OTIMIZAÇÃO DE SOFTWARE MULTIPLATAFORMA E CRÍTICO EM DESEMPENHO PARA MONITORAÇÃO DE GERADORES

Bruno de Borba[1], Fabiane Barreto Vavassori Benitti[2], Mauro Pacheco Ferreira[3], Fabrizio Leal Freitas[4], Tiago Kaoru Matsuo[5]

Resumo

Este artigo descreve as soluções utilizadas na adequação de um software multiplataforma de monitoração de geradores para o atendimento a requisitos de desempenho demandados por novas plataformas de hardware enxutas. O software em questão foi originalmente desenvolvido para ambientes Linux e Windows em arquitetura x86, e posteriormente foi adaptado para plataformas ARM e Altera/NIOS II em ambiente Linux, plataformas mais limitadas em capacidade de processamento. As soluções utilizadas envolvem o uso de diretivas de compilação condicional, priorização de processos e tarefas, operações atômicas, seções críticas, semáforos, eventos, mutex e spinlock. Os resultados do artigo mostram que os requisitos foram atendidos após as adequações do software embarcado.

Abstract

This paper describes the solutions used in the suitability of a multiplatform generators monitoring software to meet the performance requirements demanded by new lean hardware platforms. The software in question was originally developed for Linux and Windows environments on x86 architecture, and was subsequently adapted for ARM and Altera/NIOS II platforms in Linux environments, platforms in more limited processing capacity environment. The used solutions involve the use of conditional compilation directives, prioritizing tasks and processes, atomic operations, critical sections, semaphores, events, mutex and spinlock. The paper results show that the requirements have been met after de embedded software adaptations.

1. Contextualização

O projeto de P&D ENDESA – ANEEL executado pela AQTech Engenharia e Instrumentação S.A., Projeto de pesquisa e desenvolvimento em "Nacionalização de produto para monitoramento de grupos geradores" consiste no desenvolvimento de uma solução nacional para o monitoramento de grupos motor-gerador. Consiste na adaptação de conceitos e filosofias de monitoração de usinas hidrelétricas aplicados nos últimos 10 anos pela AQTech para o contexto enxuto de monitoramento de grupos motor-gerador, que são máquinas de menor porte, muitas vezes operando em geração distribuída.

[1] Especialista em Qualidade e Engenharia de Software, Engenheiro Eletricista – AQTech Engenharia e Instrumentação S.A.
[2] Doutora em Engenharia de Produção, Mestre em Ciências da Computação, Graduada em Ciências da Computação – Universidade Federal de Santa Catarina.
[3] Mestre em Engenharia de Produção, Engenheiro Eletricista – AQTech Engenharia e Instrumentação S.A.
[4] Mestre em Engenharia de Produção, Engenheiro Eletricista – AQTech Engenharia e Instrumentação S.A.
[5] Engenheiro Eletricista – AQTech Engenharia e Instrumentação S.A.

Um dos objetivos do projeto é tornar as tecnologias de monitoramento já existentes mais baratas para que possam atingir o mercado de grupos geradores. Com a redução de custos, ocorre naturalmente a redução da capacidade de processamento do equipamento de monitoração, entre outras limitações.

O objetivo deste artigo é apresentar as soluções utilizadas na adequação do software embarcado Pulsar para integração ao hardware desenvolvido no projeto, no contexto do desenvolvimento de aplicações multiplataforma críticas em desempenho.

2. Introdução

O principal componente do hardware desenvolvido no projeto é o FPGA (Field Programmable Grid Array) modelo Cyclone V da Altera. Este circuito integrado controla todos os demais componentes do hardware, como o RTC, RS-232, RS-485, LAN, Entradas Analógicas (Conversor A/D), Entradas Digitais, Saídas Digitais, LEDs, Sync, Watchdog, cartão micro SD e memória DDR3.

Dentro do FPGA é embarcado um firmware em RTL (Real Time Logic) que controla os componentes supracitados. Além disso, o firmware também implementa um processador em lógica programável chamado NIOS II (ALTERA). Este processador, integrado com a memória DDR3 e a memória Flash do hardware, compõe a plataforma básica de processamento do hardware.

Esta plataforma de hardware suporta o sistema operacional uClinux (microC Linux) em que o software embarcado Pulsar é executado.

O software embarcado Pulsar é um kernell embarcado de aquisição de dados e cálculo de diagrama de blocos. Trata-se de um software multiplataforma que é crítico em desempenho devido à grande quantidade de operações que deve realizar em um curto período de tempo, mesmo em processadores mais limitados.

