Relógio Virtual Estritamente Crescente para o Computador Raspberry Pi

Edilson C. Corrêa[1], **Diego L. C. Dutra**[1], **Claudio L. Amorim**[1]

[1]COMPASSO/Programa de Engenharia de Sistemas e Computação
COPPE/Universidade Federal do Rio de Janeiro - UFRJ
Caixa Postal 68511 – 21941-972 – Rio de Janeiro – RJ – Brasil

{ecc,ddutra,amorim}@cos.ufrj.br

Abstract. *Wireless sensor network projects's require energy-efficient computing platforms such as the Raspberry Pi (RPI). To achieve this, a typical mechanism that such platforms support is the DVFS (Dynamic Voltage and Frequency Scale). However, the use of the DVFS mechanism can negatively impact the performance of the clock circuits of the RPI platform as a trade-off, to the energy efficiency it can provide. This paper proposes a virtual clock, RVEC, as a new solution that utilizes the cycle counter of the RPIs ARM processor while ensuring that the timekeeping is strictly-increasing and accurate. The RVEC solution also provides nanosecond time resolution with an access cost equivalent to that of system clocks.*

Resumo. *Projetos de redes de sensores sem fio requerem plataformas computacionais energeticamente eficientes como a plataforma Raspberry Pi (RPI). Para esse fim, um mecanismo típico que tais plataformas oferecem é o DVFS (Dynamic Voltage and Frequency Scale). Porém, o uso desse mecanismo pode afetar negativamente o desempenho dos circuitos contadores de tempo da plataforma RPI, em oposição a sua eficiência energética. Este trabalho propõe o relógio RVEC como uma nova solução que viabiliza o uso do contador de ciclos do processador ARM da plataforma RPI enquanto que garante a temporização ser estritamente crescente e precisa. A solução RVEC também provê resolução de nanosegundos com um custo de acesso equivalente aos dos relógios de sistemas.*

1. Introdução

Em sistemas computacionais, os relógios de sistemas proveem os meios para a medição correta e precisa da passagem do tempo, fundamental para a depuração e a avaliação de desempenho de aplicações, redes de comunicação e serviços do sistema operacional. Tradicionalmente, os relógios do sistema se baseiam em um contador de ciclos invariante no tempo e implementado em *hardware* (e.g., o *Time Stamp Counter* (TSC)). Esta solução, porém, tem sido adaptada em sistemas computacionais atuais que utilizam processadores *multicore* e também ofereçam o mecanismo de variação dinâmica de frequência e voltagem (*Dynamic Voltage and Frequency Scaling* (DVFS)) para controle eficiente do consumo energético.

Por exemplo, no Linux, os relógios de sistema foram adaptados para receber interrupções periódicas geradas por circuitos auxiliares em *hardware* de contadores de

ciclos invariantes, porém de menor frequência presentes nas novas arquiteturas *multicore*. Neste caso, esses relógios oferecem menor precisão e sua acurácia é definida pelo grau de estabilidade dos períodos entre interrupções. Na prática, além de eventuais perdas de interrupções também pode ocorrer variação na duração do período entre elas, tornando necessária a utilização de mecanismos de ressincronização de relógios como o do NTP [Mills 1992] ou TPSN [Fan et al. 2004]. Complementarmente, o sistema operacional pode utilizar mecanismos de interpolação para aumentar a precisão oferecida pelos relógios de sistema, utilizando o contador de ciclos do processador. O resultado é que a utilização desses mecanismos acaba por inviabilizar a manutenção de um relógio de sistema com a propriedade de contagem de tempo estritamente crescente e precisa (ECP) [Corrêa 2014]. Ademais, a introdução de mecanismos de ressincronização representa uma sobrecarga à eficiência energética desses ambientes computacionais em geral, enquanto que o tratamento de interrupções aumentam as sobrecargas em sistemas embarcados dependentes de energia de bateria.

Neste trabalho, investigamos a proposta do relógio virtual denominado RVEC [Dutra et al. 2013] como alternativa para os relógios de sistema presentes na placa Raspberry Pi [Broadcom] construída com processadores ARM, que são amplamente utilizados em sistemas computacionais embarcados [Simunic et al. 2001] com DVFS, como é o caso do processador ARM1176JZF-S [Khan et al. 2012] utilizado em placas Raspberry Pi. Neste caso, o uso de DVFS inviabiliza a utilização direta do contador de ciclos do processador (CCNT, na arquitetura ARM) para aferição do tempo.

Especificamente, este trabalho descreve e avalia uma implementação do RVEC para a plataforma ARM/Raspberry quando submetido à alteração da frequência de operação do processador. As principais contribuições deste trabalho são:

- Desenvolvimento e avaliação experimental de uma implementação RVEC em um *kernel* Linux na plataforma ARM/Raspberry Pi com suporte ao mecanismo DVFS;
- Validação do relógio virtual RVEC como uma solução de relógio de sistema para o sistema computacional ARM/Linux.

