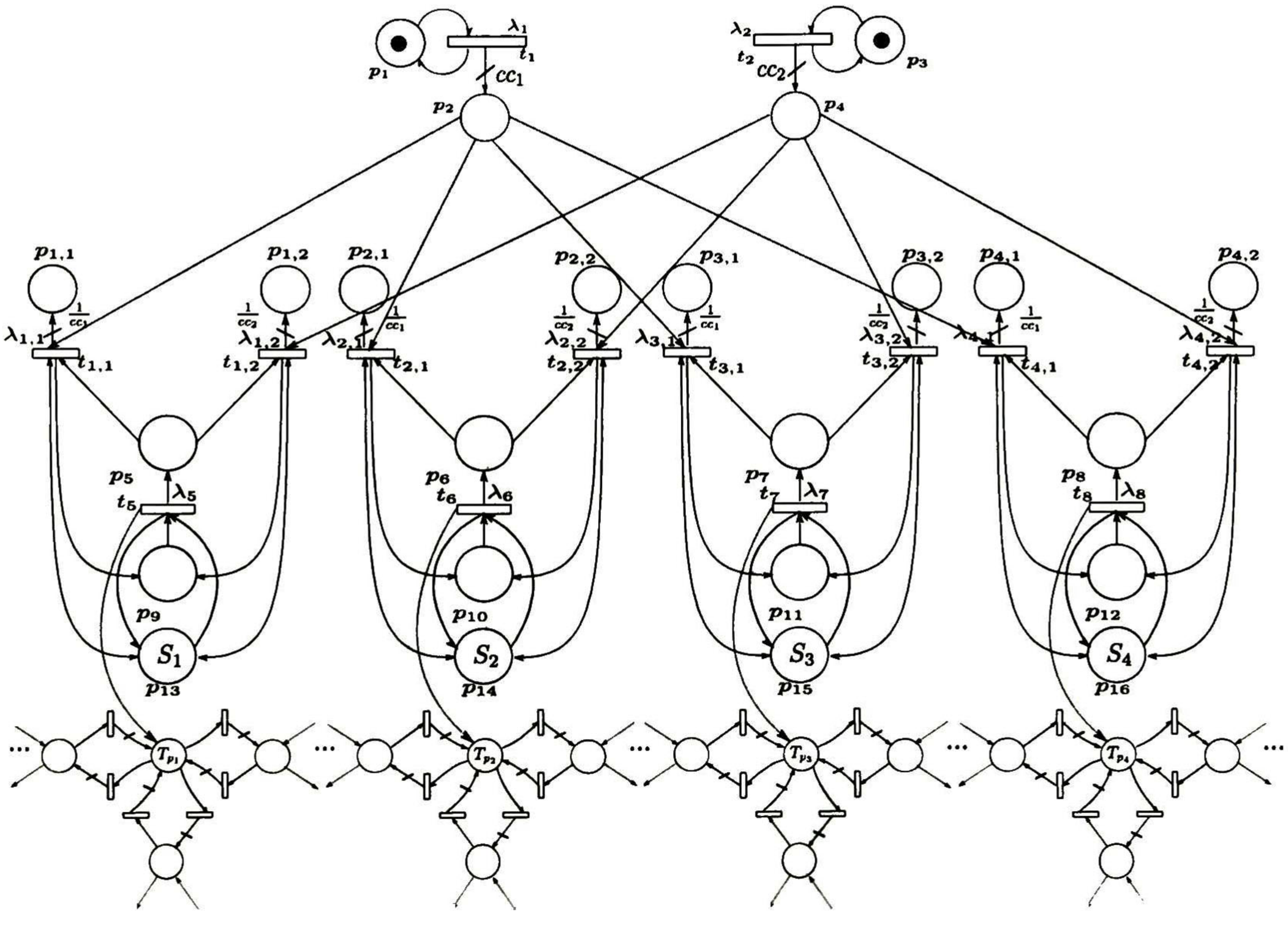


Figura 5.4: Modelo global en $TCPN$ para 2 Tareas y 4 Procesadores.

Discretización para el scheduling continuo

Los modelos de tareas y procesador con $TCPN$, es utilizado para resolver el problema de asignación sin violar las restricciones térmicas y temporales especificadas. Aunque el *throughput* proporcionado por el scheduler continuo mediante el scheduler en $TCPN$ presentado anteriormente se calcula eficientemente evitando el problema de la combinatoria (problema de scheduling), éste no puede ser aplicado a un $MPSoC$, dado que el flujo calculado es un número *real* mientras que las tareas que deben de ser ejecutas en cada procesador del $MPSoC$ deben de ser un número *natural.*

Afortunadamente, podemos conocer del modelo global en $TCPN$ una cantidad finita de tareas periódicas que han sido ejecutadas en un procesador q_i mediante el lugar $p_{i,j}$ el cual está conectado a la transición $t_{i,j}$ mediante un arco con peso $\frac{1}{cc_j}$ como se muestra en la Fig. 4.4. El marcado en $p_{i,j}$ indica cuántas tareas τ_j son ejecutas en el procesador q_i en cualquier tiempo η.