Conforme Sommerville (2007) define, "Sistemas críticos de segurança são sistemas cuja falha pode resultar em prejuízo, perda de vida ou danos sérios ao ambiente". Como o software Pulsar é responsável pelo monitoramento de máquinas de alto valor agregado, normalmente em ambientes de risco, o software embarcado Pulsar pode ser classificado como um sistema crítico de segurança e deve possuir desempenho adequado para conseguir executar sua função na plataforma de hardware em que estiver sendo executado.

O termo multiplataforma refere-se a um software que pode ser executado em mais de uma plataforma de hardware e software. No contexto da informática, plataforma refere-se a dispositivos computacionais utilizados para iteração com o sistema, seja hardware ou software. Esse conceito normalmente é aplicado à portabilidade entre sistemas operacionais, porém o conceito também contempla a execução do software em diferentes plataformas de hardware, como em processadores x86, ARM e Altera / NIOS II (BLOOM, 2013).

O artigo inicialmente apresenta o software Pulsar, suas funções e sua arquitetura. Em seguida são apresentados os requisitos definidos previamente com a equipe do projeto. Posteriormente são apresentadas as soluções utilizadas nas adequações. Em seguida, são apresentados os resultados obtidos através do atendimento aos requisitos e da comparação de processamento em diferentes arquiteturas e ambientes. Por fim, são apresentadas as conclusões.

3. Software Pulsar

O software Pulsar foi desenvolvido em C++ e sua função primária é adquirir amostras de sinais digitalizados e processá-las matematicamente. Os sinais podem ser adquiridos através de protocolos industriais ou através de entradas analógicas e digitais presentes no próprio hardware desenvolvido no projeto. Os sinais analógicos são digitalizados através de conversores analógico-digitais (FREIRE, 2014) e repassados para o software Pulsar, que processa as informações adequadamente.

Para a digitalização os sinais analógicos são multiplexados (SOUZA, 2014) para que seja possível reduzir custos de hardware. O processamento matemático é realizado através de modelos de cálculo, representáveis em diagramas de blocos, que executam diversas funções para tratar adequadamente cada sinal e obter informações relevantes do monitoramento.

O software Pulsar deve ainda armazenar em disco informações pertinentes ao monitoramento dos sinais, para que essas informações possam ser transmitidas posteriormente a um datacenter e analisadas adequadamente por um especialista. O software dispõe ainda de diversos serviços implementados via TCP/IP em protocolo proprietário para visualização e controle do processo de monitoramento.

Dependendo da arquitetura de hardware em que o software está sendo executado, é possível ainda acionar alarmes ou saídas digitais e com isso interromper o funcionamento de equipamentos de alto valor agregado em caso de falha.

O software foi desenvolvido visando os padrões ANSI C (American National Standards Institute for the C programming language) e POSIX (Portable Operating System Interface), especificado pela ISO/IEC 9945 (ISO/IEC/IEEE 9945:2009), o que torna o software compatível com a maioria dos compiladores.

O Software Pulsar possui classes de gerenciadores, que são responsáveis por gerenciar instâncias de objetos que são especializações de uma determinada classe-base. Por exemplo, existe o gerenciador de dispositivos de comunicação, que é responsável por gerenciar as instâncias de objetos que controlam o envio de dados ao datacenter. Cada especialização da classe-base representa um dispositivo que pode ser controlado pelo software Pulsar e enviar dados ao datacenter, como um modem GPRS ou um modem para comunicação via satélite.

O diagrama de classes ilustrado na Figura 1 mostra parte do software Pulsar. A documentação não é apresentada de forma integral devido a confidencialidade do software, que é comercial.

