Contribuições para o Sistema Digital De Seleção Online de Múons a Partir de Informações de Calorimetria no Detector Atlas

Linton T. C. Esteves, Eduardo F. Simas Filho, Paulo C. M. A. Farias, Luciano M. Andrade, Augusto S. Cerqueira, Victor A. Ferraz, Rafael G. Gama e José M. de Seixas

***Resumo*— O sistema de seleção online de eventos (*trigger*) do detector ATLAS é dividido em três estágios sequenciais, onde o primeiro é implementado em hardware dedicado para atender às severas restrições de tempo de processamento e utiliza separadamente informações de três subdetectores (traços, calorímetro e câmara de múons). Este trabalho está inserido no contexto de um sistema eletrônico para permitir a utilização de informação do calorímetro (medidor de energia) na detecção de múons no primeiro nível de *trigger* do ATLAS. O artigo apresenta resultados relativos à implementação de um módulo digital para controle do fluxo de informações na placa desenvolvida.**

***Palavras-Chave*— FPGA, ATLAS, LHC, detecção de múon.**

***Abstract*— The ATLAS trigger system concerns three sequential selection levels. The first level is implemented in hardware, in order to comply with the severe time latency restriction and uses information from three sub-detectors (tracking, calorimeter and muon system) to produce the trigger decision. This work deals with an electronic system which allows to use calorimeter (energy measurement system) information for muons identification in the first trigger level. In this article the results of the implementation of a data control digital module are presented.**

***Keywords*— FPGA, ATLAS, LHC, muon detection.**

I. Introdução

O detector de partículas ATLAS (*A Toroidal LHC ApparatuS*)[1] opera no grande colisionador de hádrons (do inglês *Large Hadron Collider*) (LHC)[2] com o objetivo de mensurar os fenômenos físicos de interesse gerados nas colisões entre prótons. Atualmente, existem três sub-detectores no ATLAS: o detector de traços, o calorímetro e o detector de múons, responsáveis pela detecção da trajetória das partículas, da energia depositada e da trajetória dos múons, respectivamente [1].

Devido à complexidade e raridade dos fenômenos observados é necessário produzir uma grande quantidade de colisões e mensurar seus resultados através de sensores distribuídos nos sub-detectores. Com isso, é gerada uma elevada taxa de informações (da ordem de 60 TB/s), o que torna necessário realizar um processo de seleção online (*trigger*) das informações de interesse.

Como apresentado em [3], existem estudos que apontam para uma deterioração no desempenho do detector de múons com o início do segundo período de operações do LHC (conhecido como *Run 2*). Este efeito está relacionado ao aumento do ruído de fundo, devido à quantidade maior de colisões.

Esse problema pode ser contornado com uma filtragem cruzada, ao combinar as informações do calorímetro de telhas (TileCal, do inglês *Tile Calorimeter*) [4] com um dos detectores de múons o *Thin Gap Chambers* (TGC). Esta associação possibilita reduzir a quantidade de falsos positivos (sinais que são confundidos com a informação de interesse) repassados ao próximo nível do *trigger* de múons [5]. Os componentes necessários à viabilização dessa solução foram então agrupados em um módulo denominado *Tile-Muon Digitizer Board* (TMDB).

Após realizar a filtragem, as amostras aceitas devem ser encapsuladas em um formato específico e então repassadas para o segundo nível de *trigger*. No entanto, devido à latência inerente ao processo de seleção, essas informações necessitam ser armazenadas em uma estrutura temporária de memória até que o processo de análise seja concluído. Essa etapa de armazenamento, empacotamento e envio de informações relacionadas aos múons detectados foi atribuída ao ROD *Fragment Builder*, modulo digital a ser embarcado no TMDB e foco desse trabalho.

O artigo foi dividido em cinco seções: A seção II aborda a utilização de informações do calorímetro na detecção de múons juntamente com uma breve descrição do TMDB; Na seção III é apresentada uma solução para o problema de armazenamento e envio, para o próximo nível de seleção, das informações relacionadas a um evento de *trigger*; A seção IV apresentada os resultados obtidos a partir das contribuições do projeto. Por fim, a seção V apresenta as conclusões desse trabalho.

II. Detecção de múons assistida pelo Calorímetro

A partir de dados coletados no TileCal a uma alta taxa de digitalização (2 GHz) com o auxílio de um osciloscópio [6], foi possível extrair informações suficientes para a construção de um modelo do pulso típico de um sinal de múon [6].

