Escalonador de Tempo Real em SystemC

Hugo Manaia, Arnaldo Oliveira, Nuno Lau, Orlando Moreira

Abstract – **This paper discusses the implementation, features and utilization of a real-time scheduler simulator modeled with SystemC. Currently, this scheduler handles only periodic hard real-time tasks. Three scheduling policies were implemented: RM (Rate Monotonic), EDF (Earliest Deadline First) and LSF (Least Slack First). A set of task-related commands is available, which allows the user to create, destroy, stop, start, sleep, suspend, resume and change the task parameters. These commands can be specified in a configuration file that is read during initialization. In the configuration file, besides the specific parameters of each command, a time stamp is assigned to each command. In each simulation a VCD (Value Change Dump) file is generated, which allows the visualization of the evolution of some of the most important system signals.**

Resumo – **Este artigo descreve a implementação, as funcionalidades e a utilização de um simulador de um escalonador de tempo real desenvolvido em SystemC. Actualmente este escalonador aceita apenas tarefas críticas periódicas de tempo real. Três políticas de escalonamento foram implementadas: RM (*Rate Monotonic*), EDF (*Earliest Deadline First*) e LSF (*Least Slack First*). É disponibilizado um conjunto de comandos que permite manipular as tarefas, dando a possibilidade ao utilizador ou ao escalonador de criar, destruir, parar, arrancar, adormecer, suspender, retomar e alterar parâmetros das tarefas. Estes comandos podem ser especificados num ficheiro de configuração que é lido durante a fase de inicialização. Neste ficheiro de configuração especifica-se, para além dos parâmetros específicos de cada comando, o tempo no qual se pretende que este ocorra. Em cada simulação é gerado um ficheiro VCD (*Value Change Dump*) através do qual se pode acompanhar a evolução de alguns dos sinais mais importantes que caracterizam o estado do sistema.**

Keywords – **Real-time systems, Task scheduling, Rate Monotonic, Earliest Deadline First, Least Slack First, SystemC**

Palavras chave – **Sistemas de tempo real, escalonamento de tarefas, *Rate Monotonic*, *Earliest Deadline First*, *Least Slack First*, SystemC**

I. INTRODUÇÃO

Os sistemas computacionais uniprocessador lidam cada vez mais com tarefas de tempo real, em especial tarefas críticas de tempo real. A estes sistemas dá-se o nome de sistemas de tempo real.

As tarefas de tempo real têm características diferentes das usuais, com restricções temporais que terão que ser cumpridas. Aos sistemas de tempo real é pedido que executem as tarefas de forma a estas cumprirem as restricções. Para isso, os sistemas agendam as tarefas para execução mediante as suas necessidades, e não numa política *First Come First Served*.

O escalonador é o componente de um sistema de tempo real encarregue do agendamento para execução de tarefas. Este agendamento é feito, normalmente, com base em prioridades. Para a realização desta actividade, um escalonador tem em conta diversas características das próprias tarefas, em função da política de escalonamento a utilizar. Em cada instante, é agendada para execução a tarefa pronta a executar com prioridade mais elevada. As prioridades podem ser fixas ou dinâmicas. No caso de um sistema de tempo real que incorpore escalonamento de tarefas com atribuição dinâmica de prioridades é a política de escalonamento que define como devem as prioridades das tarefas variar.

Para além desta introdução, este artigo estende-se ao longo de mais 6 secções. Na secção seguinte resumem-se alguns conceitos relativos a sistemas de tempo real, úteis na compreensão deste artigo. Na terceira secção é apresentada a arquitectura do escalonador desenvolvido. Na quarta secção discutem-se alguns pormenores de implementação. Na secção V é descrito como criar um exemplo e são analisados três exemplos apresentados. Por fim são apresentados alguns tópicos referentes a possível trabalho futuro bem como algumas conclusões.

II. SISTEMAS DE TEMPO REAL

Ao contrário de outros tipos de sistemas nos quais o mais importante pode passar pelo desempenho médio ou pela sequência de operações, um sistema de tempo real é [1], [2]:

- Um sistema computacional capaz de responder a eventos dentro de restrições temporais precisas;
- Um sistema cujo funcionamento correcto depende do instante de produção das saídas e não apenas do valor destas;
- Um sistema cuja dinâmica deve estar sincronizada com a do ambiente em que opera.

Assim, um sistema de tempo real não necessita necessariamente de responder rapidamente, mas sim de responder no instante certo, de responder a tempo.

A. Tarefas

Os sistemas de tempo real são, em geral, constituídos por um conjunto de tarefas ou processos que executam concorrentemente. Uma tarefa é uma sequência de instruções que, na ausência de outras actividades, é executada ininterruptamente pelo processador até ser completada. Simplificando um pouco, uma tarefa pode encontrar-se num de 3 estados [1]:

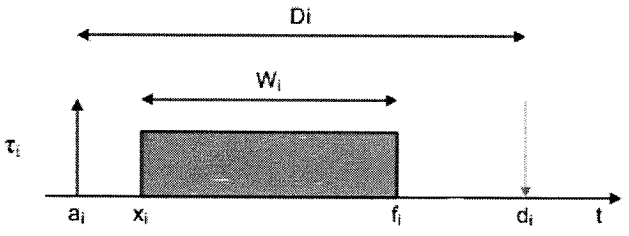

Figure 1 - Parâmetros temporais de uma tarefa.

- A executar
- Pronta a executar
- Bloqueada

As tarefas a executar ou prontas a executar, ou seja, as tarefas que num dado instante puderem ser executadas pelo processador, designam-se por activas.

Em termos de modo de activação, as tarefas podem ser de dois tipos distintos [1], [2]:

- Periódicas - a tarefa é automaticamente activada em intervalos de tempo regulares;
- Aperiódicas - a tarefa é activada assincronamente quando ocorrer um evento ou através da invocação explícita de uma primitiva de activação.