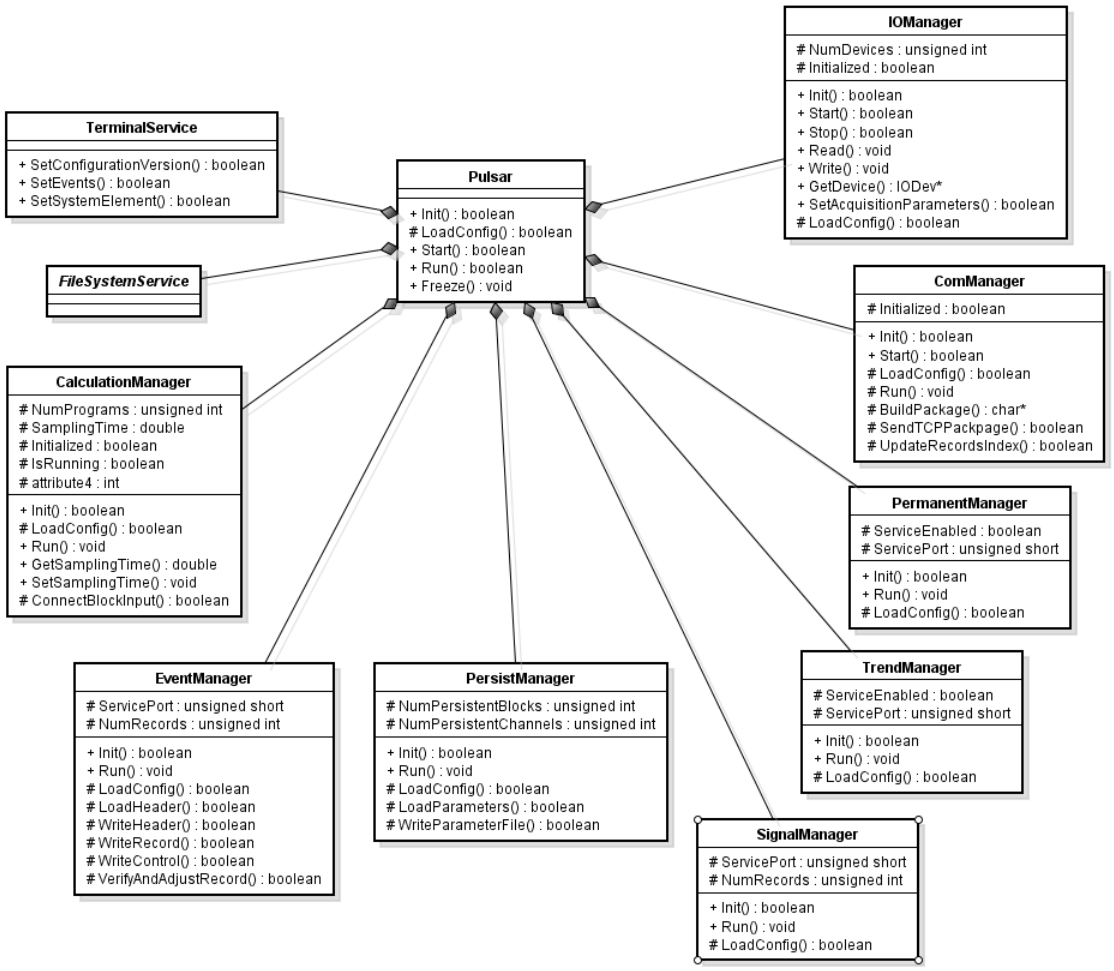

Figura 1. Janela do programa

Quando este software, que é responsável por adquirir as amostras e processá-las matematicamente, não dispõe de tempo suficiente para realizar suas tarefas, um buffer circular acumula informações até que amostras comecem a ser descartadas. Portanto, quanto mais eficiente for o software em termos de desempenho, maior será a quantidade de amostras que conseguirá processar em um espaço de tempo limitado. No monitoramento de algumas variáveis, é necessário o processamento matemático de milhares de amostras por segundo, o que torna crítico o desempenho do software.

Como o equipamento produto do projeto de P&D tem como um de seus principais requisitos o baixo custo, sua capacidade de processamento é muito limitada. A necessidade de baixo custo do equipamento provém do objetivo do projeto em atingir o mercado de grupos geradores, que são equipamentos de custo mais baixo quando comparados a grandes geradores de usinas hidrelétricas de energia.

4. Requisitos Elicitados

Esta seção apresenta os requisitos de desempenho e multiplataforma definidos previamente com a equipe do projeto. Os requisitos são do tipo não-funcionais (SOMMERVILLE, 2007). A Tabela 1 mostra a relação de requisitos.

Tabela 1. Requisitos elicitados

N°	Requisito	Descrição
RNF1	Multiplataforma	As funcionalidades implementadas devem ser compatíveis com Windows XP SP3, Linux Gentoo 11.0, uClinux 2.6.0 e Raspibian 2012-12-16-wheezy
RNF2	Desempenho compatível com plataforma desenvolvida	O sistema deve ser capaz de executar com folga de processamento (utilizar menos de 100% da CPU) na plataforma desenvolvida no projeto (FPGA Altera Cyclone IV EP4CE75) quando executa o modelo de cálculo padrão previamente estabelecido
RNF3	Multitarefa	As funcionalidades devem ser implementadas buscando o processamento paralelo sempre que possível para otimização de desempenho em sistemas com vários processadores/núcleos. Para verificação, o sistema deve conter no mínimo 8 tarefas
RNF4	Prioridade alta para tarefas consideradas mais importantes (aquisição e processamento matemático)	As tarefas de aquisição de dados e processamento matemático devem ser priorizadas com o maior nível de prioridade que pode ser definido nos sistemas operacionais compatíveis

Esses requisitos devem ser atendidos na integração do software à nova plataforma de hardware desenvolvida para tornar possível o monitoramento a taxas satisfatórias. Estão omitidos diversos requisitos do projeto não relacionados ao contexto deste artigo.