Linton T. C. Esteves, Eduardo F. Simas Filho e Paulo C. M. A. Farias, Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal da Bahia, Salvador, Bahia, Brasil. Luciano M. Andrade e Augusto S. Cerqueira, Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de Juiz de Fora, Juiz de Fora, Minas Gerais, Brasil. Victor A. Ferraz, Rafael G. Gama e José M. de Seixas, Laboratório de Processamento de Sinais COPPE/POLI, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil. E-mails: lintonthiago@gmail.com, eduardo.simas@ufba.br, paulo.farias@ufba.br, luciano.andrade@engenharia.ufjf.br, augusto.santiago@ufjf.edu.br, victor.ecomp@gmail.com, rafael.g.gama@gmail.com, seixas@lps.ufrj.br

O modelo permitiu a criação de um sistema de seleção responsável por identificar um sinal de múon. Quando ocorre a detecção de um múon, e o mesmo atravessa o calorímetro, o sinal lido é identificado como um possível candidato a múon, caso contrário é considerado ruído. Utilizando informações de detecção dos algoritmos do último nível do sistema de *trigger* (processamento *offline*) foi possível distinguir os dois tipos de assinaturas (múon e ruído) e então compara-las com os resultados do sistema combinado.

A eficiência do sistema proposto (círculos) e a taxa de falsos positivos (triângulos) pode ser observada através da Figura 1. A partir desta imagem é possível observar que para um limiar de 500 MeV[1], a eficiência do sistema é de 93% com uma taxa de falsos múons de 17% [3].

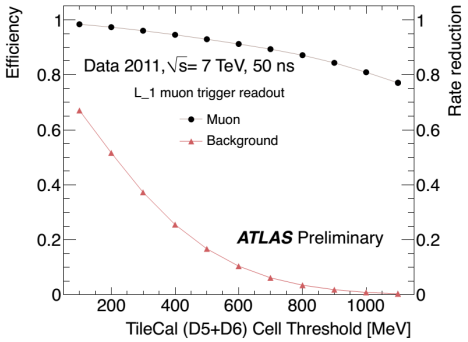


Fig. 1. Eficiência do sistema na região da tampa (Extraído de [7]).

Após a etapa de validação do modelo foi então desenvolvido o TMDB, módulo eletrônico responsável por embarcar os componentes necessários à implementação da solução e realizar a interface entre o TGC, o TileCal e o sistema de *trigger* do ATLAS. Cada TMDB possuí 32 canais de conversão analógico-digital, permitindo realizar o processamento de até 8 módulos do barril estendido do TileCal [8].

O processamento dos sinais e controle (Core) é realizado em um FPGA Spartan 6 da Xilinx acoplado ao TMDB. Internamente, o Core é subdividido em diversos sub-módulos (Figura 2). Nesse trabalho serão descritos a Unidade de Processamento (do inglês *Module Processing Unit*) (MPU) e o ROD *fragment builder*. A MPU tem como função processar os sinais de um módulo do TileCal objetivando extrair informações de energia que possam indicar a presença de um múon. Já o O ROD *Fragment Builder* (ou Controlador de pacotes) é responsável por construir um bloco de dados, denominado ROD *fragment*, contendo informações relevantes sobre o processamento realizado em um TMDB. Após construído, o ROD *fragment* é repassado ao módulo *High-speed Optical Link for ATLAS* (HOLA) [9], e então encaminhado ao barramento S-Link [10]

III. ROD *Fragment Builder*

Quando o TMDB aponta um candidato a múon, é necessário repassar para o próximo nível de seleção um conjunto de informações de interesse. Essas informações são encapsuladas

[1]Unidade de medida de energia. Um eV equivale a 1,602 177 33 (49) x 10^{-19} joules.

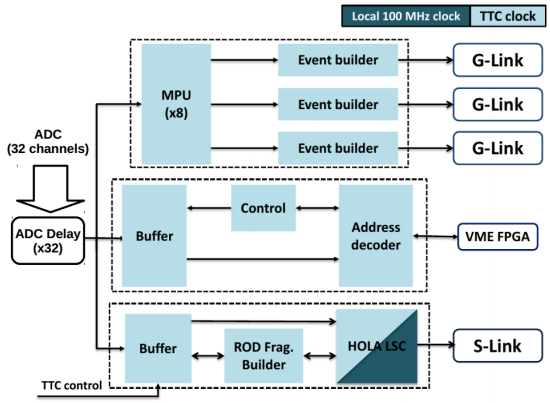


Fig. 2. Diagrama de blocos do core da TMDB

em um fragmento de dados de saída (ROD *Fragment*) dividido em três seções: i) dados brutos da conversão analógico-digital; ii) respostas das MPU do TMDB; iii) resultados dos filtros. Além disso, dados para fins de identificação e controle são adicionados.