Este artigo está organizado da seguinte maneira. A Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve o método usado pelo RVEC e a implementação do RVEC no computador Raspberry Pi. Na Seção 4, discutimos os resultados da validação da implementação efetuada no computador Raspberry Pi. A Seção 5 apresenta nossas conclusões e propõe trabalhos futuros.

2. Trabalhos Relacionados

Dutra et al. [Dutra et al. 2013] apresenta a proposta e avaliação do RVEC como uma alternativa aos relógios de sistema existentes por atender a propriedade de contar o tempo de forma estritamente crescente e precisa. Os autores mostraram que os nós em um *cluster* de computadores usando RVEC podem permanecer sincronizados globalmente após a sincronização inicial de seus RVECs por meio do uso de um algoritmo cliente-servidor de sincronização remoto. Veitch et al. [Veitch et al. 2009] apresentam o RADClock, um sistema construído sobre os relógios do sistema existentes ou contadores de tempo de ciclos, que fornece informações sobre o tempo global, bem como o tempo

global absoluto para a sincronização de nós da rede. No entanto, RADClock depende de NTP para ressincronização periódica. Além disso, o uso do TSC limita a sua aplicabilidade em processadores com múltiplos núcleos.

Tian et al. [Tian et al. 2008] tratam do desenvolvimento de um relógio global usando o TSC como relógio de base juntamente com um algoritmo de sincronização remoto que é semelhante ao do NTP. Devido ao uso direto de TSC, tal relógio global não pode funcionar corretamente com processadores que possuam DVFS ou vários núcleos. Souza et al. [de Souza et al. 2008], propuseram um *hardware* auxiliar de sincronização de rede utilizando um gerador de pulsos remoto. Em um *cluster* de computadores, esta sincronização de rede assegura que todos os nós do *cluster* recebam simultaneamente o pulso de oscilador remoto e o usa para atualizar o relógio do nó local sem envolver o sistema operacional. Apesar de garantir a contagem estritamente crescente e precisa de relógios locais, a solução proposta depende de *hardware* dedicado.

3. Relógio Virtual Estritamente Crescente - RVEC

O RVEC é um relógio de sistema aderente à propriedade ECP, o método utiliza o contador de ciclos do núcleo de processamento (*core*) e a memória principal para criar e manter as informações necessárias para o funcionamento do relógio virtual. O contador de ciclos do núcleo foi escolhido por operar sem o uso de interrupções e pela precisão oferecida por ele, tendo em vista que ele opera na mesma frequência que o núcleo e comumente é alimentado por um oscilador de alta precisão e estabilidade. Por ser interno ao núcleo e operar na frequência do mesmo, a taxa de atualização deste contador está sujeita a variações provocadas pelas mudanças na frequência de operação do núcleo de processamento que oferecem suporte à DVFS, impactando assim a correta contagem do tempo.

O RVEC utiliza uma estrutura de dados contendo dois campos: o contador_base que armazena o último valor do número de ciclos lido no registrador do núcleo e, o campo tempo_consolidado que armazena o intervalo de tempo decorrido do instante da criação do RVEC até o momento da última leitura do registrador que foi armazenada no campo contador_base. Sendo assim, a estrutura de dados deve ser atualizada sempre que ocorrer uma alteração de frequência. A Figura 1 apresenta conceitualmente o procedimento de atualização utilizado pelo RVEC.

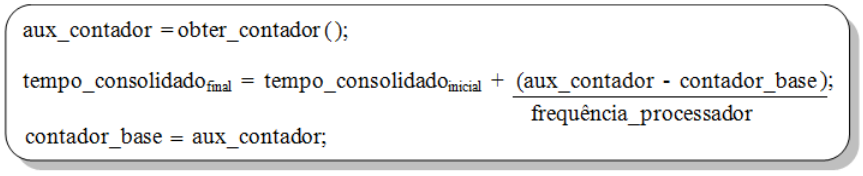

Figura 1. Procedimento de manutenção do RVEC quando do uso do DVFS

A Figura 2 apresenta conceitualmente o procedimento de manutenção do RVEC em sistemas computacionais que suportam o DVFS. Nela é possível observar que na iminência da mudança da frequência, o valor lido no contador de ciclos é armazenado no campo contador_base e o campo tempo_consolidado é atualizado ao final. Por exemplo, a

Figura 2 ilustra a alteração na frequência de operação de um núcleo de processamento que encontra-se operando a $800\ MHz$ e passará a operar a $400\ MHz$, no instante em que o contador de ciclos é igual a 4. Utilizando o procedimento mostrado na Figura 1 o campo contador_base do RVEC será atualizado com o valor 4 e o campo tempo_consolidado armazena o valor $5\ ns$.