Con el conocimiento de la cantidad de tareas que se ejecutan en cierto tiempo η en el modelo global en $TCPN$, se propone un algoritmo que discretiza el schedule continuo obtenido en la Sección

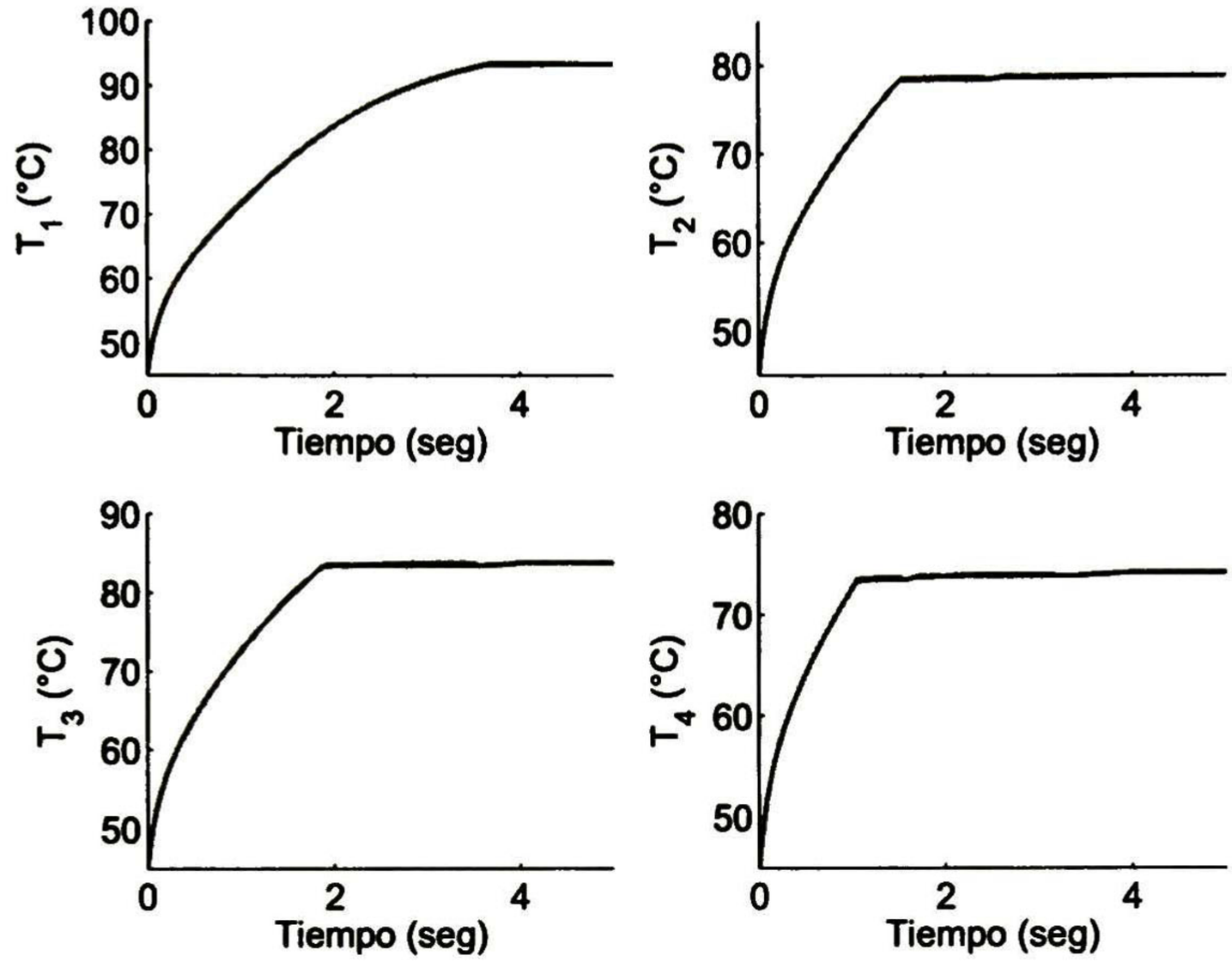


Figura 5.5: Temperatura

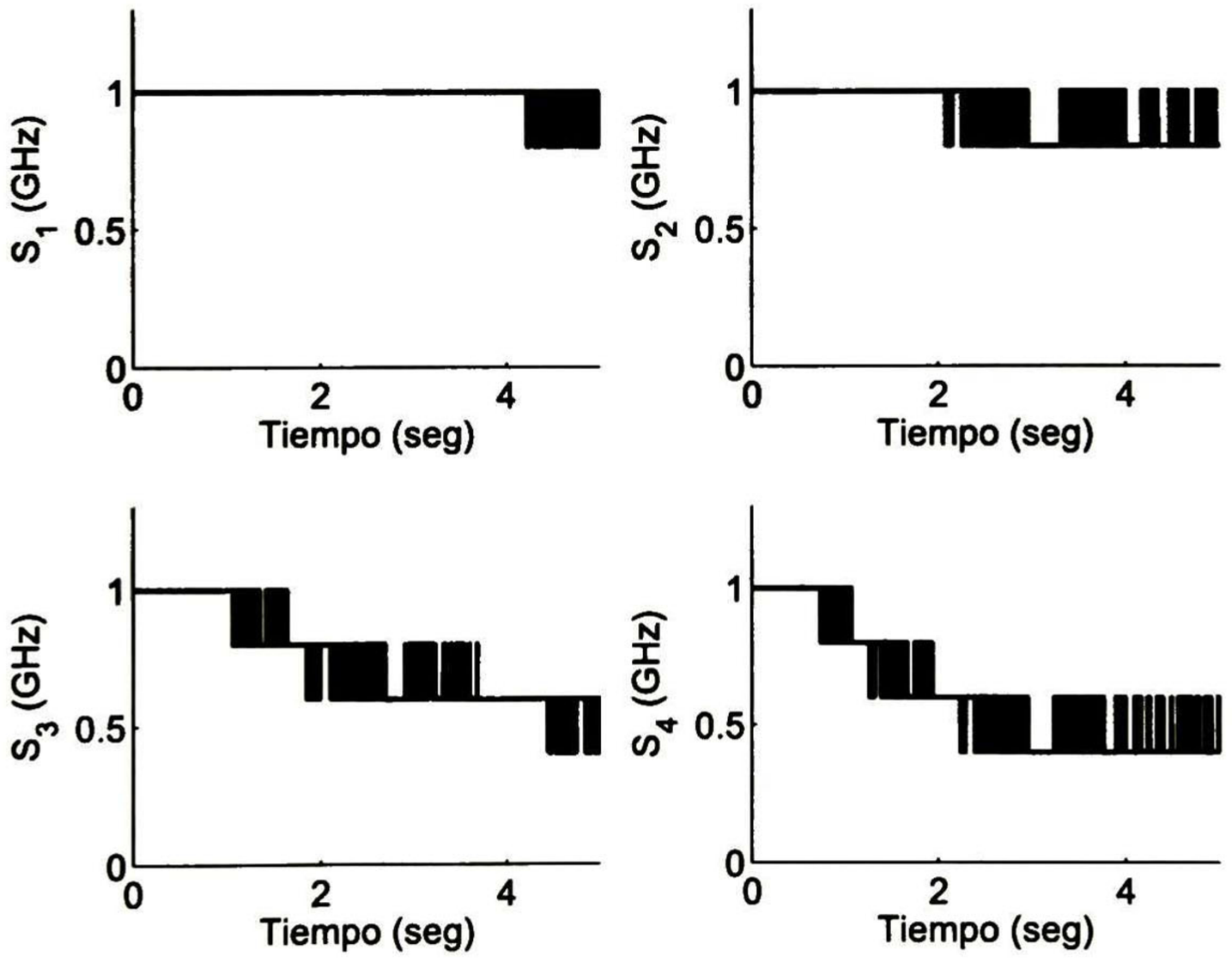


Figura 5.6: Frecuencia asignada

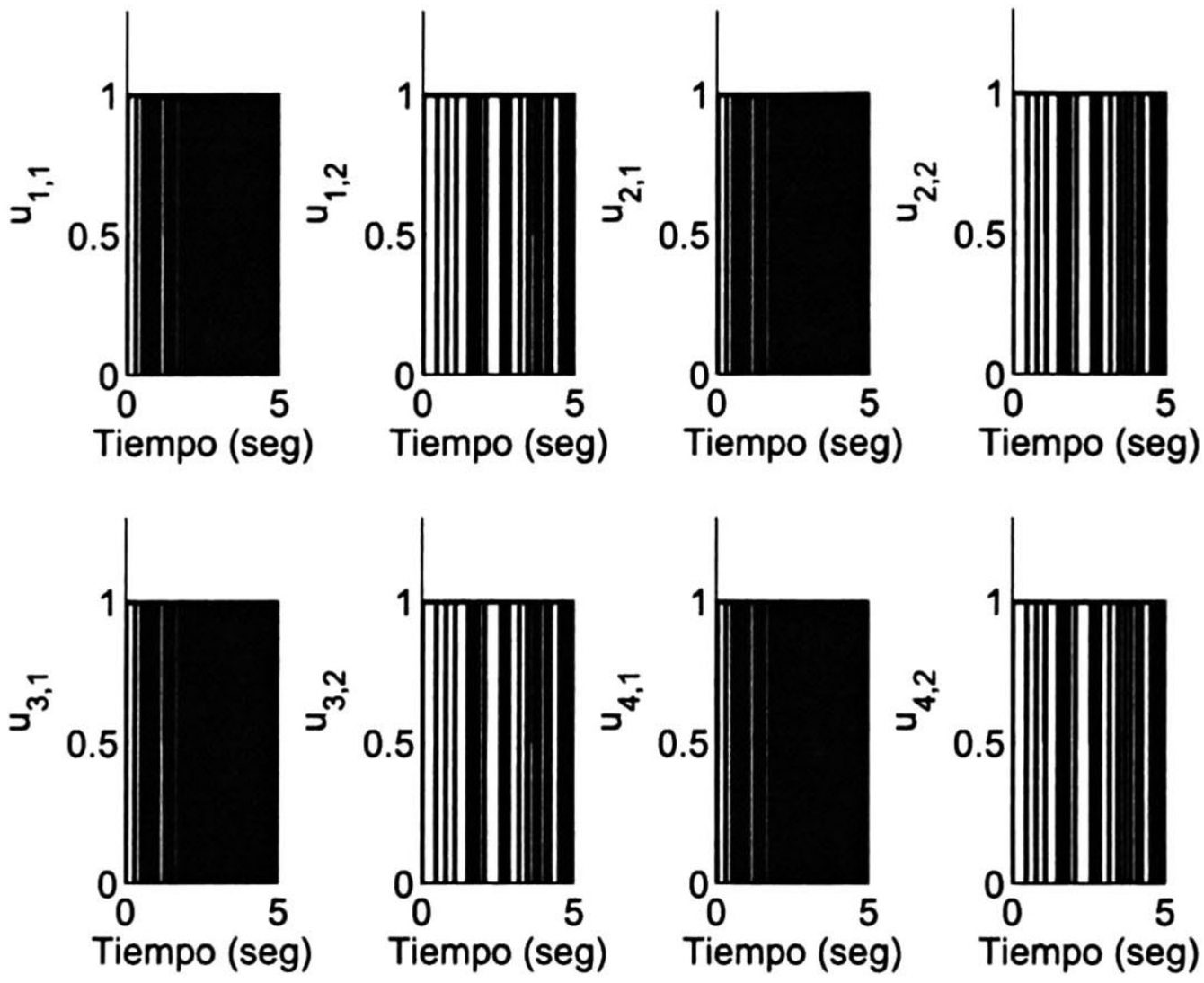


Figura 5.7: Control para asignación de tareas

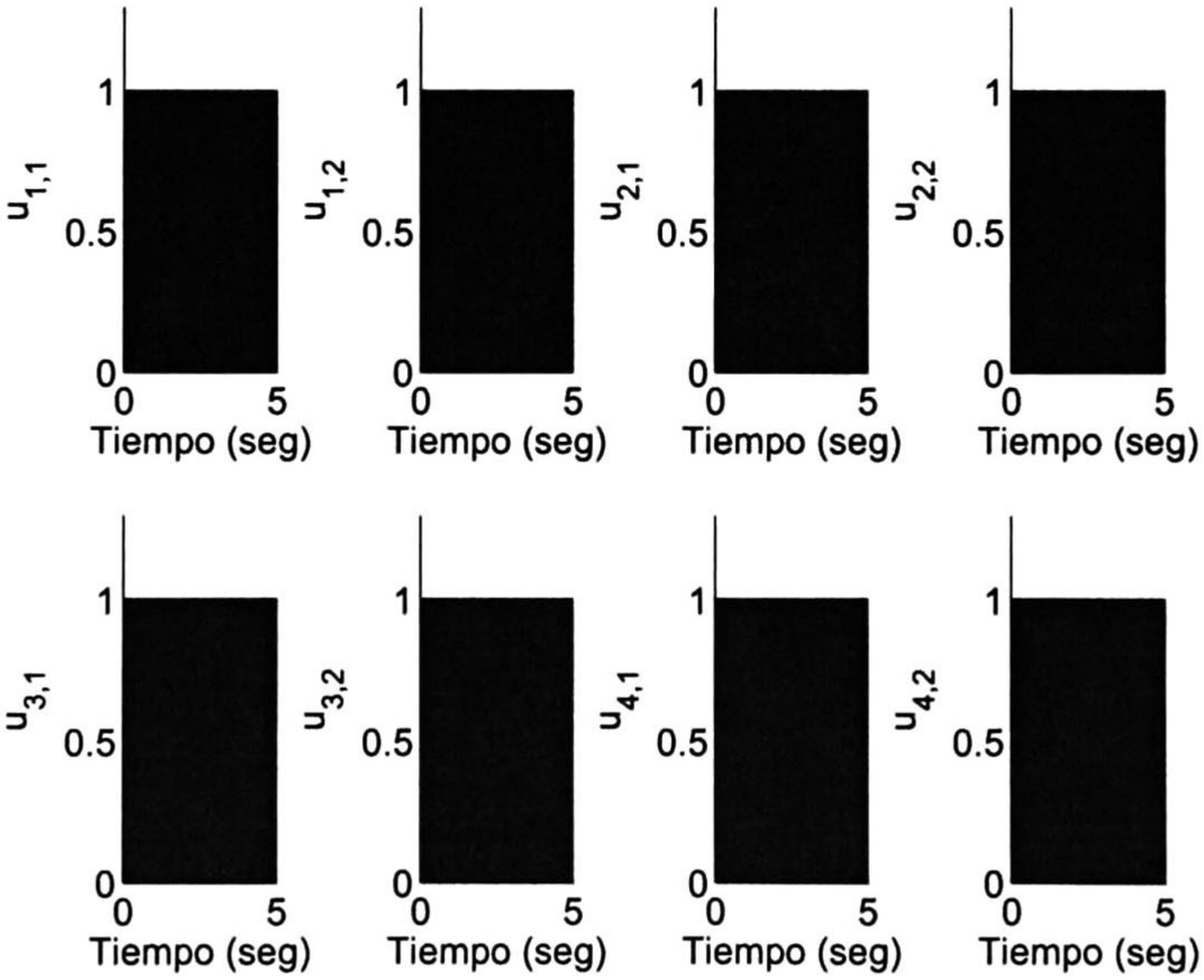


Figura 5.8: Control para asignación de tareas, sistema sobrecargado

anterior con el fin de asignar las tareas de manera discreta.

El algoritmo propuesto opera en pequeños intervalos de tiempo, es decir, en cada instante de tiempo t_d (donde t_d es el