Essa etapa apresenta uma importância crucial para o projeto como um todo, uma vez que o seu funcionamento incorreto implicaria em uma falha no repasse das informações de interesse para o próximo nível de seleção, inviabilizando a solução proposta.

Nas subseções a seguir serão apresentados os requisitos do projeto, seguidos da arquitetura desenvolvida.

A. *Requisitos do Sistema*

Devido às informações de interesse serem geradas antes do término da detecção, foi necessário criar uma estrutura que armazenasse esses sinais até que o processo de análise fosse concluído. Como cada canal opera de forma independente, seria necessário uma estrutura separada por canal.

Apesar dos sinais de *trigger* ocorrerem, em média, a uma baixa frequência (100 kHz), podem existir intervalos de tempo menores entre eventos. O projeto deve ser capaz de identificar e processar corretamente esse tipo de ocorrência operando a uma frequência de 40 MHz, mesma frequência de operação dos *Analog-to-Digital Converters* (ADC) e da MPU.

Além disso, como os sensores estão fisicamente espalhados os sinais gerados possuem caminhos diferentes, o que gera uma variação entre o tempo de propagação dos sinais provenientes dos ADCs até o TMDB. Dessa forma, é necessário realizar um processo de sincronização com o intuito de compensar os atrasos existentes.

Ademais, ao detectar um *trigger*, além da amostra do ADC correspondente ao evento, devem ser também repassados os seus seis vizinhos mais próximos, tornando possível representar um pulso do calorímetro de forma eficaz.

Foi necessário também identificar as interfaces externas do projeto, como é possível observar na Figura 3. A solução desenvolvida neste trabalho (ROD *Fragment Builder*) se comunica com dois blocos internos do TMDB- a MPU e o HOLA - além de se comunicar com os ADCs do TileCal. A MPU fornece os sinais de controle e configuração, informando ao

ROD *Fragment Builder* quando um *trigger* ocorre, além de parte dos dados que se deseja enviar. Após a ocorrência de um *trigger*, os dados selecionados devem ser enviados através de um barramento de 33 bits para o módulo HOLA, responsável por encaminhá-los para o próximo nível de seleção. A interface com os ADCs é implementada através de 32 portas, com 8 bits cada, operando a uma frequência de 40 MHz.

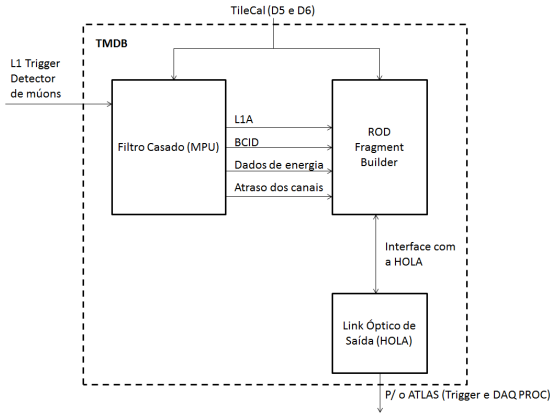


Fig. 3. Conexão entre o ROD *Fragment Builder*, o TMDB e a HOLA (Fonte: próprio autor).