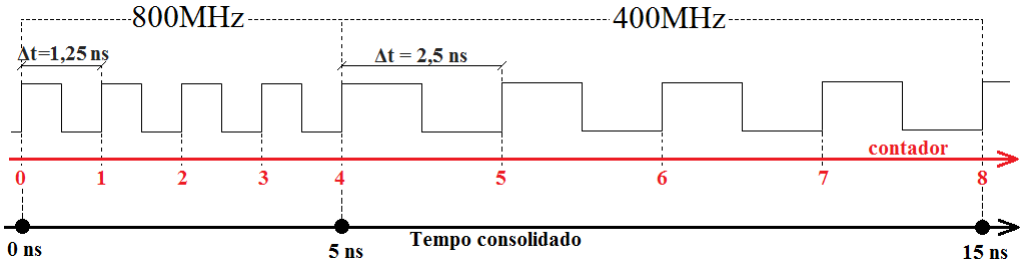

Figura 2. Cálculo da contagem do tempo com a mudança da frequência do núcleo

A Figura 3 apresenta o procedimento de leitura do RVEC. Assim, tomando como base o exemplo da Figura 2, a leitura do RVEC no instante em que o contador de ciclos chegar a 8, retornará o valor $15\ ns$. A combinação da estrutura de dados do RVEC e os procedimentos de manutenção e leitura do RVEC criam uma abstração de uma contagem de tempo aderente à propriedade ECP.

```
aux_contador = obter_contador() - contador_base ;

tempo_consolidado_final = tempo_consolidado_inicial + aux_contador / frequência_processador ;
```

Figura 3. Procedimento de leitura do RVEC de um núcleo

3.1. RVEC no Raspberry Pi

O sistema computacional Raspberry Pi tem sido utilizado em diferentes projetos de sensoriamento [Nagy and Gingl 2013, Neves and Matos 2013], demonstrando sua viabilidade como uma plataforma para construção de redes de sensores sem fios [Vujovic and Maksimovic 2014]. Nesses ambientes, um problema importante é a eficiência energética da plataforma. Como visto na Seção 1, o uso do DVFS permite reduzir o consumo energético em momentos de baixo uso do dispositivo. Contudo, a utilização deste tipo de mecanismo acaba impactando alguns dos circuitos que são utilizados para a construção dos relógios de sistemas. A implementação do RVEC no Linux utilizado pelo Raspberry Pi, representa uma solução a este problema permitindo o uso do contador de ciclos do processador ARM, o *Cycle Counter Register* (CCNT) [ARM], mesmo na presença de operações de alteração de frequência.

A Figura 4 apresenta como o CCNT é acessado pelo núcleo de processamento. O acesso ao contador é feito pela leitura do registrador c15, presente no coprocessador CP15. O CCNT é sempre acessível dentro do *kernel* do sistema operacional e a leitura deste registrador é feita através da instrução MCR, onde os campos devem ter os seguintes dados:

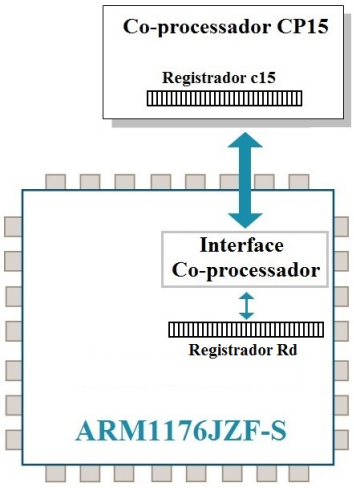

Figura 4. Processador ARM1176JZF-S e o Co-processador CP15

- Opcode_1 definido como 0;
- CRn definido como c15;
- CRm definido como c12; e
- Opcode_2 definido como 1.

Observe, contudo, que o CCNT é um contador de 32 bits, o que em um núcleo operando a frequência de 1 GHz provocaria um *overflow* a cada $4,295\ s$. Assim o uso desse contador para aferir intervalos de tempo superior a $4,3\ s$ violaria a propriedade ECP. No Raspberry Pi, esta limitação é solucionada através da expansão do contador de 32 bits para 63 bits utilizando a macro $cnt32\_to\_63$ presente no *kernel* do Linux (include/linux/cnt32_to_63.h), que na corrente implementação é executada durante o procedimento de escalonamento de tarefas.

A estrutura de dados **struct tb** apresentada na Figura 5 é a implementação da estrutura de dados conceitual do RVEC descrita no início desta seção, sendo que o campo base_counter é o contador_base e o campo age_time_ns representa o tempo_consolidado. Para o correto funcionamento do RVEC, ela deve ser incluída na estrutura de dados que representa a fila de processamento do núcleo (*struct rq*) e na estrutura de dados que representa as tarefas do sistema (*struct task_struct*).

```
struct tb{
   u64 base_counter;
   u64 age_time_ns;}
```

Figura 5. Estrutura de Dados do RVEC

A Figura 6 apresenta a implementação final do RVEC no Linux utilizada na plataforma ARM e, para facilitar à visualização, a figura apresenta um processador hipotético com dois núcleos. Nessa figura, é possível observar os dois subsistemas do Linux que sofreram alterações para o correto funcionamento do RVEC. Essa dupla modificação no *kernel* do Linux é necessária para garantir que a alteração na frequência do processador não cause um efeito cascata de atualização em todos os RVEC das aplicações que executam no núcleo de processamento. Esta implementação da técnica RVEC permite di-

ferentes tarefas no sistema verificar os seus RVECs associados através da chamada de sistema clock_gettime(), necessitando apenas a aplicação utilizar o identificador de relógio CLOCK_RVEC como o parâmetro de entrada.