5. Soluções Utilizadas

Esta seção apresenta as soluções utilizadas para que o software multiplataforma Pulsar obtivesse bom desempenho nos diferentes ambientes em que é executado, atendendo aos requisitos elicitados. As soluções são baseadas em diretivas de compilação condicional, implementadas em bibliotecas multiplataformas com operações atômicas, como contador atômico, e com o uso de seção crítica, semáforos e eventos. Ainda, por se tratar de um software multitarefa, foram definidas prioridades no escalonamento de suas tarefas.

Nas seções a seguir as soluções utilizadas são exploradas e justificadas quanto à adequação da solução ao contexto do problema.

5.1. Diretivas de Compilação Condicional

As diretivas de compilação condicional são utilizadas para definir ações que o compilador deverá tomar durante a compilação em função de condições testadas imediatamente antes da compilação do software (RICARTE, 2003) através do pré-processador C.

Por exemplo, a diretiva "#ifdef WIN32" foi utilizada em diversas partes do código fonte para identificar quando a macro WIN32 está definida ou não. A macro WIN32 não é definida no código fonte através da diretiva "#define", mas sim através dos próprios compiladores utilizados para compilar o software Pulsar para o sistema operacional Windows.

Esta macro é largamente verificada no código fonte do software Pulsar. Todas as bibliotecas implementadas para criar e gerenciar as diversas tarefas do processo utilizam diretivas de compilação condicional. No software Pulsar, quando a macro "WIN32" está definida, significa que o software está sendo compilado para o sistema operacional Microsoft Windows. Quando a macro não está definida, o software está sendo compilado para Linux.

O efeito da diretiva de compilação condicional é uma alteração no código fonte imediatamente antes da compilação do código. Note que funções que não satisfazem diretivas de compilação condicional podem nem mesmo ser implementadas. Nesse caso, os trechos de código são equivalentes a comentários, pois não são compilados.

O resultado é a abstração de uma função que pode ser utilizada largamente no código fonte sem a preocupação do sistema operacional em que está sendo executada, formando parte de uma biblioteca multiplataforma.

Se as diretivas de compilação condicional não fossem utilizadas, não seria possível utilizar funções que existem somente para um determinado sistema operacional. Perder-se-ia, portanto, a possibilidade de se utilizar as funções atômicas mostradas acima como exemplo, o que geraria perda de desempenho e aumento de complexidade. O uso de funções atômicas é descrito na seção 5.3.

O uso de diretivas de compilação condicional está diretamente relacionado ao requisito não-funcional RNF1, pois permite o uso de funções especificas de cada plataforma.

5.2. Escalonamento de Processos e Tarefas

Para entender o escalonamento de processos e tarefas, é necessário primeiramente definir processo e tarefa.

Tarefas são algoritmos executados de forma serial (procedural), que podem ser sincronizadas com outras tarefas do mesmo processo ou até mesmo com tarefas de outros processos com o uso de funções e variáveis especiais.

Já os processos são agrupamentos de tarefas, iniciados a partir da execução de um arquivo executável ou binário. Um processo é também chamado de "aplicação" (SILBERSCHATZ, 2014).

Ainda segundo Silberschatz (2014), o escalonamento de tarefas é uma atividade realizada pelo escalonador da CPU e possibilita a execução de processos e tarefas de forma concorrente. O efeito macro do escalonamento é a realização de tarefas em paralelo, porém todo o processamento é realizado de forma serial, com o chaveamento constante entre os diferentes processos e tarefas que estão sendo executados pelo sistema operacional. Esse escalonamento ocorre em curtos espaços de tempo, variando de acordo com o hardware e com o software.

A importância desses conceitos em relação ao problema está na priorização de tarefas e processos para o escalonamento. Para funções de um programa que devem ser executadas prioritariamente, é possível atribuir altas prioridades a processos e tarefas. Com isso, o escalonador da CPU vai priorizar determinados processos e tarefas em prejuízo de outros considerados de menor importância.

No programa Pulsar, as tarefas que controlam os dispositivos de aquisição de dados e o processamento matemático das amostras são consideradas críticas, pois sem elas o software não executa sua principal função, que é adquirir amostras e processá-las matematicamente. Outras tarefas, como o armazenamento de registros de sinais da memória RAM para uma memória não-volátil, são realizadas com mais baixa prioridade, pois a informação presente na memória RAM não se perde caso o programa demore um pouco mais para armazenar a informação de forma permanente. Com isso o programa tem tempo de executar suas tarefas mais críticas em tempo hábil e posteriormente realiza funções consideradas secundárias.