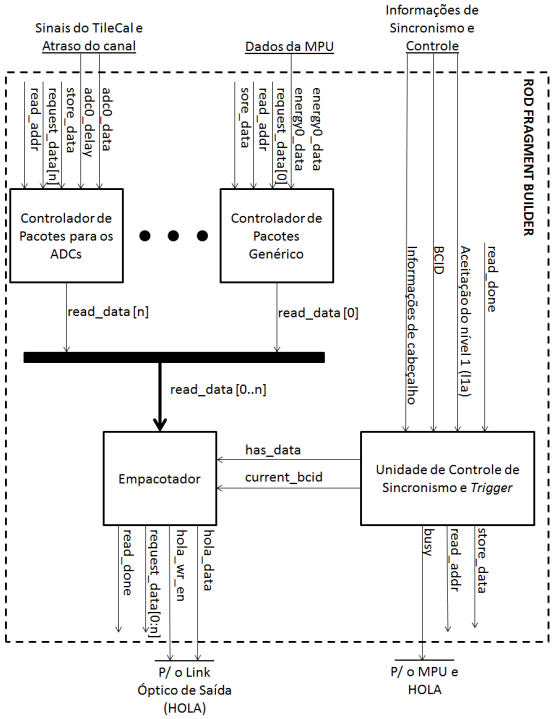


Fig. 4. Diagrama do topo do controlador de pacotes

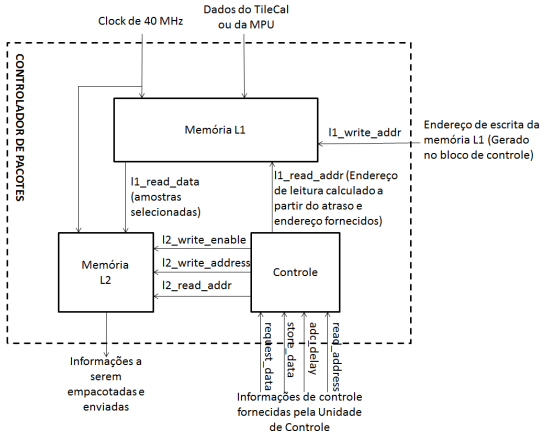


Fig. 5. Controlador de Pacotes do ADC

B. Arquitetura Concebida

Objetivando atender as especificações listadas, foi desenvolvida uma arquitetura dividida em quatro blocos principais: uma unidade de controle (*ctrl*), um controlador de pacotes de 32 bits (*packet_ctrl*), um controlador de pacotes de 8 bits para os ADCs (*packet_ctrl_adc*) e um módulo empacotador (*gen_packet*) (Figura 4).

Os controladores de pacotes (*packet_ctrl_adc* e o *packet_ctrl*) são os módulos responsáveis por receber e armazenar as informações de interesse provenientes tanto da MPU quanto dos ADCs. Permitindo, dessa forma, seu acesso quando um *trigger* ocorrer (Figura 5). Internamente, são compostos por dois níveis de memória e uma lógica de controle complementar. O primeiro nível, *l1_mem*, assim como todos os demais blocos, opera a 40 MHz, mesma frequência dos conversores analógico-digitais e da MPU, armazenando as amostras recebidas em uma memória circular com 256 posições.

Uma das funções especificas dos controladores de pacotes do ADC é compensar os atrasos entre os ADCs. Após identificado, o valor de atraso é repassado a esse bloco através de um registrador de configuração, que pode ser acessado via barramento VME [11]. Quando um *trigger* é gerado, o controlador de pacotes do ADC recebe um sinal indicando sua ocorrência juntamente com o endereço da memória de nível 1 relacionado ao evento. O endereço de leitura final é encontrado ao somar o endereço recebido com o valor de atraso previamente configurado. O valor lido no endereço resultante é então armazenado em um segundo nível de memória (*l2_mem*).

O *packet_ctrl* funciona de forma semelhante ao *packet_ctrl_adc*. Contudo, os sinais recebidos no primeiro não possuem problemas de sincronização, visto que são gerados internamente no TMDB, ademais, apenas uma amostra é armazenada na *l2_mem*. Com capacidade para armazenar amostras relacionadas a até sete sinais de *triggers* diferentes, a *l2_mem* opera como uma memória FIFO (*First In First Out*).

A unidade de controle é o módulo responsável por receber o sinal de *trigger* proveniente do algoritmo de seleção e repassá-lo para os controladores de pacotes. Visto que o sinal de *trigger* não ocorre a uma frequência constante, é possível que ocorram eventos muito próximos entre si. Além disso, como o processo de transferência dos dados da l1_mem para a l2_mem nos controladores de pacote é realizado de forma sequencial, caso o sinal de *trigger* fosse repassado antes da conclusão de uma transferência um problema de sobreposição poderia ser causado. Intentando evitar a ocorrência desses dois fenômenos, a unidade de controle implementa um sistema de espera ao

armazenar os *triggers* recebidos em uma FIFO. Deste modo, o próximo sinal de *trigger* é repassado para os controladores de pacotes no mínimo sete pulsos de *clock* após o anterior, tempo necessário a realização de uma transferência entre as memórias de nível um e dois. Sempre que a fila de *triggers* está cheia, um sinal é enviado ao TMDB informando que eventos podem ser descartados.