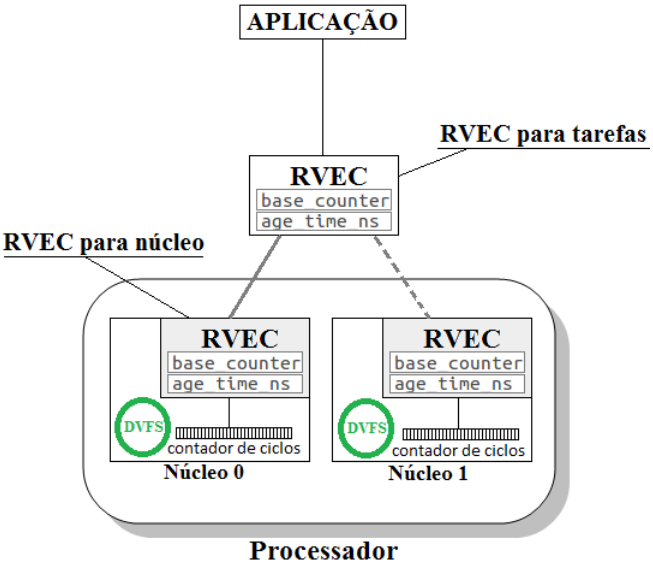

Figura 6. RVEC para núcleos e para tarefas

A Figura 6, anteriormente apresentada, não possibilita mostrar as alterações necessárias no *drive* de DVFS do Linux que possibilitam o correto funcionamento do RVEC, sendo esses passos apresentados na Figura 7. As modificações vistas na Figura 7 foram feitas no *driver* **bcm2835-cpufreq** usado pela família de processadores ARM11, a qual o processador ARM1176JZF-S utilizado pelo Raspberry Pi pertence. A chamada de função é usada para alterar a frequência do núcleo. A função original bcm2835_cpufreq_set_clock() é estendida em duas funções RVEC, rvec_cpu_freq_change_pre() e rvec_cpu_freq_change_pos(). A função rvec_cpu_freq_change_pre() lida com a configuração anterior a mudança da frequência (pre-change) para o relógio RVEC no subsistema de gerenciamento do núcleo, executando uma implementação de atualização do RVEC. A função rvec_cpu_freq_change_pos() define (após a mudança da frequência) a soma do tempo gasto desde a execução da função rvec_cpu_freq_change_pre() mais o tempo gasto com a frequência atual e armazena o valor atual do CCNT do núcleo no base_counter.

4. Validação Experimental

A seguir são apresentados os resultados da avaliação experimental do Relógio Virtual Estritamente Crescente (RVEC) para a plataforma computacional Raspberry Pi que foi proposto neste trabalho. A Tabela 1 apresenta um resumo das características da versão da plataforma que foi utilizada, assim como a do software. Os resultados são apresentados com os valores da média e do desvio-padrão em microssegundos (μs) para um intervalo de confiança de $99,9\%$.

4.1. Avaliando a propriedade ECP do RVEC

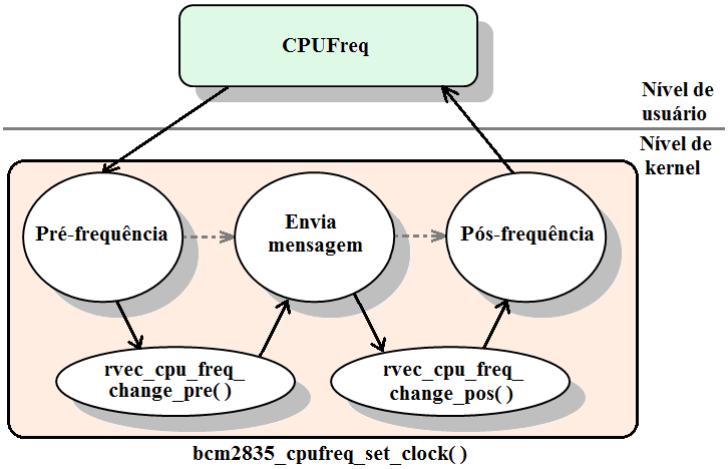

Figura 7. Driver bcm2835-cpufreq modificado para o RVEC

Tabela 1. Características técnicas da plataforma Raspberry Pi

Componente	Descrição
Raspberry Pi	Modelo B+
Processador	ARM1176JZF-S
Circuito de Relógio	STC
Linux	3.2.27+