Em relação ao nível de criticalidade, as tarefas podem ser caracterizadas da seguinte forma [1], [2]:

- Ordinárias - se não possuírem quaisquer restrições temporais;
- *Soft real-time* - se o não cumprimento duma restrição temporal causar apenas uma degradação do desempenho do sistema;
- *Hard real-time* - se o não cumprimento duma restrição temporal causar efeitos catastróficos no sistema controlado.

A.1 Parâmetros

A execução de uma tarefa τ_i é caracterizada por diversos parâmetros, entre os quais (figura 1) [1], [2]:

- a_i - instante de activação;
- x_i - instante de execução;
- f_i - instante de terminação;
- d_i - *deadline* absoluta;
- D_i - *deadline* relativa;
- W_i - tempo máximo de execução (*Worst Case Execution Time* - WCET).

Uma tarefa periódica, além dos parâmetros acima, tem também um período T_i associado. Numa tarefa periódica caracterizada pelo terno $\tau_i(W_i, T_i, D_i)$ verificam-se as seguintes relações [1], [2]:

$$a_{i,k} = a_{i,k-1} + T_i$$

$$d_{i,k} = a_{i,k} + D_i$$

$$W_i \leq T_i$$

em que k é a k-ésima activação da tarefa.

B. Escalonamento

Um escalonamento é uma atribuição particular de tarefas ao processador. À estratégia usada para escolher uma tarefa entre as que se encontram activas e atribuir-lhe tempo de processador, chama-se algoritmo ou política de escalonamento.

O escalonamento em sistemas de tempo real pode ser efectuado de diversas maneiras. Uma das mais utilizadas é o escalonamento baseado em prioridades, que poderão ser fixas ou dinâmicas. As políticas disponíveis no escalonador implementado são:

- *Rate Monotonic* (RM);
- *Earliest Deadline First* (EDF);
- *Least Slack First* (LSF).

B.1 Prioridades fixas e dinâmicas

As políticas de escalonamento baseadas em prioridades podem usar prioridades fixas ou dinâmicas.

Utilizando uma política baseada em prioridades fixas, a política define a prioridade a atribuir a cada tarefa e esta não mais é alterada ao longo do escalonamento. As prioridades das tarefas são definidas com base em parâmetros ou características fixas das tarefas.

Com uma política que use prioridades dinâmicas, pelo contrário, as prioridades das tarefas podem ser alteradas durante o processo de escalonamento. As políticas de escalonamento baseadas neste tipo de prioridades têm em conta não só características fixas das tarefas mas também parâmetros só conhecidos em tempo de execução do escalonador.

De entre as políticas disponíveis no escalonador implementado, a RM baseia-se em prioridades fixas sendo que as restantes (EDF e LSF) utilizam prioridades dinâmicas.

B.2 Rate Monotonic

O algoritmo RM é um algoritmo de escalonamento baseado em prioridades fixas. A atribuição das prioridades é feita com base no período das tarefas, sendo inversamente proporcionais a este. É seleccionada para execução a tarefa com período mais pequeno. Uma tarefa pode ser interrompida por uma de prioridade mais elevada, pelo que se diz que o algoritmo é preemptivo.

O algoritmo RM é considerado óptimo entre os algoritmos de escalonamento de prioridades fixas, ou seja, se um conjunto de tarefas for escalonável utilizando uma qualquer política de prioridades fixas, então é escalonável utilizando RM.

A figura 2 mostra um exemplo de 3 tarefas usando a política de escalonamento RM.

Na figura cada linha temporal representa uma tarefa, identificadas por τ_i. As setas ascendentes representam activações das tarefas, ao passo que as descendentes representam as *deadlines* das mesmas. Os blocos representam os intervalos de tempo nos quais as tarefas executam realmente. De notar que a tarefa τ_3 é a tarefa com mais alta prioridade pois é aquela que executa mais frequentemente. Assim, cada vez que esta tarefa fica pronta a executar a tarefa que estiver na posse do processador sofre preempção, ou seja, o escalonador interrompe a sua execução. No lado

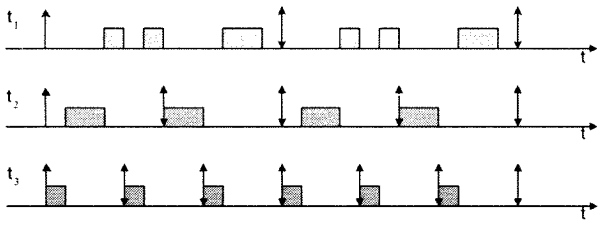

Figure 2 - Exemplo de um escalonamento de 3 tarefas usando o algoritmo RM.

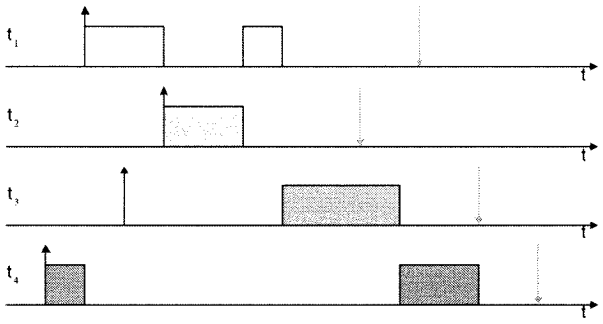

Figure 3 - Exemplo de um escalonamento de 4 tarefas usando o algoritmo EDF.

oposto encontra-se a tarefa τ_1 que, sendo a tarefa com prioridade mais baixa, é interrompida cada vez que uma outra tarefa se encontra pronta a executar.

B.3 Earliest Deadline First

O algoritmo EDF é um algoritmo de escalonamento baseado em prioridades dinâmicas, isto é, tem em conta um parâmetro dinâmico só conhecido em tempo de execução do escalonador. Neste algoritmo, as prioridades das tarefas são inversamente proporcionais ao tempo até à *deadline*. É seleccionada para execução a tarefa com *deadline* absoluta mais próxima. Tal como o algoritmo RM, este é um algoritmo preemptivo.