Quando os dados são repassados ao segundo nível de *buffer* dos controladores de pacote, é gerada uma interrupção no bloco *gen_pkt* informando que há dados prontos para serem enviados. Nesse momento, são lidos e enviados sequencialmente para a HOLA seguindo o ROD *fragment format* [12]. A cada amostra lida dos controladores de pacotes do ADC é gerada uma requisição pelo *gen_pkt* solicitando que a próxima amostra da *l2_mem* seja disponibilizada.

O pacote gerado é composto de basicamente três campos: *fragment header*, *sub-fragment* e *fragment end*.

No *fragment header* estão contidas informações de identificação do pacote como: início do fragmento, versão do formato utilizado, *Bunch Crossing Identification* (BCID), *Event Identifier* (LV1ID) e tipo de *trigger*. Essas informações poderão ser utilizadas posteriormente para identificar o pacote [12].

Logo após o *fragment header* é enviado o *sub-fragment*, contendo os dados recebidos tanto da MPU quanto os coletados dos ADCs. São gerados três *sub-fragment*, um para cada tipo de dados, cada um com seu cabeçalho específico contendo informações como o seu tamanho e tipo. Como os dados provenientes dos ADCs são de 8 bits e o formato de dados é de 33 bits, sendo um reservado, são agrupados 4 palavras dos ADCs por envio. Ao final, um *fragment end* é enviado informando o final de um pacote [12]. Sequencialmente, é gerado um sinal informando ao bloco de controle que o processo de transferência foi concluído e que novos envios podem ser iniciados.

Após a conclusão dos blocos funcionais do projeto foi inserida uma estrutura de *debug* que poderia ser acessada externamente via um barramento VME. Através dessa estrutura é possível acessar diversos tipos de informações: quantidade de pacotes enviados; quantidade de ocorrências do estouro da *fifo* de *triggers*; quantidade de *triggers* recebidos; LV1ID atual e ultimo enviado; ultimo BCID recebido.

IV. Resultados

A partir das especificações e da arquitetura do projeto, foram identificadas funcionalidades críticas para o correto funcionamento do sistema. Sendo assim, foram desenvolvidos cenários de testes específicos para esses pontos: i) Detecção correta de um *trigger*; ii) Seleção dos dados relacionados ao *trigger* recebido; iii) Ajuste do atraso entre os ADCs; iv) Empacotamento e envio correto dos dados; v) Sistema de espera de *triggers*.

Na Figura 6 é ilustrado o comportamento do bloco de controle durante a ocorrência de um sinal de *trigger*, nesse momento, a solução proposta foi capaz de detectá-lo, armazenar o valor do BCID relacionado e informar para os demais blocos do sistema que existe um *trigger* a ser enviado. Ao mesmo tempo, o BCID armazenado é disponibilizado para leitura.

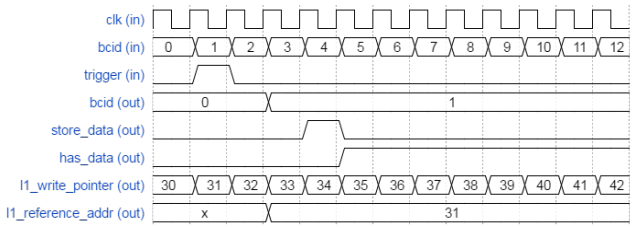


Fig. 6. Diagrama de sinais digitais na ocorrência de um *triggers*

Sequencialmente, foi testado o processo de armazenamento e envio dos dados correspondentes ao sinal de *trigger* gerado. Para esse cenário foram inseridos valores conhecidos nas entradas correspondentes aos ADCs, e paralelamente, foram inseridos sinais de *trigger* a uma frequência conhecida. Ao final, os valores da saída foram comparados com os esperados comprovando o correto funcionamento do módulo. Na Figura 7 o processo de transferência de amostras do ADC entre as memórias de nível um e dois pode ser observado. Na imagem, como foi configurado um atraso de três unidades, a amostra desejada está localizada três amostras antes do endereço recebido.