A avaliação da propriedade ECP do RVEC, passa por confirmar experimentalmente a aderência desta propriedade para o CCNT. O *microbenchmark* utilizado para o experimento implementou um laço com um bloco de 12 operações de soma por iteração e executa este laço 400 vezes. O experimento foi executado 100 vezes. Avaliou-se o tempo de execução do *microbenchmark* usando uma instrução de acesso (MRC) para leitura do registrador CCNT de 32 bits. A Figura 8 apresenta uma das execuções deste experimento, sendo possível observar que nela não ocorreu nenhuma violação da propriedade de crescimento estrito do tempo. A avaliação da propriedade ECP do CCNT mostrou um tempo médio de execução de $361,91\ \mu s$ e desvio padrão de $18,33\ \mu s$ com frequência do processador fixa em $800MHz$. O coeficiente de variação do experimento foi $0,051$, o que fornece um indicativo preliminar da estabilidade do contador de ciclos, assim a propriedade o CCNT foi aderente à propriedade ECP em todas as execuções do experimento.

A Figura 9 apresenta uma das execuções realizadas para a avaliação da propriedade ECP do RVEC utilizando o mesmo *microbenchmark* do experimento anterior com o CCNT, sendo que este experimento também é executado 100 vezes. A avaliação da propriedade ECP do RVEC mostrou o tempo médio de execução de $372,88\ \mu s$ e desvio de $20,02\ \mu s$ com frequência do processador fixa em $800MHz$. O coeficiente de variação do experimento foi $0,054$ o que fornece um indicativo preliminar da estabilidade desta implementação do RVEC e, portanto, que o RVEC é aderente à propriedade ECP. É importante salientar que o aumento observado no coeficiente de variação é dado pelo maior número de instruções necessárias para acessar o relógio RVEC.