O algoritmo EDF é considerado óptimo entre todos os algoritmos de escalonamento, ou seja, se um conjunto de tarefas for escalonável, então é escalonável utilizando EDF, e permite alcançar uma taxa de ocupação do processador de 100%.

A figura 3 mostra um exemplo de 4 tarefas usando a política de escalonamento EDF.

B.4 Least Slack First

O algoritmo LSF é um algoritmo de escalonamento baseado em prioridades dinâmicas, isto é, tem em conta um parâmetro dinâmico só conhecido em tempo de execução do escalonador. Neste algoritmo, as prioridades das tarefas são inversamente proporcionais ao tempo livre das mesmas, isto é, são inversamente proporcionais à folga que a tarefa teria para a *deadline* caso executasse ao longo do seu tempo máximo de execução. É seleccionada para execução a tarefa com tempo livre mais pequeno. Tal como os algoritmos anteriores, este é um algoritmo preemptivo. Comparando com o algoritmo EDF, este provoca maior número de preempções.

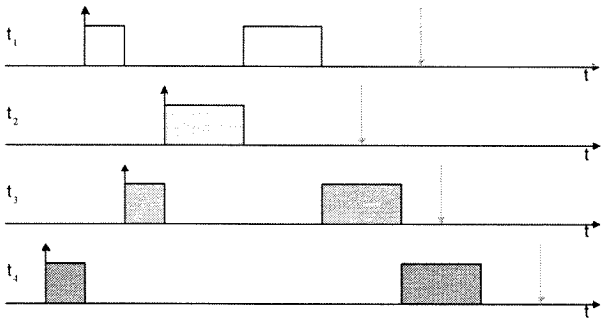

Figure 4 - Exemplo de um escalonamento de 4 tarefas usando o algoritmo LSF.

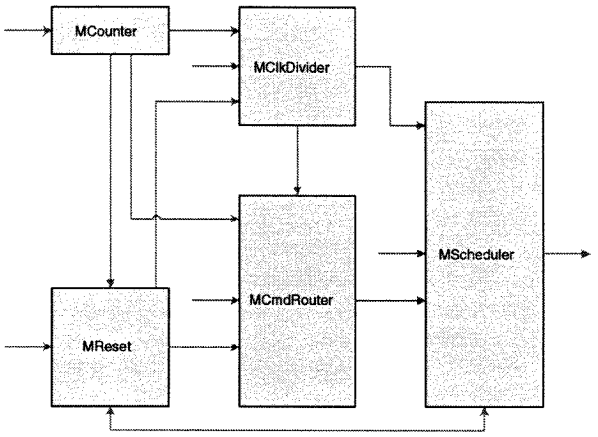

Figure 5 - Arquitectura do escalonador.

Tal como o algoritmo EDF, o algoritmo LSF é considerado óptimo entre todos os algoritmos de escalonamento, ou seja, se um conjunto de tarefas for escalonável, então é escalonável utilizando EDF, e permite alcançar uma taxa de ocupação do processador de 100%.

A figura 4 mostra um exemplo de 4 tarefas usando a política de escalonamento LSF.

III. ARQUITECTURA

A figura 5 ilustra a arquitectura do escalonador implementado, dando já uma ideia do fluxo de informação no sistema.

A figura 6 refere-se às características dinâmicas do escalonador, as entradas que este considera para sua configuração inicial bem como as saídas produzidas por este. Como entradas admite dois ficheiros de configuração, `scheduler.conf` e `reset.conf`, e alguns parâmetros passados na linha de comandos. Como saídas produz algum texto directamente no ecrã tal como um ficheiro VCD, que pode mais tarde ser visualizado com o programa `gtkwave` [3].

Mais à frente será dada mais informação acerca das entradas e saídas do escalonador, bem como uma descrição mais detalhada de cada módulo implementado.

A. *Tarefas*

Uma tarefa pode estar num dos seguintes estados:

- *Idle* (Inactiva)
- *Ready* (Pronta)

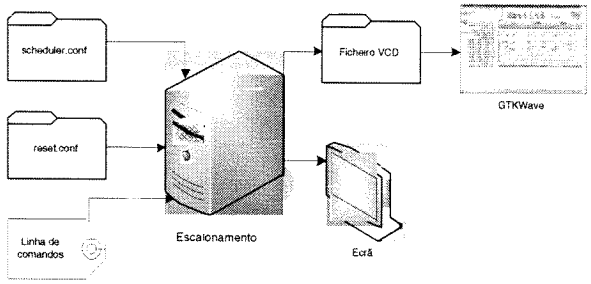

Figure 6 - Características dinâmicas do escalonador.

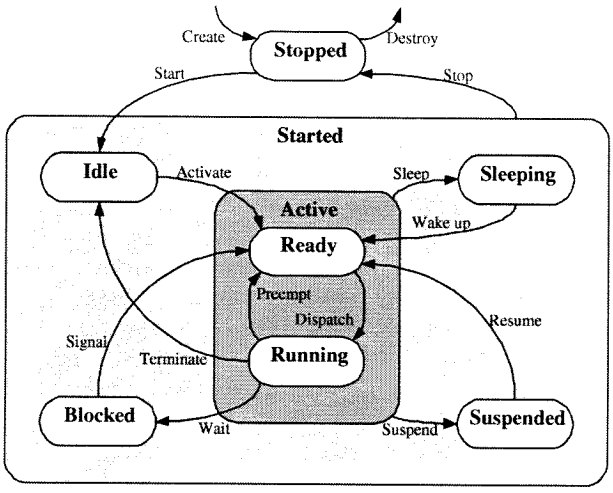

Figure 7 - Estados das tarefas e possíveis transições.