mínimo común divisor de los periodos en el conjunto de tareas τ), el algoritmo calcula la integral $I_{i,j}$ de cada flujo $w_{i,j}$ ($I_{i,j} = \int_{(n-1)t_d}^{nt_d} w_{i,j}(\eta)d\eta$) que se obtiene durante la ejecución del scheduler continuo en $TCPN$. Esta integral, representa la cantidad de tareas τ_j que deberán ser ejecutadas en el procesadore q_i en el tiempo t_d. Después, el algoritmo asigna $I_{i,j}$ ciclos de cómputo de la tareas τ_j al procesador q_i.

Algoritmo 1 Discretización del scheduling continuo

1: **Entrada** $I_{i,j}$ calculado de $(n-1)t_d$ a $(n)t_d$ por el control de scheduling continuo.
2: **Salida** Un schedule discreto.
3: *Por cada par* q_i, τ_j *calcular la integral*

$$I_{i,j} = \int_{(n-1)t_d}^{nt_d} w_{i,j}(\eta)d\eta$$

4: *Asignar* $I_{i,j}$ *ciclos de CPU de la tarea* τ_j *a* q_i *en el periodo de tiempo* $(n-1)t_d$ *a* $(n)t_d$. *La tarea* τ_j *no debe ser asignada a ningún otro procesador q simultáneamente con* q_i.
5: ***Regresa*** *Un schedule en el tiempo* $(n-1)t_d$ *a* $(n)t_d$

Para mostrar el funcionamiento del algoritmo anterior, considere el Ejemplo 5.2, con las mismas consideraciones térmicas. El modelo en $TCPN$ de asignación y ejecución obtenido anteriormente para este ejemplo proporciona en los lugares $p_{1,1}$ y $p_{1,2}$ la cantidad de tareas periódicas τ_1 ó τ_2 que fueron ejecutadas por el procesador q_1, de la misma forma los lugares $p_{2,1}$,$p_{2,2}$ proporcionan la cantidad de tareas para el procesador q_2, y así sucesivamente para los procesadores q_3 y q_4 se tienen los lugares $p_{3,1}$, $p_{3,2}$ y $p_{4,1}$, $p_{4,2}$ respectivamente. El tiempo t_d se calcula como el mínimo común divisor de los periodos de las tareas τ_1 y τ_2 ($t_d = 1$). A medida que estos lugares aumentan en sus marcados, el algoritmo 1 regresa un schedule cada t_d unidades de tiempo.

La Fig. 5.10 muestra la asignación para el Ejemplo 5.2, preservando las fechas límites de entrega con su respectivo consumo de potencia en cada procesador. En la Fig. 5.9 se puede observar la temperatura de los procesadores generados por los schedules del Algoritmo 1, note que las restricciones térmicas se preservan.

