Um framework para coprojeto de hardware e software de sistemas ADAS baseados em visão

Leandro A. Martinez[1], Tiago Lobo[1], Eduardo Marques[1]

[1] Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo
Av. Trabalhador São-Carlense, 400 – CEP 13566-590 São Carlos, SP, Brasil

lmartinez@usp.br, tiagomelobo@gmail.com, emarques@usp.br

Abstract. *This paper presents a framework for hardware and software co-design to building systems designed to driver assistant using computer vision. This work is part of a doctoral research project nearing completion. To validate the model, a modular pedestrian detection is implemented by comparing the results obtained with other design.*

Resumo. *Este artigo apresenta um framework para co-projeto de hardware e software para a construção de sistemas destinados ao apoio de motoristas utilizando visão computacional. Este trabalho faz parte de um projeto de pesquisa de doutorado em fase de conclusão. Para sua validação, um sistema modular de detecção de pedestres é implementado comparando-se os resultados obtidos com outro projeto.*

1. Introdução

Evitar acidentes de trânsito ainda é um dos grandes problemas que o mundo enfrenta. Com as novas tecnologias de sistemas embarcados surgem diversas propostas que precisam ser exaustivamente validadas e testadas. Contudo, simular e testar novas propostas nos diversos possíveis ambientes unindo algoritmos de software com hardware reprogramável torna o problema complexo e multidisciplinar dentro da computação. Assim, com objetivo de contribuir nesta área de pesquisa, este artigo apresenta um framework para co-projeto de hardware e software para a construção de sistemas destinados ao apoio de motoristas que utilizam visão computacional.

Os processos e componentes utilizados para construir sistemas avançados para auxílio a motoristas (ADAS - Advanced Driver Assistance Systems) devem ser seguros e confiáveis. Validação e testes exaustivos acarretam em um enorme investimento tornando-se um dos aspectos mais desafiadores do desenvolvimento de ADAS, especialmente quando isso acontece com sistemas de visão. Para testar todos os cenários e produzir precisão 100%, com zero falsos positivos, e em todas as condições possíveis, necessita-se de uma grande quantidade de horas de clipes de vídeo para que possam ser reunidos e executados em banco de dados para teste de regressão (Kisačanin e Gelautz, 2014) (Bengler et al. 2014).

Nesse contexto, o framework apresentado, visa facilitar o desenvolvimento de aplicações permitindo melhor explorar o espaço de projeto, e assim contribuir para um ganho de desempenho no desenvolvimento de sistemas embarcados principalmente quando comparados à construção totalmente em hardware. Um dos recursos do sistema é a possibilidade da simulação da aplicação antes da síntese em uma plataforma reconfigurável.

Nota-se que a taxa de reuso em circuitos lógicos na indústria deverá crescer linearmente. Em 2011 essa taxa era de 54% e deverá chegar a 98% em 2026. Com estes pressupostos, mantendo constante esforço de projeto em SOC faz-se necessário uma melhoria da produtividade em projetos em dez vezes longo dos próximos dez anos. Para resolver o desafio da produtividade, várias abordagens devem ser combinadas. Em primeiro lugar, os níveis de abstração de projetos devem ser elevados. Em segundo lugar, deve ser aumentado o grau de automação, particularmente na verificação e implementação de projetos (ITRS, 2011)

2. Frameworks de Visão Computacional

Existem diversos frameworks para processamento de imagem usados em sistemas de visão automotiva e estão em constante evolução, no entanto, muitos destes projetos encontram-se com alguma desvantagem: alguns foram descontinuados, outros são inadequados para processamento de visão em tempo real com desenvolvimento rápido, muitos carecem de uma interface amigável com o usuário, e por fim os que não possuem design modular. A seguir, uma relação de alguns importantes sistemas:

O Baselabs fornece a infraestrutura para o desenvolvimento de ADAS de prototipagem rápida para desenvolver ADAS permitindo a captura, registo e reprodução de dados do sensor e da fusão de dados de sensores complexos. Disponibiliza uma interface gráfica do usuário com vários componentes (Baselabs, 2016).

Intempora RTMaps é um framework para prototipagem algoritmos com um ambiente modular para testar e avaliar funções com base em diferentes conjuntos de sensores e em diferentes configurações. Também fornece uma biblioteca de componentes preparados para desenvolver ADAS. É extensível permitindo a personalização de módulos pelo usuário (Intempora, 2016).

Comemso ADTF é também um framework para sistemas de processamento de imagem usado na indústria automobilística. Permite a criação de filtros de imagem e conexões de interfaces de hardware para sistemas embarcados. Também fornece ferramentas para teste e simulação (Comemso, 2016).

MontiVision fornece um conjunto de ferramentas que permite o desenvolvimento de aplicações de processamento de imagem de forma personalizada. O kit de desenvolvimento inclui vários módulos de processamento de imagem. Os filtros de imagens podem ser definidos e personalizados pelo usuário através de uma interface visual. Também inclui um conjunto exemplos para demonstrar o uso (MontiVison, 2016).

ImprovCV oferece um sistema modular de código aberto para processamento de fluxo de dados de visão. O software permite interatividade rápida com o usuário para o desenvolvimento de aplicações de visão ADAS (Boeing e Braunl, 2008).