- *Running* (A executar)
- *Sleeping* (Adormecida)
- *Suspended* (Suspensa)
- *Stopped* (Parada)
- *Blocked* (Bloqueada)

Qualquer tarefa que se encontre num dos cinco primeiros estados (*Idle*, *Ready*, *Running*, *Sleeping* ou *Suspended*) diz-se iniciada (*Started*), caso contrário diz-se parada (*Stopped*). Uma tarefa que se encontre num dos estados *Ready* ou *Running* diz-se activa (*Active*). A figura 7 ilustra os vários estados em que uma tarefa se pode encontrar, bem como as transições permitidas.

Uma tarefa, quando é criada, é colocada no estado *Stopped*, permanecendo nesse estado até ser explicitamente iniciada. Depois de iniciada é colocada no estado *Idle* até ser alcançado o instante da primeira activação. Nessa altura transita para o estado *Ready*. Uma tarefa no estado *Ready* está pronta a executar. A passagem para o estado *Running* é feita pelo algoritmo de escalonamento através da atribuição de tempo de processador à tarefa. Uma tarefa que se encontre no estado *Running* pode ser interrompida pelo escalonador e passando para o estado *Ready*, ao que se chama preempção. Uma tarefa activa pode ser adormecida (passagem para o estado *Sleeping*) ou suspensa (passagem para o estado *Suspended*). O retorno ao estado *Ready* é automático no caso de uma tarefa adormecida e feito quando expirar o respectivo tempo. Uma tarefa suspensa deve ser reactivada explicitamente. Uma tarefa, depois de terminar, é colocada no estado *Idle* até à próxima activação. Uma tarefa activa é colocada no estado *Blocked* sempre que estiver à espera dum evento ou da libertação dum recurso partilhado. Este estado não se encontra implementado no escalonador. Em qualquer estado pode ser dada ordem de paragem a uma tarefa, transitando neste caso para o estado *Stopped*. Uma tarefa pode ser destruída, deixando de existir no sistema.

B. *Módulos*

A arquitectura do escalonador contempla diversos módulos sendo que cada um representa um possível componente físico, na perspectiva de simulação da implementação do escalonador em *hardware*.

Assim, os seguintes módulos estão presentes actualmente no escalonador:

- `MClkDivider` - Módulo responsável pela geração do relógio interno do escalonador de tempo real, que tem por base o *clock* interno do sistema;
- `MCounter` - Módulo encarregue da contagem do número de ciclos do *clock* interno do sistema já decorridos desde o início da simulação;
- `MReset` - Módulo responsável pelo botão de *reset* do sistema;
- `MScheduler` - Módulo no qual todo o escalonamento é feito, bem como todas as operações relativas aos blocos de controlo das tarefas (*Task Control Blocks* - TCB's);
- `MCmdRouter` - Módulo encarregue da interpretação dos comandos através dos quais se efectuam as operações relativas às tarefas.

IV. IMPLEMENTAÇÃO

A implementação do escalonador, feita na linguagem SystemC, assenta essencialmente em duas partes: a estrutura que define os blocos de controlo das tarefas e os diferentes módulos. Um outro aspecto é a possibilidade de configuração do sistema, discutida mais à frente nesta secção.

A. *O SystemC*

O escalonador desenvolvido foi implementado usando a linguagem de programação SystemC [4].

O SystemC é uma biblioteca da linguagem C++, que a dota de mecanismos que permitem simular a modelação de *hardware*. Assim, esta linguagem foi escolhida pois permite simular a implementação do escalonador em *hardware*, com todos os aspectos que lhe estão associados. O SystemC simula a execução concorrente dos módulos desenvolvidos e permite a utilização de *interfaces* tipo *hardware*, incluindo a possibilidade de tornar métodos sensíveis a determinados sinais, que despoletam a sua execução. Ao mesmo tempo, permite a composição de algoritmos de alto nível com algoritmos de baixo nível e algoritmos sequenciais com algoritmos paralelos, permitindo o refinamento do código até uma eventual implementação concreta em *hardware*.

B. A Estrutura sc_TCB

A estrutura sc_TCB define o bloco de controlo da tarefa (TCB - *Task Control Block*). Esta estrutura possui vários campos onde são armazenados os parâmetros de configuração e guardado o estado da respectiva tarefa. Do ponto de vista do escalonador implementado a execução das tarefas é modelada através da passagem do tempo de simulação, não existindo qualquer execução de código da tarefa. O tempo de execução de uma tarefa é determinado pelo simulador através dos parâmetros definidos pelo utilizador durante a criação da tarefa. Para cada tarefa presente no sistema existe apenas um TCB que contém a informação relativa aos seus parametros de execução.

A definição desta estrutura (ver figura 8) engloba diversos campos e um construtor, encarregue de assegurar que quando é criado um TCB este se encontra livre, não representando nenhuma tarefa.

Os campos que constam na estrutura são os seguintes:

- m_id - identificador atribuído ao TCB;
- m_state - estado da tarefa;
- m_period - período da tarefa;
- m_deadline - tempo limite de execução da tarefa;
- m_firstActivation - instante da primeira activação relativamente ao arranque da tarefa;
- m_activation - contador decrescente usado no escalonamento das tarefas. Possui o número de unidades temporais restantes para a (re)activação da tarefa;
- m_ponctuality - contador decrescente usado no escalonamento das tarefas. Armazena o número de unidades de tempo que a tarefa possui para terminar antes de ser atingida a sua *deadline*;
- m_bestCaseExecTime - número mínimo de unidades de tempo que a tarefa gastará com uma execução. Este parâmetro não está, normalmente, presente nos sistemas reais e é apenas utilizado na geração aleatória do tempo de execução da tarefa;
- m_worstCaseExecTime - número máximo de unidades de tempo que a tarefa gastará com uma execução. É utilizado na geração aleatória do tempo de execução da tarefa e também no escalonamento usando a política LSF;
- m_execCyclesCounter - contador decrescente que representa o número de unidades de tempo que faltam correr à tarefa nesta activação;
- m_cyclesExecuted - número de unidades de tempo que a tarefa já executou na última activação. É usado no algoritmo de escalonamento LSF para determinar o tempo livre da tarefa e assim a sua prioridade;
- m_sleepingCounter - contador decrescente que armazena o número de unidades de tempo restantes para a tarefa acordar. É utilizado para tarefas no estado *Sleeping*.