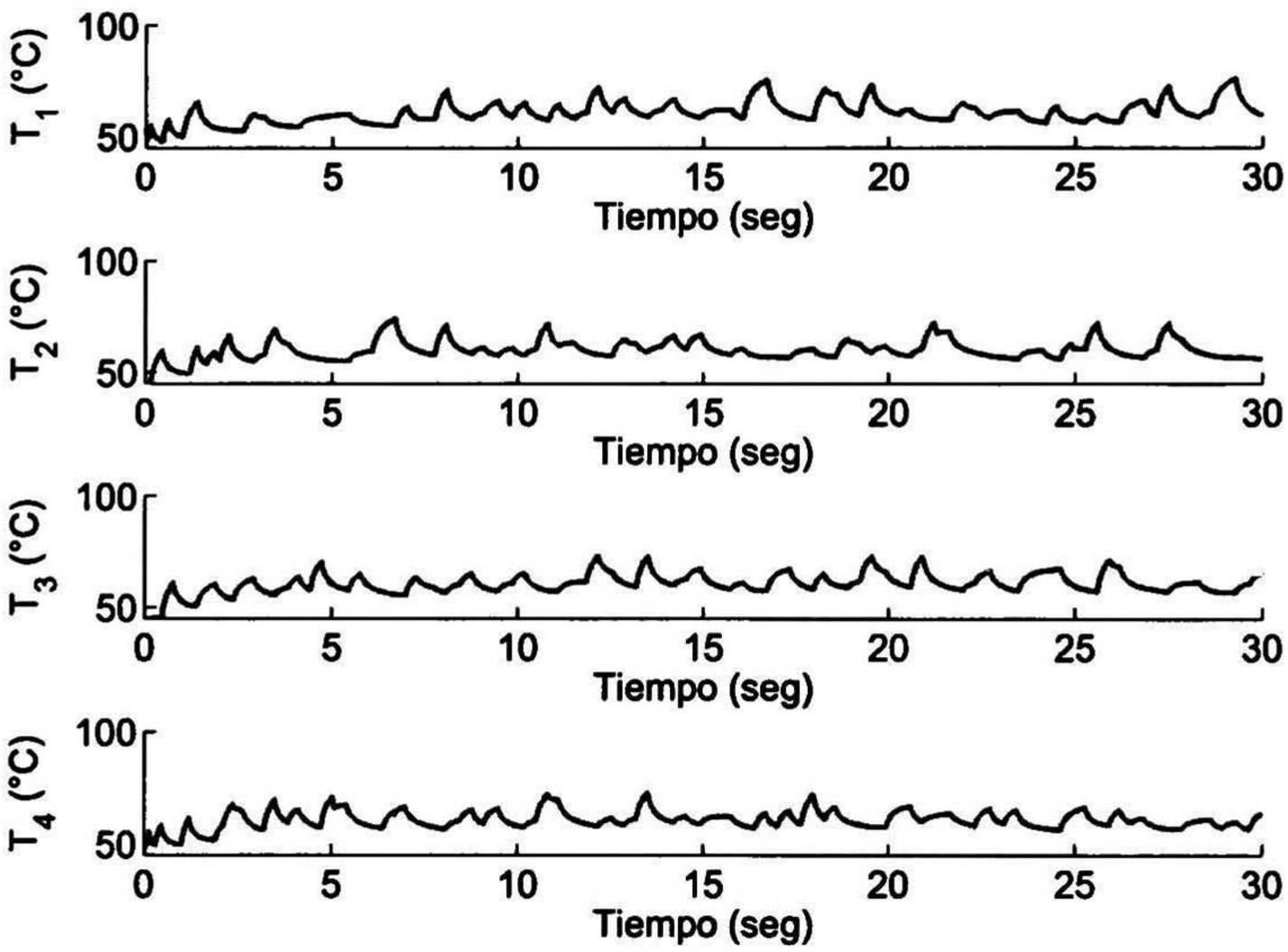


Figura 5.9: Temperatura de los procesadores durante la ejecución del Algoritmo 1.