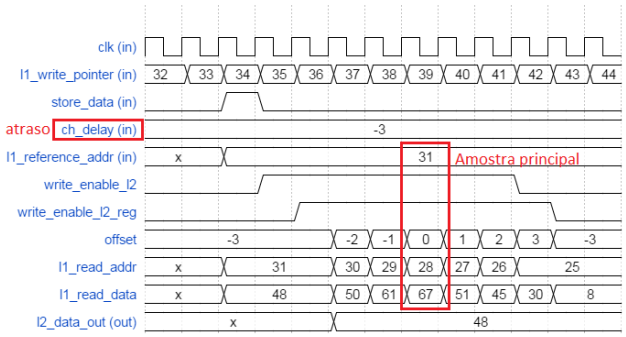


Fig. 7. Transferência dos dados do ADC para a *l2_mem*

O último teste implementado em simulação foi validar o sistema de espera construído. Sendo assim, foram gerados *triggers* com diferentes configurações: de forma continua; com intervalos aleatórios; por fim, sequencias entre um e oito *triggers* sem intervalos.

Objetivando verificar se o sistema está apto a suportar a taxa de eventos do ATLAS, se calculou o período necessário necessários para se enviar um pacote. Para esse calculo se considerou que o período se inicia com a detecção de um evento de *trigger* e termina no momento em que a última informação relacionada é enviada. A partir das informações coletadas obteve-se uma taxa máxima de transferência de dados (*throughput*) de 360 KHz. Como os eventos de múon ocorrem em média a 100 KHz a taxa de envio de *triggers* acaba sendo, na média, aproximadamente três vezes maior que a de recepção. Essa característica permite compensar eventuais sobrecargas nas ocorrências de *triggers*. A partir dos testes realizados, foi observado que a utilização das memórias de nível 2 com oito níveis de profundidade seria suficiente para que não ocorressem perdas de pacotes.

Após a conclusão do processo de simulação, o projeto foi então embarcado em um dispositivo XC6SLX150T, da

família Spartan 6 da Xilinx, onde foram extraídos dados como consumo, temporização e recursos utilizados (Tabela I).

TABELA I
RESULTADOS DA SÍNTESE DO PROJETO PARA O XC6SLX150T

Consumo	
Item	Potência (mW)
Clock	9,0
Lógica	7,0
Memórias	92,0
IOs	6,0
Leakage	118,0
Total	250,0

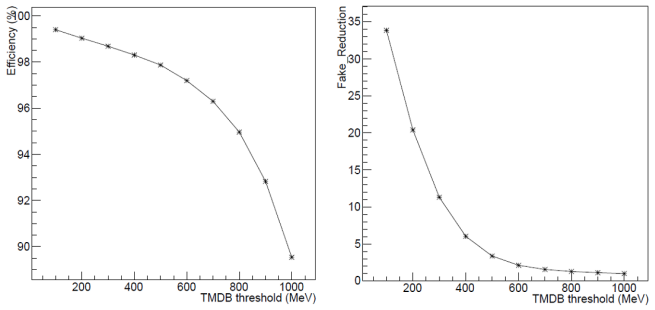
Temporização	
Tipo	Pior caso
Setup Time	11,316 ns
Hold Time	0,254 ns

Recursos Utilizados	
Recurso	Utilização
Registrador	1%
LUT	3%
Memoria	50%

A partir dos dados levantados, foi possível então identificar que o projeto não apresenta problemas de temporização. Apesar da alta utilização das memórias do FPGA, a quantidade utilizada se mostrou dentro do previsto, pois os outros módulos do TMDB não utilizam a mesma quantidade de memória que o ROD *fragment builder*. Por outro lado, recursos como *Look-up Table* (LUT) e registradores não apresentaram grande utilização. Da mesma forma, os resultados da análise de potência foram proporcionais à utilização dos recursos.

Finalmente, o ROD Fragment Builder foi então inserido em um FPGA para a análise de eficiência/rejeição do sistema combinado com os outros módulos do TMDB, foram utilizados dados de uma coleta experimental no ATLAS realizada em maio de 2016.

Dessa forma, como exibido na Figura 8 para resultados experimentais do TileCal, a eficiência ao utilizar um limiar de 500 MeV é de 97,87%. Para o mesmo cenário, o falso alarme foi de 3,52%.