A atribuição de prioridades a tarefas e processos para modificar a forma como o sistema operacional realizada o escalonamento de processos e tarefas está relacionada ao requisito não-funcional RNF4.

5.3. Operações Atômicas

Operações atômicas são operações ou conjuntos de operações que devem ser executadas instantaneamente, ou seja, devem alterar o estado do sistema completamente em caso de sucesso ou não apresentar efeito algum em caso de falha. Em operações atômicas, não é possível executar parte do código ou estar em um estado intermediário entre o começo e o fim da execução da operação (LOVE, 2010).

Operações atômicas em si já são uma garantia de isolamento entre processos ou tarefas concorrentes. Por exemplo, não é necessário proteger uma variável através de uma seção crítica se a variável tiver seu estado alterado através de uma operação atômica. O conceito de seção crítica é descrito na seção 5.4.

O uso de operações atômicas está diretamente relacionado ao requisito não-funcional RNF2, pois funções atômicas costumam diminuir a complexidade do código fonte, tornando desnecessário o uso de seções críticas em alguns casos, além de diminuir o número de instruções que deverão ser realizadas pelo processador para a execução de uma determinada função, contribuindo dessa forma para a diminuição da utilização da CPU.

5.4. Seção Crítica

O objetivo de uma seção crítica (ou região crítica) é tornar um conjunto de operações atômico para outra tarefa, ou seja, não permitir que hajam estados intermediários na execução de um determinado trecho de código (MAZIERO, 2013).

A utilização de seções críticas é essencial na utilização de recursos compartilhados, como áreas de memória ou até mesmo dispositivos de hardware. Uma seção crítica assegura o uso exclusivo de um recurso.

A implementação de seções críticas varia de acordo com os sistemas operacionais. A utilização de diretivas de compilação é uma boa estratégia para abstrair as funções utilizadas na implementação de seções críticas.

O uso de seções críticas está diretamente relacionado aos requisitos RNF2 e RNF3, pois permite que tarefas sejam realizadas em paralelo de forma segura, o que otimiza o uso da CPU, principalmente em sistemas com vários processadores ou núcleos.

5.5. Contador Atômico

O contador atômico é uma classe implementa no software Pulsar que permite a alteração do estado de contadores (variáveis do tipo int) de forma atômica (instantânea). Desta forma, cada operação em um contador atômico, que normalmente é um recurso compartilhado, é realizada sem necessidade de entrar ou sair em regiões críticas (seções críticas), o que melhora o desempenho do software.

A implementação varia de acordo com os sistemas operacionais utilizados, sendo necessária a utilização de diretivas de compilação condicional na abstração das funções.

A implementação de uma classe que utiliza exclusivamente funções atômicas especificas de cada plataforma para alterar o estado de contadores está relacionada ao requisito não-funcional RNF2.

5.6. Semáforos e Eventos

Semáforos são variáveis especiais protegidas que podem controlar o acesso a recursos compartilhados em um sistema multitarefa. Os semáforos, após inicializados, disponibilizam pelo menos duas operações atômicas: wait e signal (LOVE, 2010).

A função signal sinaliza (incrementa) um semáforo, indicando a ocorrência de algum evento, enquanto que a função wait verifica se o semáforo está sinalizado ou não.

Normalmente a função wait decrementa o semáforo. A diferença entre Semáforos e Eventos é que Eventos são sinalizados somente uma vez, enquanto que semáforos podem ser sinalizados diversas vezes. Eventos também são chamados de "Semáforos Binários" (SEIXAS FILHO, 2005).

Um exemplo de utilização de Semáforos é o processo de aquisição de dados do Pulsar. Quando uma nova amostra é digitalizada, um semáforo é incrementado (ou sinalizado), indicando que uma nova amostra foi disponibilizada para o processamento matemático, que é realizado em uma tarefa concorrente.

No exemplo dado, é possível que diversas amostras sejam disponibilizadas para o processamento matemático sem que o mesmo seja executado imediatamente, ou seja, é possível que haja um acúmulo de amostras no buffer do dispositivo (várias amostras disponíveis para processamento matemático). Neste caso o semáforo estará sinalizado diversas vezes.

O tamanho do buffer do dispositivo é limitado para evitar que haja um acúmulo constante de amostras disponíveis para processamento. Isso indicaria que o programa está adquirindo mais amostras do que consegue processar. Neste caso o descarte de amostras é inevitável.