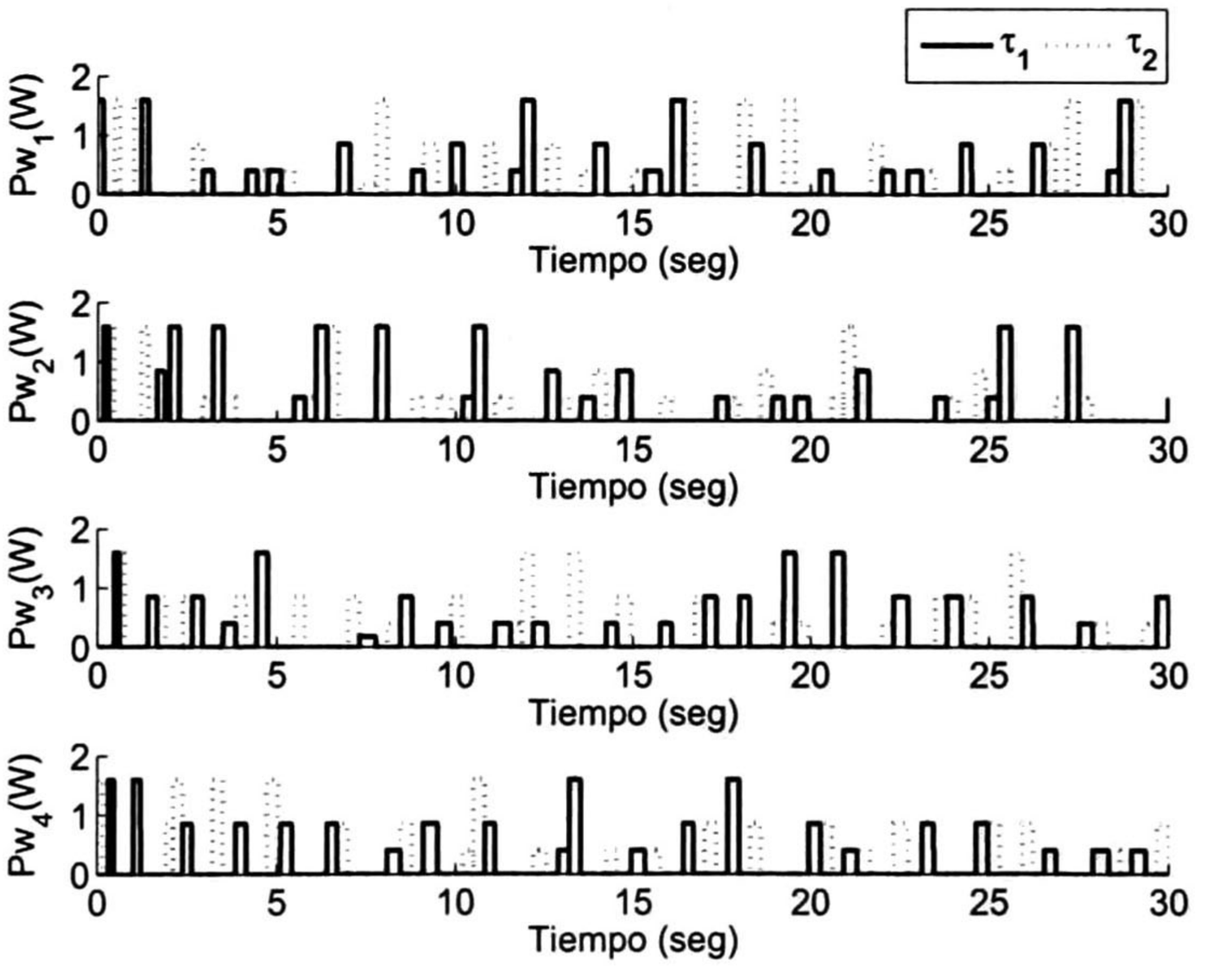


Figura 5.10: Potencia generada por medio de la asignación de tareas a los procesadores usando el Algoritmo 1

Conclusiones

En este trabajo, se presentó una metodología eficiente para modelar el comportamiento térmico de un $MPSoC$. La metodología consistió en transformar las ecuaciones lineales del modelo térmico en un modelo monolítico de red de Petri. El modelo térmico se construyó de forma incremental, mediante la fusión de los módulos en $TCPN$ de conducción, convección y generación de calor. La metodología propuesta obtiene modelos con la misma precisión que los modelos que usan métodos de diferencias finitas evitando la etapa de calibración que otras metodologías de modelado necesitan; además esta metodología proporciona un modelo de espacio de estado útil para propósitos de control de temperatura en un $MPSoC$.

Por otra parte, se presentó una metodología para el modelado de conjuntos de tareas periódicas e independientes, donde el tiempo de ejecución y las fechas límites de entrega son modelados como parámetros en la red de Petri. Se modeló también mediante el mismo formalismo, a un conjunto de procesadores homogéneos que fusionado con el modelo de tareas conforman el modelo de asignación y ejecución de tareas en un $MPSoC$. Mediante la unión de los modelos térmico y de ejecución de tareas en procesadores se obtuvo un modelo global que usa un solo formalismo.

Con el fin de mostrar la utilidad del modelo global derivado, se presentó dos schedulers térmicos, uno con el algoritmo MPC y el otro basado en un scheduler continuo. Dado el carácter discreto del problema de asignación, se propuso un algoritmo de discretización para el scheduler continuo.

Como trabajo futuro se abordará el problema de diseño de schedulers térmicos utilizando el modelo $TCPN$ global propuesto en esta tesis, con el fin de reducir la complejidad computacional al tratar con un gran número de tareas. Además, se considerarán módulos de tareas esporádicas, procesadores no homogéneos, y se propondrán otros algoritmos de dicretización para el scheduler continuo.

Bibliografía

[1] D. J. Jamieson, A. D. Mansell, J. A. Staniforth, and D. W. Tebb, "Application of finite difference techniques for the thermal modelling of power electronic switching devices," in *Power Electronics and Variable-Speed Drives, 1994. Fifth International Conference on*, pp. 313–318, Oct 1994.

[2] S. Rao, *The finite element method in engineering.* Butterworth-Heinemann, Elsevier, 2011.

[3] S. Liu, J. Zhang, Q. Wu, and Q. Qiu, "Thermal-aware job allocation and scheduling for three dimensional chip multiprocessor," in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, pp. 390–398, March 2010.

[4] S. Zhang and K. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pp. 281–288, Nov 2007.

[5] T. Chantem, X. Hu, and R. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on mpsocs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, pp. 1884–1897, Oct 2011.

[6] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on*, pp. 2–13, June 2003.

[7] K. Skadron, T. Abdelzaher, and M. Stan, "Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management," in *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pp. 17–28, Feb 2002.

[8] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. De Micheli, "Temperature-aware processor frequency assignment for mpsocs using convex optimization," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2007 5th IEEE/ACM/IFIP International Conference on*, pp. 111–116, Sept 2007.

[9] D. Bild, S. Misra, T. Chantemy, P. Kumar, R. Dick, X. Huy, L. Shangz, and A. Choudhary, "Temperature-aware test scheduling for multiprocessor systems-on-chip," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pp. 59–66, Nov 2008.

[10] J.-J. Chen, C.-M. Hung, and T.-W. Kuo, "On the minimization fo the instantaneous temperature for periodic real-time tasks," in *Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07. 13th IEEE*, pp. 236–248, April 2007.