C. Módulos

Tal como já foi referido, a implementação do escalonador contempla diversos módulos. As subsecções seguintes descrevem sucintamente cada um dos módulos implementados.

```
typedef struct sc_TCB
{
   unsigned int m_id;

   unsigned int m_state;

   unsigned int m_period;
   unsigned int m_deadline;
   unsigned int m_firstActivation;

   int m_activation;
   int m_ponctuality;

   int m_bestCaseExecTime;
   int m_worstCaseExecTime;

   int m_execCyclesCounter;
   int m_cyclesExecuted;

   int m_sleepingCounter;

   sc_TCB()
   {
      m_state = FREE;
   }
}SC_TCB;
```

Figure 8 - Definição da estrutura sc_TCB.

C.1 O Módulo MClkDivider

O módulo MClkDivider encarrega-se da criação do relógio interno do escalonador. Para isso tem em conta um parâmetro de configuração do sistema, designado por factor de divisão, que vai especificar quantas vezes mais lento é o relógio interno do escalonador relativamente ao relógio do sistema. Este módulo compreende as seguintes entradas:

- Clk - relógio de simulação;
- Clk_Count - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação;
- Factor - factor de divisão para criação do relógio interno do escalonador a partir do relógio do sistema;
- Rst - botão de reset.

O módulo produz ainda as seguintes saídas:

- Tick - relógio interno do escalonador;
- Tick_Count - tempo absoluto do escalonador. Número de ciclos de relógio do escalonador passados desde o início da simulação.

Este módulo está representado na figura 9.

C.2 O Módulo MCounter

O módulo MCounter encarrega-se apenas do incremento dum contador que representa o número de ciclos do relógio do sistema já ultrapassados desde o início da simulação.

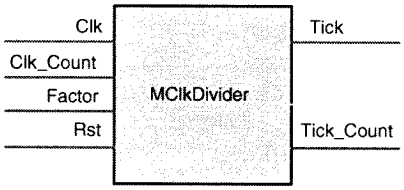

Figure 9 - Módulo `MClkDivider`.

Figure 10 - Módulo `MCounter`.

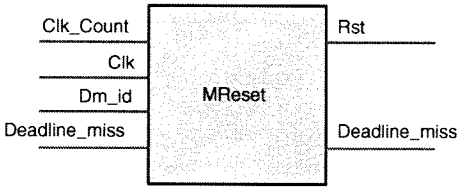

Figure 11 - Módulo `MReset`.

Este módulo compreende uma entrada:

- `Clk` - relógio de simulação.

O módulo produz ainda uma saída:

- `Clk_Count` - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação.

Este módulo está representado na figura 10.

C.3 O Módulo `MReset`

O módulo `MReset` tem por função a manipulação do *reset* do escalonador. Principalmente, encarrega-se da leitura do ficheiro de configuração `reset.conf` e activação ou desactivação a tempo do *reset* como especificado neste.

Também aceita como entrada uma *flag* que sinaliza uma falha duma restrição temporal, situação na qual faz um *reset* do escalonador. Este módulo compreende as seguintes entradas:

- `Clk` - relógio de simulação;
- `Clk_Count` - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação;
- `Deadline_miss` - *flag* que sinaliza uma falha duma restrição temporal;
- `Dm_id` - identificador da tarefa que violou a sua restrição temporal.

O módulo produz ainda as seguintes saídas:

- `Deadline_miss` - *flag* que sinaliza uma falha duma restrição temporal;
- `Rst` - sinal de *reset*.

Este módulo está representado na figura 11.

C.4 O Módulo `MScheduler`

O módulo `MScheduler` é o centro do escalonador implementado. É este módulo que se encarrega do escalonamento das tarefas propriamente dito, aplicando o algoritmo de escalonamento escolhido ao conjunto de tarefas presente no sistema. É também neste módulo que se processam as transições de estado das tarefas tal como os comandos relacionados com estas. Este módulo compreende as seguintes entradas:

- `Tick` - relógio interno do escalonador;
- `Tick_Count` - tempo absoluto do escalonador. Número de ciclos de relógio do escalonador passados desde o início da simulação;
- `Algorithm` - algoritmo a ser utilizado pelo escalonador. É determinado através de um parâmetro passado na linha de comandos;
- `Rst` - sinal de `reset`;
- `Activate` - sinal que indica que um novo comando deve ser executado. Os seguintes sinais caracterizam o comando:
  - `Cmd` - comando a ser executado;
  - `T_id` - identificador da tarefa à qual o comando se refere;
  - `Period` - período da tarefa;
  - `Deadline` - *deadline* da tarefa;
  - `FirstActivation` - tempo da primeira activação da tarefa;
  - `BestCaseExecTime` - tempo mínimo de execução da tarefa;
  - `WorstCaseExecTime` - tempo máximo de execução da tarefa.

O módulo produz ainda as seguintes saídas:

- `Handle` - identificador da tarefa agendada para execução em cada instante;
- `State[]` - vector com os estados actuais de cada tarefa. Apenas foi incluído para poder facultar esta informação ao utilizador;
- `Deadline_miss` - *flag* que sinaliza uma falha duma restrição temporal;
- `Dm_id` - identificador da tarefa que violou a sua restrição temporal. Apenas foi incluído para poder facultar esta informação ao utilizador.

Este módulo está representado na figura 12.