O uso de semáforos e eventos está diretamente relacionado ao requisito não-funcional RNF3.

5.7. Mutex e Spinlock

O acrônimo Mutex (para Mutual Exclusion, do Inglês) é utilizado para prevenir acesso simultâneo a um mesmo recurso em programação paralela. Um Mutex pode ser utilizado, por exemplo, para implementar um Semáforo (TAUBENFELD, 2004).

O uso de Mutex pode gerar problemas como deadlock e inanição. Deadlock ocorre quando uma tarefa fica bloqueada eternamente, aguardando a liberação de um ou mais recursos. Inanição ocorre quando uma tarefa nunca dispõe de recursos suficientes para ser executada, ou seja, outras tarefas mais prioritárias consomem constantemente os recursos necessários para a execução da tarefa.

Por exemplo, uma situação que gera um deadlock é quando uma tarefa aguarda a liberação de um recurso que está em uso por outra tarefa, que por sua vez aguarda a liberação de um recurso que está em uso pela primeira tarefa. Isso gera um impasse onde as duas tarefas permanecem bloqueadas eternamente, caracterizando um deadlock.

Spinlock funciona da mesma forma que um Mutex, porém a tarefa que aguarda a liberação de um recurso verifica constantemente a liberação do recurso, repetitivamente, de forma intensiva (LOVE, 2010).

Devido à grande utilização de recursos de processamento durante a checagem pela liberação de um recurso protegido do sistema, o Spinlock é mais eficiente em tarefas que bloqueiam um recurso por um período de tempo muito curto.

Se um recurso deverá ficar bloqueado por um período de tempo mais longo, a utilização de Mutex se torna mais adequada.

Essas variáveis especiais são largamente utilizadas no software Pulsar para o compartilhamento de recursos compartilhados entre tarefas, como áreas de memória e até mesmo dispositivos.

O uso de mutex e spinlock está diretamente relacionado aos requisitos RNF2 e RNF3, pois permite o acesso a recursos compartilhados de forma eficiente.

6. Resultados

Este capítulo evidencia o atendimento aos requisitos elicitados no Capítulo 3 e apresenta uma comparação da porcentagem de utilização de processadores em ambientes Linux (em ARM e em FPGA) e Windows (em Intel Core i5).

O requisito não-funcional RNF1 (multiplataforma) foi atendido, já que todas as funcionalidades foram implementadas e testadas em todas as plataformas citadas. O uso de diretivas de compilação condicional foi essencial na construção de bibliotecas multiplataforma que utilizam funções especificas de cada plataforma em seus métodos, como operações atômicas.

O requisito não-funcional RNF2 (desempenho compatível com plataforma desenvolvida) foi atendido. A utilização da CPU ficou em aproximadamente 9% na plataforma desenvolvida no projeto executando o modelo de cálculo padrão previamente estabelecido. A medição foi realizada com o uso do programa “top” do uClinux no chip EP4CE75.

Para evidenciar a limitação da capacidade de processamento da plataforma desenvolvida no projeto, foi criado posteriormente um modelo de cálculo simulado que é executado no software embarcado Pulsar a uma mesma taxa nos três ambientes testados. Os canais de hardware são simulados através de um dispositivo interno ao software embarcado, tornando possível a análise de um mesmo modelo de cálculo em diferentes plataformas. O modelo utilizado foi executado a uma taxa de 625 Hz.

A primeira medição foi realizada na plataforma desenvolvida no projeto, sobre um chip FPGA Altera Cyclone IV EP4CE75, com NIOS II.

O resultado da medição foi obtido através do programa “top” da distribuição uClinux 2.6.1. Verificou-se que a porcentagem de utilização do processador foi de 56% em média.

A segunda medição foi realizada no hardware Raspberry Pi modelo B, que possui um chip ARM1176JZFS.

O resultado da medição foi obtido através do programa “top” da distribuição Linux utilizada (Raspibian 2012-12-16-wheezy). Verificou-se que a porcentagem de utilização do processador foi de 1.3% em média.

A terceira e última medição foi realizada em um notebook comercial com Windows 8.1 e processador Intel Core i5-560M 2.66GHz.

O resultado da medição foi obtido através do programa “Gerenciador de Tarefas” do Windows. Pode-se verificar na Figura 2 que a porcentagem de utilização do processador foi inferior a 1%.