A Figura 10 apresenta a avaliação da propriedade ECP do RVEC ao usar o me-

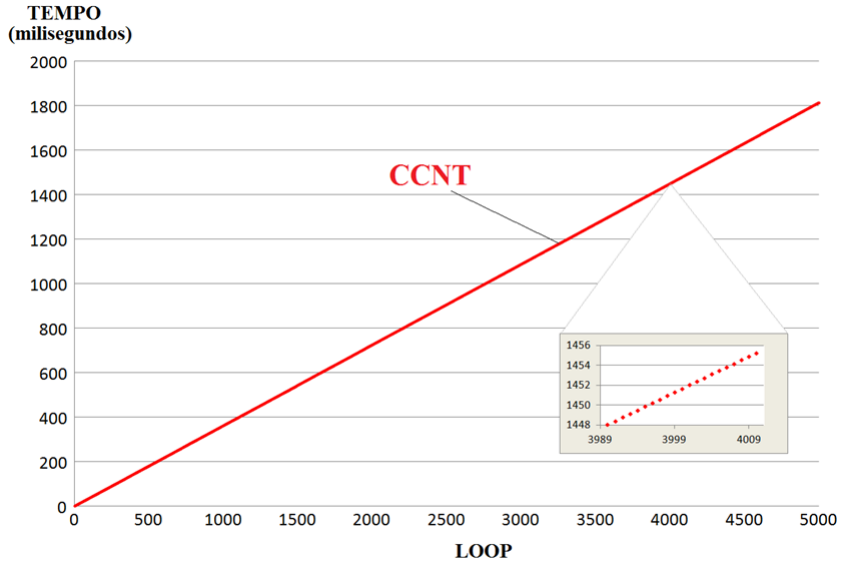

Figura 8. Propriedade ECP do CCNT

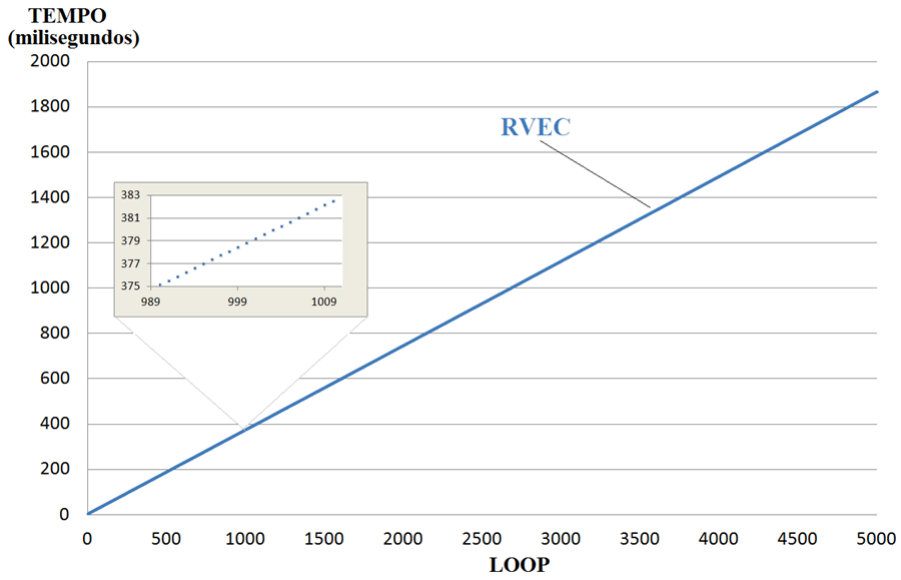

Figura 9. Propriedade ECP RVEC

canismo de DVFS para alteração da frequência do processador. O *microbenchmark* utilizado contém um laço com 5.000 instruções aritméticas, dividido em dois blocos de 2.500 instruções. Assim, ocorre a diminuição da frequência de $800MHz$ para $400MHz$ após o primeiro bloco de 2.500 instruções. A figura mostra o aumento do tempo médio de execução do bloco de instruções para o RVEC, como esperado, com a linha pontilhada representando o tempo aferido quando da utilização direta do CCNT.

4.2. Custo de acesso ao relógio de sistema

O custo de acesso ao RVEC como relógio de sistema foi realizado através do CCNT, que oferece a melhor precisão disponível para um relógio do sistema. Utili-

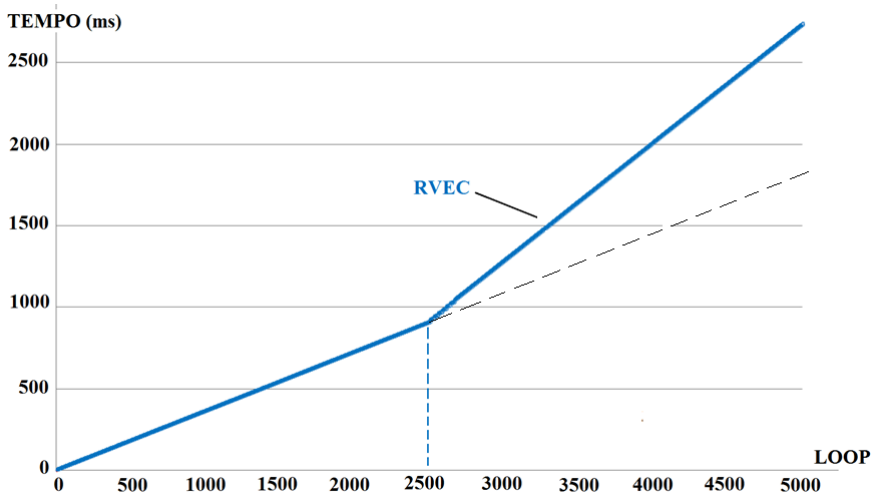

Figura 10. Propriedade SIP com alteração da frequência de 800 MHz para 400 MHz

zando este artifício, foi possível aferir também o custo de acesso do próprio CCNT, assim como dos relógios de sistema MONOTONIC e REALTIME. O *microbenchmark* é composto de um laço de 2.400 instruções aritméticas, onde após as primeiras 1.200 instruções aritméticas do laço, o relógio de sistema avaliado é acessado. O laço foi executado 1.000 vezes e, em cada iteração, foram avaliados CCNT, RVEC, MONOTONIC e REALTIME usando a chamada de sistema clock_gettime() para estes três últimos relógios.

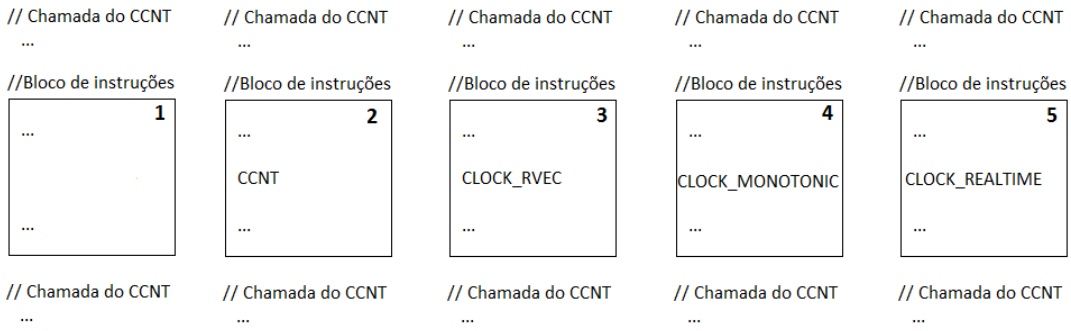

Figura 11. Configurações para cálculo da sobrecarga para cada relógio

A Figura 11 apresenta as cinco configurações utilizadas na execução dos experimentos como o *microbenchmark* descrito anteriormente. A configuração 1 não realiza consulta aos relógios de sistema, servindo assim como referência para comparação do custo de acesso. Os cenários 2 e 3 avaliam o custo de acesso ao CCNT e ao RVEC respectivamente, enquanto que os cenários 4 e 5 avaliam o custo de acesso aos relógios de sistema do Linux MONOTONIC e REALTIME, respectivamente. Note que cada configuração foi executada 100 vezes.