C.5 O Módulo `MCmdRouter`

O módulo `MCmdRouter` encarrega-se da leitura do ficheiro de configuração `scheduler.conf` e da passagem no instante de execução dum comando dos parâmetros deste ao módulo `MScheduler`. Este módulo compreende as seguintes entradas:

- `Clk` - relógio de simulação;
- `Clk_Count` - tempo absoluto de simulação. Número de ciclos de relógio passados desde o início da simulação;
- `Tick_Count` - tempo absoluto do escalonador. Número de ciclos de relógio do escalonador passados desde o início da simulação;

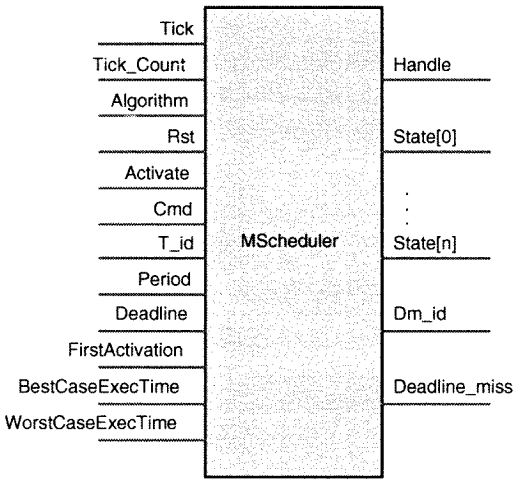

Figure 12 - Módulo `MScheduler`.

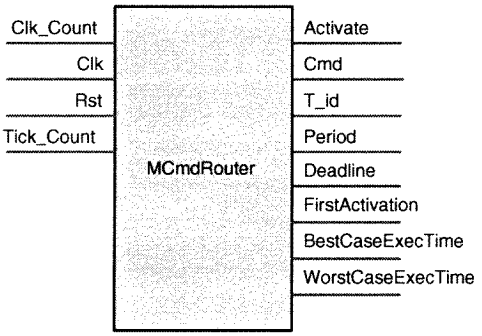

Figure 13 - Módulo `MCmdRouter`.

- `Rst` - sinal de `reset`.

O módulo produz ainda as seguintes saídas:

- `Activate` - sinal que indica que um novo comando deve ser executado. Os seguintes sinais caracterizam o comando:
  - `Cmd` - comando a ser executado;
  - `T_id` - identificador da tarefa à qual o comando se refere;
  - `Period` - período da tarefa;
  - `Deadline` - *deadline* da tarefa;
  - `FirstActivation` - tempo da primeira activação da tarefa;
  - `BestCaseExecTime` - tempo mínimo de execução da tarefa;
  - `WorstCaseExecTime` - tempo máximo de execução da tarefa.

Este módulo está representado na figura 13.

D. Configuração

O escalonador implementado é passível de ser configurado. Nas secções seguintes vão ser apresentadas as opções de configuração e modo de utilização destas.

D.1 O Ficheiro `scheduler.conf`

O ficheiro `scheduler.conf` é o meio através do qual se passam os comandos relativos a tarefas ao escalonador. O ficheiro pode ter no início texto que não será avaliado, devendo-se introduzir um ';' no fim a partir do qual deverá estar um comando por linha, eventualmente com linhas em branco a separá-los. Cada comando deverá ser precedido pelo instante no qual se pretende a sua execução, em ciclos de relógio do escalonador. Os comandos disponíveis são os seguintes:

- `create` - Sintaxe: `<t> create <t_id P D PA min max>`
  Comando que permite a criação duma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador pretendido para a tarefa a criar;
  - `P` - período a atribuir à tarefa;
  - `D` - *deadline* para a tarefa;
  - `PA` - instante da primeira activação da tarefa;
  - `min` - número mínimo de unidades de tempo que a tarefa gastará com uma execução;
  - `max` - número máximo de unidades de tempo que a tarefa gastará com uma execução.
- `destroy` - Sintaxe: `<t> destroy <t_id>`
  Comando que permite eliminar uma tarefa do sistema. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a destruir.
- `change` - Sintaxe: `<t> change <t_id P D min max>`
  Comando utilizado para alterar as características duma tarefa já presente no sistema. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a alterar;
  - `P` - período a atribuir à tarefa;
  - `D` - *deadline* para a tarefa;
  - `min` - número mínimo de unidades de tempo que a tarefa gastará com uma execução;
  - `max` - número máximo de unidades de tempo que a tarefa gastará com uma execução.
- `start` - Sintaxe: `<t> start <t_id>`
  Comando para arrancar uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a arrancar.
- `stop` - Sintaxe: `<t> stop <t_id>`
  Comando que permite parar uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a parar.
- `sleep` - Sintaxe: `<t> sleep <t_id C>`
  Comando utilizado para adormecer uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a adormecer;
  - `C` - número de ciclos ao longo dos quais a tarefa estará adormecida.

- `suspend` - Sintaxe: `<t> suspend <t_id>`
  Comando que permite suspender uma tarefa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a suspender.
- `resume` - Sintaxe: `<t> resume <t_id>`
  Comando utilizado para retomar uma tarefa suspensa. Os parâmetros do comando são:
  - `t` - tempo no qual o comando deverá ser executado;
  - `t_id` - identificador da tarefa a retomar.

D.2 O Ficheiro `reset.conf`

O ficheiro `reset.conf` disponibiliza a interacção com o botão de *reset* do escalonador. Tal como o ficheiro `scheduler.conf`, poderá ter um cabeçalho seguido de um ';', após o qual deverá estar um número por linha, eventualmente com linhas em branco entre eles, que representarão os instantes nos quais o *reset* deverá mudar de valor, começando activo, em ciclos do relógio de simulação.