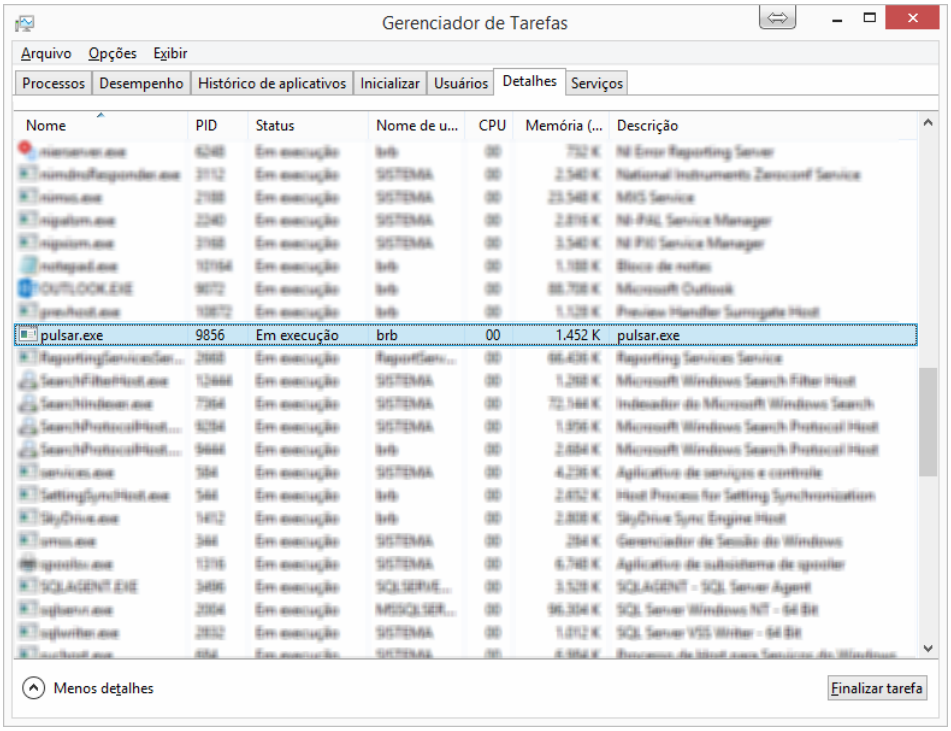

Figura 2. Utilização do processamento no chip Intel Core i5-560M 2.66GHz

A Figura 3 mostra uma comparação em gráfico de barras da utilização de processamento em cada chip analisado com o modelo simulado. A primeira coluna refere-se ao Intel Core i5-560M 2.66GHz; a segunda coluna refere-se ao ARM1176JZFS; a terceira coluna refere-se ao FPGA EP4CE75.

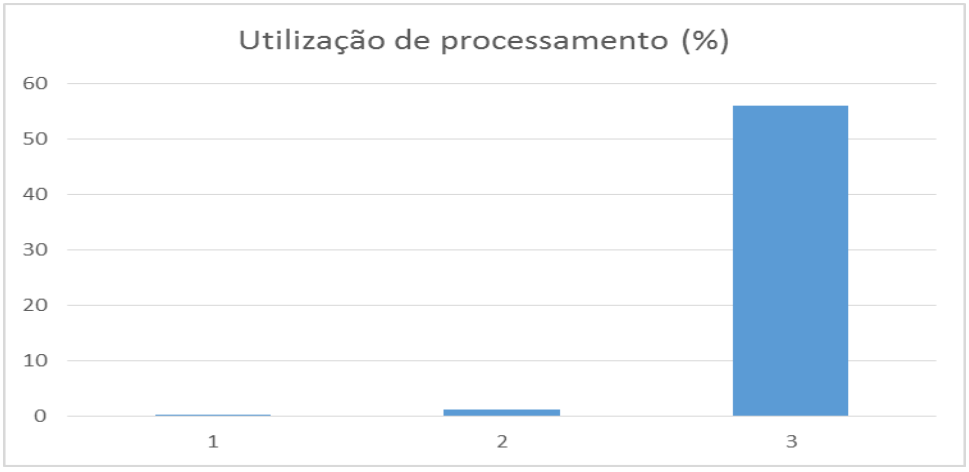

Figura 3. Comparação da utilização de processamento com modelo simulado

O requisito não-funcional RNF3 (multitarefa) foi atendido com o uso de seções críticas, semáforos, eventos, mutex e spinlock. O sistema contém no total 24 tarefas, porém nem todas são executadas simultaneamente.

O requisito não-funcional RNF4 (prioridade alta para tarefas consideradas mais importantes) foi atendido com o uso de funções de priorização de tarefas e processos no escalonador da CPU. Foi necessário o uso de diretivas de compilação condicional, pois as funções utilizadas são diferentes para cada plataforma.