(a) Eficiência (b) Falso alarme

Fig. 8. Eficiência de detecção do sistema combinado

O resultado obtido com este teste se mostrou melhor do que os esperados a partir das análises realizadas em simulação (Figura 1). Assim, foi possível confirmar que o ROD *fragment builder*, juntamente com os outros módulos do TMDB, realizou suas funções de forma correta, analisando os possíveis candidatos a múons e repassando para o sistema de seleção do ATLAS as informações coletadas.

V. CONCLUSÃO

O presente trabalho está inserido no problema da detecção de sinais de múons imersos em grande quantidade de ruído de fundo no detector ATLAS. Esse trabalho abordou o desenvolvido do ROD *fragment builder*, componente responsável por coletar informações geradas a partir da análise dos sinais do TileCal e repassá-las para o próximo nível de seleção do ATLAS, quando um candidato a múon for identificado.

Após a identificação e correção dos problemas encontrados, o módulo empacotador de dados conseguiu realizar com eficiência sua função ao implementar o processo de armazenamento, espera e empacotamento sem erros e a uma taxa superior à ocorrência de *triggers*, evitando assim a sobreposição de amostras. Além disso, foram realizados testes em campo no detector, o que permitiu a aceitação do projeto como um dos componentes do módulo TMDB que atualmente opera no *A Toroidal LHC ApparatuS*.

A partir dos dados obtidos até o momento foi possível constatar que o sistema combinado apresentou resultados satisfatórios, atingindo eficiência em torno de 97% na detecção de múons e taxa de rejeição próxima a 3%.

Finalmente, é importante ressaltar que durante a nova fase de operação do ATLAS serão gerados níveis maiores de luminosidade, o que fará com que o TileCal opere com uma nova eletrônica de aquisição de dados. Nesse contexto, pode ser necessário desenvolver novos módulos para o projeto TileMuon.

AGRADECIMENTOS

Os autores agradecem à FAPESB, FAPERJ e RENAFAE pelo apoio concedido para a realização deste trabalho e à Colaboração ATLAS pela disponibilização dos dados utilizados e pelas importantes contribuições.

REFERÊNCIAS

[1] G. Aad, E. Abat, J. Abdallah, A. Abdelalim, A. Abdesselam, O. Abdinov, B. Abi, M. Abolins, H. Abramowicz, E. Acerbi *et al.*, "The atlas experiment at the cern large hadron collider," *Journal of Instrumentation*, vol. 3, no. S08003, 2008.

[2] O. Brüning, H. Burkhardt, and S. Myers, "The large hadron collider," *Progress in Particle and Nuclear Physics*, vol. 67, no. 3, pp. 705–734, 2012.

[3] V. Ferraz Araujo, "Detecção online eficiente de eventos raros utilizando detectores finamente segmentados," Tech. Rep., 2016.

[4] ATLAS Collaboration, *ATLAS tile calorimeter: Technical design report*. CERN, 1996.

[5] T. Ciodaro, J. de Seixas, and A. Cerqueira, "Use of hadronic calorimetry information in the atlas level-1 muon trigger," *IEEE Transactions on Nuclear Science*, vol. 61, no. 2, pp. 1047–1055, 2014.

[6] T. Ciodaro, "Identificação online de sinais baseada em calorimetria de altas energias e fina segmentação," Ph.D. dissertation, Rio de Janeiro Federal U., 2012.

[7] A. collaboration *et al.*, "Technical design report for the phase-i upgrade of the atlas tdaq system," *CERN-LHCC-2013-018, ATLAS-TDR-023*, 2013.

[8] A. Ryzhov, "The level-1 tile-muon trigger in the tile calorimeter upgrade program," *Journal of Instrumentation*, vol. 11, no. 12, p. C12049, 2016.

[9] H. CERN, "High-speed optical link for atlas," Disponivel em: $http://hsi.web.cern.ch/HSI/s-link/devices/hola/$ Acesso em: 10 de abril de 2017, 2016.

[10] H. Van der Bij, R. McLaren, O. Boyle, and G. Rubin, "S-link, a data link interface specification for the lhc era," in *Nuclear Science Symposium, 1996. Conference Record., 1996 IEEE*, vol. 1. IEEE, 1996, pp. 465–469.

[11] W. D. Peterson, "The vmebus handbook–a user's guide to the vme64 and vme64x bus specification," 1997.

[12] C. Bee, G. Mornacchi, L. Mapelli, F. Wickens, R. McLaren, J. Petersen, and D. Francis, "The raw event format in the atlas trigger & daq," Tech. Rep., 1998.