D.3 Invocação

A linha de comandos permite ainda a configuração de mais alguns parâmetros. Na sequência do nome do programa poder-se-á especificar o algoritmo de escalonamento que se pretende utilizar (RM, EDF ou LSF) seguido do factor de divisão para a criação do relógio interno do escalonador e por fim o nome para o ficheiro VCD (sem a extensão `.vcd`). A execução escrevendo `default` a seguir ao nome do programa decorre com os parâmetros com os valores por defeito, sendo o algoritmo de escalonamento o EDF, o factor de divisão 4 e gerado um ficheiro VCD com o nome `Scheduler.vcd`.

V. EXEMPLOS DE APLICAÇÃO

Nesta secção irá ser descrita a sequência típica de passos para a execução de uma simulação de um escalonamento, e visualização dos resultados.

Primeiro é necessário editar o ficheiro `scheduler.conf` introduzindo os comandos pretendidos relacionados com o conjunto de tarefas que se pretende simular. Na figura 14 pode-se ver a configuração do ficheiro `scheduler.conf` usada para este exemplo.

O segundo passo consiste na edição do ficheiro `reset.conf` segundo as necessidades da simulação. Para este exemplo introduziu-se apenas um número neste ficheiro, que representou o instante a partir do qual o *reset* deixou de estar activo.

Feito isto, pode-se correr a simulação. Neste exemplo foi usado o algoritmo de escalonamento EDF, um factor de divisão do relógio para criação do tick do sistema de 4 e foi indicado que o ficheiro VCD a ser criado deveria ser nomeado `exemplo.vcd`, como ilustra o comando que foi executado:

```
scheduler.x EDF 4 exemploEDF
```

De seguida surgem no ecrã mensagens relativas ao escalonamento efectuado (figura 15).

```
0 create 1 100 100 0 4 5
0 create 2 27 27 0 7 8
0 create 3 48 48 0 4 5
0 create 4 59 59 0 9 9
0 create 5 14 14 0 5 5
100 sleep 3 20
200 suspend 2
500 stop 3
600 stop 4
700 change 4 50 50 4 9
900 start 4
950 resume 2
1000 destroy 3
1300 destroy 4
1500 change 1 120 120 2 5
```

Figure 14 - Ficheiro `scheduler.conf` usado no exemplo.

```
A tarefa nº 2 foi agendada para execução.
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
A tarefa nº 5 foi agendada para execução.
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME
IDLE TIME

Tarefa nº 1 alterada.

A tarefa nº 4 foi agendada para execução.
A tarefa nº 2 foi agendada para execução.
A tarefa nº 5 foi agendada para execução.
A tarefa nº 2 foi agendada para execução.
A tarefa nº 4 foi agendada para execução.
A tarefa nº 5 foi agendada para execução.
A tarefa nº 1 foi agendada para execução.
```

Figure 15 - *Output* no ecrã gerado pela simulação do escalonamento.

Estes resultados servem para dar uma ideia de como a simulação decorreu. No entanto existe a possibilidade de analisar o que decorreu ao longo da simulação com mais detalhe. O ficheiro VCD produzido contém muito mais informação e pode ser consultado executando o seguinte comando:

```
gtkwave exemploEDF.vcd
```

Aí é possível encontrar a evolução ao longo do tempo de simulação de diversos sinais (figura 16). São estes:

- relógio e contador deste;
- *tick* do sistema e contador deste;
- *reset*;
- identificador da tarefa em execução;
- *flag* que assinala uma violação de uma *deadline*;
- identificador da tarefa que violou a sua *deadline*;
- estados das tarefas presentes no sistema.

A. *Análise dos resultados*

Para além do exemplo descrito acima foram corridos outros dois exemplos, utilizando os mesmo ficheiros de configuração, desta vez indicando ao escalonador para

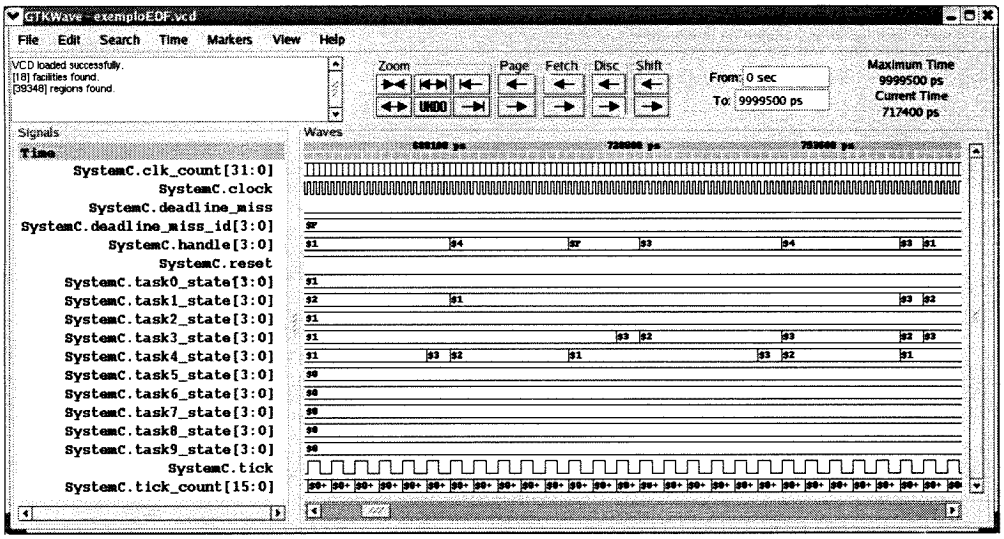

Figure 16 - Visualização do ficheiro VCD gerado na simulação utilizando a política EDF.