A Tabela 2 apresenta os resultados obtidos para a execução do experimento com o Raspberry Pi operando a frequência de $800MHz$, sendo que o tempo médio aferido para a configuração 1, configuração de referência, foi de $170,354$ μs com um desvio padrão de $6,507$ μs. A configuração 2 teve uma sobrecarga média de execução de $1,333$ μs com

Tabela 2. Tempo de Execução vs. Opção de Relógio (800 MHz)

Opções de Relógio	Média (μs)	Desvio Padrão (μs)	Sobrecarga (μs)	Sobrecarga (%)
CCNT	$171,687$	$6,729$	$1,333$	$0,78$
RVEC	$174,502$	$7,419$	$4,148$	$2,43$
MONOTONIC	$173,107$	$6,918$	$2,753$	$1,62$
REALTIME	$173,013$	$6,891$	$2,659$	$1,56$

relação a configuração 1, o que representa uma sobrecarga de $0,78\%$. A execução do experimento com o RVEC, configuração 3, obteve uma sobrecarga média de $4,148$ μs o que representa uma sobrecarga de $2,43\%$. As configurações 4 e 5 apresentaram uma sobrecarga média de $2,753$ μs $(1,62\%)$ e $2,659$ μs $(1,56\%)$ respectivamente, onde a pequena diferença entre elas é explicada pelas validações adicionais realizadas pelo relógio MONOTONIC.

Tabela 3. Tempo de Execução vs. Opção de Relógio (400 MHz)

Opções de Relógio	Média (μs)	Desvio Padrão (μs)	Sobrecarga (μs)	Sobrecarga (%)
CCNT	$344,381$	$19,831$	$2,702$	$0,79$
RVEC	$349,680$	$20,448$	$8,002$	$2,34$
MONOTONIC	$346,867$	$19,187$	$5,188$	$1,52$
REALTIME	$346,804$	$19,720$	$5,125$	$1,50$

A Tabela 3 apresenta os resultados obtidos para a execução do experimento descrito no paragrafo anterior, agora com o Raspberry Pi operando a frequência de $400MHz$, sendo que o tempo médio aferido para a configuração 1, configuração de referência, foi de $341,679$ μs. A configuração 2 teve uma sobrecarga média de execução de $2,702$ μs em relação à configuração 1, o que representa uma sobrecarga de $0,79\%$. A execução do experimento com o RVEC, configuração 3, obteve uma sobrecarga média de $8,002$ μs o que representa uma sobrecarga de $2,34\%$. As configurações 4 e 5 apresentaram uma sobrecarga média de $5,188$ μs $(1,52\%)$ e $5,125$ μs $(1,50\%)$, respectivamente.

4.3. Precisão

A avaliação da precisão dos relógios de sistema é realizada utilizando como base a configuração 1 do *microbenchmark* da Seção 4.2. Contudo, neste experimento a duração do bloco de 2.400 instruções aritméticas é aferida tanto com o CCNT como com os diferentes relógios de sistemas e o RVEC. O experimento foi repetido 100 vezes e como anteriormente foram realizadas 1.000 iterações do laço.

Tabela 4. Precisão de Tempo vs. Opção de Relógio (800 MHz)

Opções de Relógio	Média (μs)	Desvio Padrão (μs)
CCNT	$170,439$	$6,554$
RVEC	$172,820$	$7,248$
MONOTONIC	$172,047$	$7,086$
REALTIME	$171,459$	$7,140$

A Tabela 4 apresenta os resultados da precisão dos diferentes relógios avaliados com a plataforma Raspberry Pi operando a frequência de $800MHz$. A duração média do bloco aferida através do CCNT foi de $170,439\ \mu s$, com um desvio padrão de $6,554\ \mu s$, o que presenta um coeficiente de variação de $0,038$. Por sua vez a execução deste experimento aferida com o RVEC levou em média $172,820\ \mu s$, $7,248\ \mu s$ de desvio padrão e um coeficiente de variação de $0,042$. Ainda nesta tabela, as últimas duas linhas apresentam os resultados da aferição do bloco de instrução com os relógios de sistemas MONOTONIC e REALTIME, que mediram respectivamente $172,047\ \mu s$ e $171,459\ \mu s$, e representam coeficientes de variação de $0,041$ e $0,042$, respectivamente.