[11] R. Jayaseelan and T. Mitra, "Dynamic thermal management via architectural adaptation," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*, pp. 484–489, IEEE, 2009.

[12] A. Qadi, S. Goddard, and S. Farritor, "A dynamic voltage scaling algorithm for sporadic tasks," in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, pp. 52–62, IEEE, 2003.

[13] A. K. Coskun, T. S. Rosing, and K. Whisnant, "Temperature aware task scheduling in mpsocs," in *Proceedings of the conference on Design, automation and test in Europe*, pp. 1659–1664, EDA Consortium, 2007.

[14] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*, pp. 171–182, IEEE, 2001.

[15] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic thermal management through task scheduling," in *Performance Analysis of Systems and software, 2008. ISPASS 2008. IEEE International Symposium on*, pp. 191–201, IEEE, 2008.

[16] K. Kang, J. Kim, S. Yoo, and C.-M. Kyung, "Temperature-aware integrated dvfs and power gating for executing tasks with runtime distribution," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 9, pp. 1381–1394, 2010.

[17] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha, "Hybdtm: a coordinated hardware-software approach for dynamic thermal management," in *Proceedings of the 43rd annual Design Automation Conference*, pp. 548–553, ACM, 2006.

[18] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *ACM SIGARCH computer architecture news*, vol. 37, pp. 314–324, ACM, 2009.

[19] J. Dessel and j. Esparza, *Free Choice Petri Nets*. Press Syndicate of the University of Cambridge, 1995.

[20] C. Mahulea, A. Ramirez-Trevino, L. Recalde, and M. Silva, "Steady-state control reference and token conservation laws in continuous petri net systems," *Automation Science and Engineering, IEEE Transactions on*, vol. 5, pp. 307–320, April 2008.

[21] F. DiCesare, *Practice of Petri nets in manufacturing*. Champman and Hall, 1993.

[22] C. Ramchandani, *Analysis of Asynchronous Concurrent Systems by Petri Nets*. PhD thesis, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC, Feb 1974.

[23] H. Alla and R. David, "Continuous and hybrid Petri nets," *Revue Journal of Circuits, Systems and Computers, Special Issue on Petri Nets*, vol. 1, pp. 159–188, 1998.

[24] F. Balduzzi, A. D. Febbraro, A. Giua, and C. Seatzu, "Decidability results in first order hybrid petrinets," *Discrete Event Dynamic Systems*, vol. 11, pp. 41–57, Jan. 2001.

[25] N. Zerhouni, M. Ferney, and A. El Moudni, "Transient analysis of manufacturing systems using continuous petri nets," *Math. Comput. Simul.*, vol. 39, pp. 635–639, Nov. 1995.

[26] J. B. J. Emilio, "Tecnicas Algebraicas para el Analisis y Control de Redes de Petri Continuas," 2005.

[27] A. Balluchi, L. Benvenuti, M. D. D. Benedetto, and A. L. Sangiovanni-Vincentelli, "Design of observers for hybrid systems," in *Hybrid Systems: Computation and Control, volume 2289 of LNCS*, pp. 76–89, Springer-Verlag, 2002.

[28] J. Júlvez, C. Mahulea, and C.-R. Vázquez, "SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets," *Nonlinear Analysis: Hybrid Systems*, vol. 6, pp. 806–817, May 2012.

[29] M. Silva and L. Recalde, "Petri nets and integrality relaxations: A view of continuous petri net models," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, pp. 314–327, 2002.

[30] J. Welty, C. E. Wicks, R. E. Wilson, and G. L. Rorrer, *Fundamentals of Momentum, Heat, and Mass Transfer*. John Wiley and Sons, Inc., 2007.

[31] A. Burns and A. Wellings, *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. Addison-Wesley Educational Publishers Inc, 2009.

[32] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*, vol. 24. Springer, 2011.

[33] K. Jeffay, D. F. Stanat, and C. U. Martel, "On non-preemptive scheduling of period and sporadic tasks," in *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*, pp. 129–139, IEEE, 1991.

[34] A. Ramírez, *Scheduling en redes de Petri*. PhD thesis, Universidad de Zaragoza, Feb 1993.

[35] M. Pinedo, "Scheduling: Theory, and systems," 2008.

[36] B. Grabot and L. Geneste, "Dispatching rules in scheduling dispatching rules in scheduling: A fuzzy approach," *The International Journal of Production Research*, vol. 32, no. 4, pp. 903–915, 1994.

[37] J.-J. Chen and T.-W. Kuo, "Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems.," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pp. 289–294, Nov 2007.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Modelado y Scheduling Térmico para MPSoCs usando Redes de Petri Continuas Temporizadas

del (la) C.

Gaddiel DESIRENA LÓPEZ

el día 11 de Diciembre de 2014.

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

Dr. José Javier Ruíz León
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

Dr. Antonio Ramírez Treviño
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dra. Susana Ortega Cisneros
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

CINVESTAV - IPN
Biblioteca Central
SSIT0013025