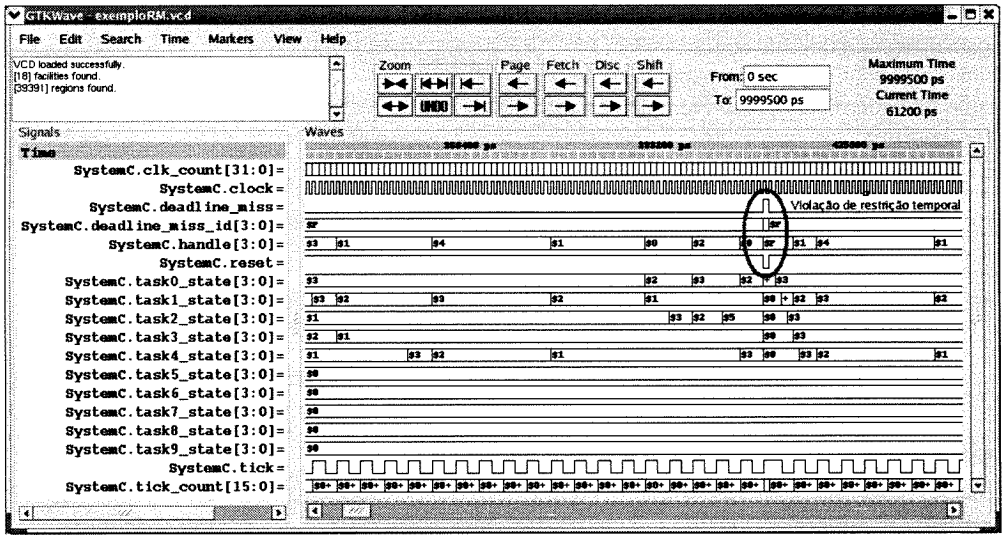

Figure 17 - Visualização do ficheiro VCD gerado na simulação utilizando a política RM.

agendar as tarefas segundo as outras duas políticas suportadas: RM e LSF. As figuras 17 e 18 mostram uma parte do ficheiro VCD criado em cada um dos casos.

O conjunto de tarefas utilizado nos exemplos poderia ser executado num processador com uma taxa de ocupação entre 89% e 96%. No entanto, utilizando a política de escalonamento RM ocorreram violações de restricções temporais. Isto acontece devido ao facto desta política, que utiliza prioridades fixas, não conseguir garantir que atinja uma taxa de ocupação do processador de 100% como as outras duas políticas implementadas, que se baseiam em prioridades dinâmicas.

Comparando as políticas de prioridades dinâmicas pode-se ver que ambas conseguiram escalonar o conjunto de tarefas. Apenas de registar que usando a política LSF se verificam mais preempções do que com a EDF, sendo que este facto poderá ser relevante num sistema real uma vez que as comutações de contexto que acontecem nessas alturas levam algum tempo a serem efectuadas.

VI. CONCLUSÃO E TRABALHO FUTURO

A implementação do escalonador descrito neste artigo permite a simulação do escalonamento de tarefas concorrentes processado por um módulo de *hardware*. Neste momento a sua funcionalidade consiste apenas na simulação do escalonamento de conjuntos de tarefas *hard real-time*, que se podem configurar, usando um dos algoritmos de escalonamento incluídos, e visualização do resultado deste escalonamento quer através do output no ecrã mas principalmente através do ficheiro VCD gerado, que permite a visualização e análise de mais informação. No entanto este trabalho não se apresenta na sua versão final, sendo possível desenvolvê-lo na direcção de algumas ideias que já surgiram para trabalho futuro:

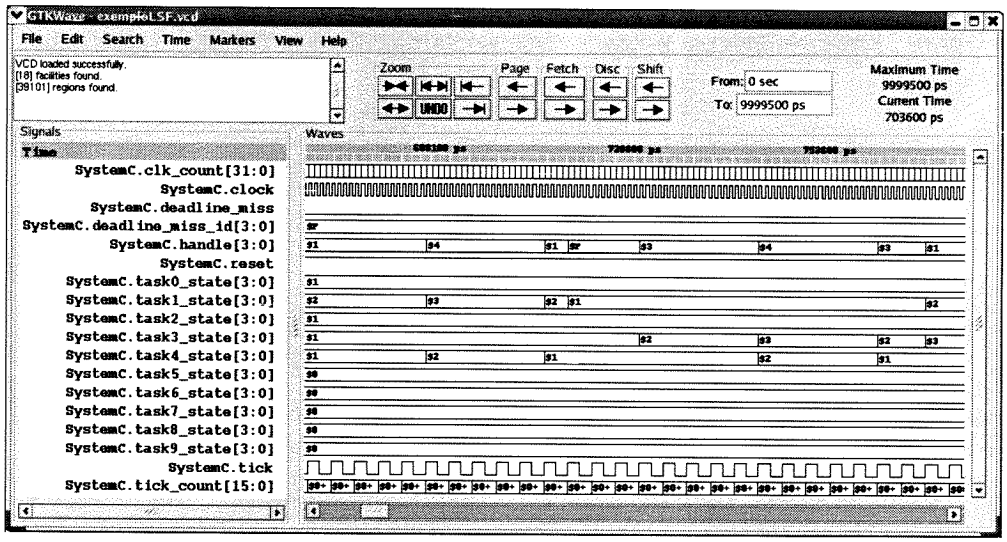

Figure 18 - Visualização do ficheiro VCD gerado na simulação utilizando a política LSF.

- Suporte para diferentes tipos de tarefas;
- Implementação de análise de escalonabilidade;
- Refinamento do código para possível implementação futura em *hardware*.

REFERENCES

[1] Arnaldo Oliveira e Luís Almeida, "Ensaio sobre os Sistemas de Tempo Real", Janeiro de 2004.

[2] Arnaldo Oliveira, Luís Almeida, e Valery Sklyarov, "OReK - Um Executivo de Tempo Real Orientado por Objectos", Janeiro de 2004.

[3] "GTKWave".
URL: *http://www.cs.man.ac.uk/apt/tools/gtkwave/*

[4] "SystemC".
URL: *http://www.systemc.org*

[5] Giorgio Buttazo, "Hard Real-Time Computing Systems: Predictadle Scheduling Algorithms and Applications", 1997.