4.4. Discussão dos resultados

Os experimentos apresentados ao longo desta Seção fornecem indícios preliminares da estabilidade do contador de ciclos presente nos processadores ARM1176JZF-S (CCNT) e da corrente implementação do RVEC realizada para a plataforma Raspberry Pi que utiliza esse processador. Em especial, os experimentos apresentados na Seção 4.3 mostram que tanto o CCNT como o RVEC obtiveram coeficientes de variação inferiores a $0,05$, $0,042$ (RVEC) e $0,038$ (CCNT). Ainda, os resultados apresentados na Seção 4.1, mostram que a única solução de relógio que consegue garantir à aderência a propriedade ECP independente do intervalo de tempo sendo aferido é o RVEC.

5. Conclusão

Este trabalho apresentou a implementação e avaliação do método RVEC para a plataforma computacional Raspberry Pi, como alternativa aos relógios de sistemas atualmente presentes nesta plataforma. No Raspberry Pi, o RVEC foi implementado sobre o contador CCNT presente nos processadores da família ARM11 como o ARM1176JZF-S. Diferente dos relógios de sistema existentes, nesta plataforma computacional o RVEC é aderente a propriedade ECP independente da duração do intervalo de tempo sendo aferido. Os resultados apresentados na Seção 4 mostram que a corrente implementação do RVEC é aderente à propriedade ECP, contudo impõem uma sobrecarga levemente maior que a dos relógios de sistema existentes na plataforma computacional. A diferença entre o custo de acesso com o Raspberry Pi operando a $800MHz$ foi de $1,395\ \mu s$ para o MONOTONIC e $1,489\ \mu s$ para o REALTIME. Ainda, os resultados dos experimentos realizados com o RVEC submetido à mudança de frequência, mostraram que a propriedade ECP foi preservada, permitindo que a plataforma computacional possa utilizar o mecanismo de DVFS para aumentar sua eficiência energética em contraposição ao impedimento imposto pelos atuais relógios de sistema.

Os resultados preliminares apresentados confirmam a aderência do RVEC a propriedade ECP e de que este e um relógio de sistema com resolução de nanosegundos enquanto mantem um custo de acesso equivalente ao dos demais relógios de sistemas existentes. Entre os trabalhos futuros estão uma avaliação desta implementação do RVEC utilizando um GPS como *"ground clock"* visando aferir a estabilidade do RVEC para intervalos de dias/meses, outra frente de trabalho quantificar os ganhos na eficiência energética que o uso de um relógio de sistema sem interrupções oferece a plataforma.

Referências

[ARM] ARM. *ARM1176JZ-S Technical Reference Manual*.

[Broadcom] Broadcom. *BCM2835 ARM Peripherals*.

[Corrêa 2014] Corrêa, E. C. (2014). Relógio Virtual Estritamente Crescente para o Computador Raspberry Pi. Master's thesis, COPPE/UFRJ.

[de Souza et al. 2008] de Souza, A. F., de Souza S. F., de Amorim, C. L., Lima, P., and Rounce, P. (2008). Hardware supported synchronization primitives for clusters. In *PDPTA'08*, pages 520–526.

[Dutra et al. 2013] Dutra, D., Whately, L., and Amorim, C. (2013). Attaining strictly increasing and precise time count in energy-efficient computer systems. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2013 25th International Symposium on*, pages 65–72.

[Fan et al. 2004] Fan, R., Chakraborty, I., and Lynch, N. (2004). Clock synchronization for wireless networks. In *In Proc. 8th International Conference on Principles of Distributed Systems (OPODIS)*, pages 400–414.

[Khan et al. 2012] Khan, J., Bilavarn, S., and Belleudy, C. (2012). Energy analysis of a dvfs based power strategy on arm platforms. In *Faible Tension Faible Consommation (FTFC), 2012 IEEE*, pages 1–4.

[Mills 1992] Mills, D. (1992). Network time protocol (version 3) specification, implementation.

[Nagy and Gingl 2013] Nagy, T. and Gingl, Z. (2013). Low-cost photoplethysmograph solutions using the raspberry pi. In *Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium on*, pages 163–167.

[Neves and Matos 2013] Neves, R. and Matos, A. (2013). Raspberry pi based stereo vision for small size asvs. In *Oceans - San Diego, 2013*, pages 1–6.

[Simunic et al. 2001] Simunic, T., Benini, L., and De Micheli, G. (2001). Energy-efficient design of battery-powered embedded systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 9(1):15–28.

[Tian et al. 2008] Tian, G.-S., Tian, Y.-C., and Fidge, C. (2008). High-precision relative clock synchronization using time stamp counters. In *Engineering of Complex Computer Systems, 2008. ICECCS 2008. 13th IEEE International Conference on*, pages 69–78.

[Veitch et al. 2009] Veitch, D., Ridoux, J., and Korada, S. (2009). Robust synchronization of absolute and difference clocks over networks. *Networking, IEEE/ACM Transactions on*, 17(2):417–430.

[Vujovic and Maksimovic 2014] Vujovic, V. and Maksimovic, M. (2014). Raspberry pi as a wireless sensor node: Performances and constraints. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, pages 1013–1018.