7. Conclusões

As implementações realizadas no software embarcado Pulsar foram testadas e validadas quanto ao seu correto funcionamento e quanto a utilização de recursos de hardware consumidos, principalmente quanto ao uso do processamento, principal limitação no produto SMGer-GMG desenvolvido no projeto.

Todos os requisitos elicitados no Capítulo 3 foram atendidos. Através do uso de diretivas de compilação condicional foi possível criar bibliotecas multiplataforma, que agilizaram o desenvolvimento e permitiram o uso de funções específicas para cada ambiente, como algumas operações atômicas.

Em testes realizados foi possível utilizar até 100% da CPU sem que houvesse comprometimento da função principal do software, que é adquirir e processar matematicamente amostras digitalizadas. Isso foi possível através da priorização de processos e tarefas no escalonador da CPU.

Com a utilização de proteções para acesso a recursos compartilhados foi possível também evoluir o software quanto a utilização de vários núcleos de um processador, pois foram criadas tarefas para a execução de funções em paralelo.

Como resultado final, o software embarcado Pulsar foi executado na plataforma de hardware desenvolvida no projeto de P&D ENDESA – ANEEL executado pela AQX Instrumentação Eletrônica S.A. e apresentou desempenho satisfatório, deixando folga na utilização do processamento no conjunto protótipo, o que permite a utilização de modelos de cálculo mais complexos e custosos em termos de processamento. Modelos mais complexos são necessários em diversas aplicações práticas de monitoramento de geradores.

7. Agradecimentos

Os autores deste artigo destacam e agradecem as relevantes participações no contexto do projeto da equipe da ENDESA, representada pelo coordenador do projeto Marcelo Golin Buzzatti.

8. Referências

ALTERA. Cyclone V Device Datasheet, Disponível em: <http://www.altera.com/literature/hb/cyclone-v/cv_51002.pdf>. Acessado em 29 abr. 2014.

ALTERA. Cyclone V Device Handbook. Volume 1. 2014. Disponível em <http://www.altera.com/literature/hb/cyclone-v/cyclone5_handbook.pdf>. Acessado em 29 abr. 2014.

ALTERA. Cyclone V Device Overview, Disponível em: <http://www.altera.com/literature/hb/cyclone-v/cv_51001.pdf>. Acessado em 29 abr. 2014.

ALTERA. Nios II Processor: The World's Most Versatile Embedded Processor. 2013. Disponível em <http://www.altera.com/devices/processor/nios2/ni2-index.html>. Acessado em 29 abr. 2014.

BLOOM, A. The Definition of "Cross Platform Cloud Applications", 2013. Disponível em: <https://blogs.vmware.com/management/2013/01/the-definition-of-cross-platform-cloud-applications.html>. Acesso em: 10 fev. 2014.

FREIRE, R. C. S. Conversão A/D e D/A. Disponível em: <http://www.die.ufpi.br/ercemapi2011/escolas/conversao_ad_%20e_da.pdf>. Acessado em 10 fev. 2014.

IEEE. Institute of Electrical and Electronics Engineers. ISO/IEC/IEEE 9945: Information technology -- Portable Operating System Interface (POSIX®) Base Specifications, Issue 7. Suíça, 2009.

LOVE, R. Linux Kernel Development, 3ª ed, Addison Wesley, 2010.

MAZIERO, C. A. Sistemas Operacionais: Conceitos e Mecanismos, 2013, Disponível em: <http://dainf.ct.utfpr.edu.br/~maziero/lib/exe/fetch.php/so:so-cap04.pdf>. Acessado em 10 fev. 2014.

RICARTE, I. L. M. O pré-processador C, 2003. Disponível em: <www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node150.html>. Acesso em: 10 fev. 2014.

SEIXAS FILHO, C. Fundamentos e Aplicações de Sistemas de Automação, 2005. Disponível em: <http://www.cpdee.ufmg.br/~seixas/Especializacao/Download/DownloadFiles/ATR_Cap4.pdf>. Acesso em: 10 fev. 2014.

SILBERSCHATZ, A., Galvin, P. B., Gagne, G. Operating System Concepts, 8ª ed, John Wiley & Sons, 2008.

SOUZA, V. A. Multiplexação por Divisão do Tempo. Disponível em: <http://www.cerne-tec.com.br/TDM.pdf>. Acessado em 10 fev. 2014.

SOMMERVILLE, I. Engenharia de Software, 8ª ed, Addison Wesley, 2007.

TAUBENFELD, G. The Black-White Bakery Algorithm and related bounded-space, adaptive, local-spinning and FIFO algorithms, 2004. Disponível em: <http://www.cs.tau.ac.il/~afek/gadi.pdf >. Acesso em: 10 fev. 2014.