NI LabVIEW é o software base da plataforma de projeto da National Instruments, para o desenvolvimento de sistemas de medição ou controle. Integrando diversas ferramentas e com um ambiente de desenvolvimento voltado à resolução de problemas, produtividade acelerada e inovação contínua (LabVIEW, 2016).

MATLAB HDL Coder e HDL Verifier fornece um ambiente de projeto integrado que acelera o desenvolvimento em FPGA e ASIC, integrando ferramentas de design e IP (Intellectual Property) para Xilinx e Altera. Este ambiente fornece um sistema de programação extensível, bem como uma arquitetura dataflow (MATLAB, 2016).

Infelizmente, a maioria desses sistemas não são de código aberto e poucos fazem a integração entre hardware e software. Dessa forma, a proposta deste trabalho é apresentar um ambiente de desenvolvimento que permita consolidar as principais vantagens desses sistemas. Destacando-se:

- Código fonte aberto para permitir a expansão do projeto;
- Simulação funcional da integração de IPs software com IPs hardware no mesmo ambiente;
- Opção para compilar o projeto criado pelo usuário para FPGA.

3. Características do sistema

Utilizando-se como plataforma base um software de código aberto (Blender, 2016) modificado com diversas implementações, o Framework Vision-ADAS apresenta ao usuário uma interface de fluxo de dados de fácil utilização (Figura 1). Usando nós de composição é possível fazer várias combinações de filtros em fluxos de vídeo ou imagens.

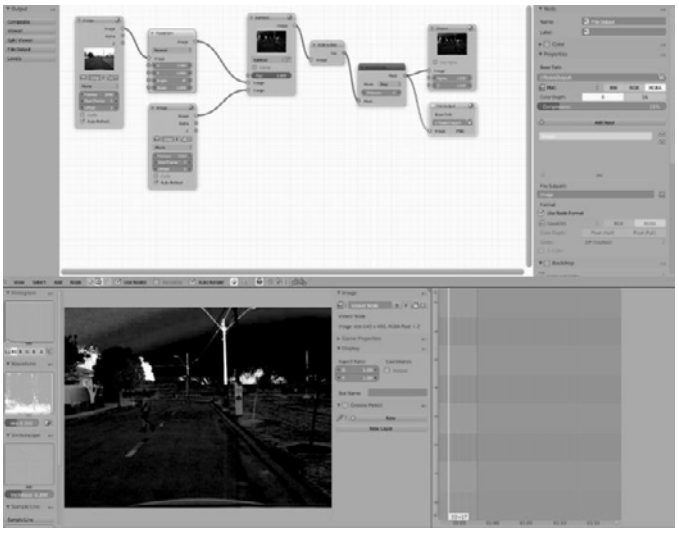

Figura 1. Tela principal da ferramenta de desenvolvimento

3.1. Ambiente de Projetos

A plataforma oferece muitos recursos para processamento de imagens. No conjunto, disponibiliza-se uma biblioteca própria de blocos que podem ser compilados pela

ferramenta para utilização em FPGA. Permite-se também utilizar filtros ainda não disponíveis para versões embarcadas de hardware e software, possibilitando simulações e entrada de dados para os outros blocos já disponíveis para embarcados.

Através de blocos IP (propriedade intelectual) o usuário pode arrastar e soltar filtros de processamento de imagem para a interface do usuário e exibir o seu efeito diretamente sobre o vídeo selecionado. Cada filtro tem parâmetros que podem ser ajustados em tempo real para fornecer feedback imediato. Permitindo testes interativos completos de sistemas de visão e simplificando ajuste de parâmetros.

3.2. Simulação

O nó de entrada de vídeo pode ser conectado permitindo comparações diretas de vários algoritmos de visão computacional. Uma vez que o aplicativo é um software de código aberto pode ser estendido ou adaptado para interoperar com outros sistemas.

Há três tipos principais de blocos:

- IP apenas para a simulação: Todos os nós podem ser utilizados para as simulações, contudo, alguns não estão disponíveis para serem embarcados diretamente em FPGA. Normalmente são codificados para ensaios e posteriormente convertidos. Como exemplo, podem ser disponibilizados bibliotecas de filtros provenientes do OpenCV (OpenCV, 2016), que podem não estar disponíveis para a compilação final. A grande vantagem deste tipo de bloco é a possibilidade de utilizar outras aplicações baseadas em C como protótipos para validar os resultados esperados.
- IP pronto para ser incorporado no processador (SoC): Os nós podem ter códigos pré-programados para serem usados em um processador embarcado. Durante a compilação, sub-rotinas de execução são inclusas com o código para ser usado pelo processador embarcado.
- IP pronto para ser sintetizado em hardware: São os blocos preparados para geração de hardware. Neste caso, o sistema utiliza códigos pré-compilados de um IP escritos em HDL (Hardware Description Language) para simular parte da aplicação. Na fase de compilação do projeto, um código de script é criado para geração do hardware, contendo os blocos e os controles de execução.

Neste ambiente, a criação de um novo IP pode ser validada por uma equipe de software, em seguida, convertido em uma versão de hardware ou até mesmo um software embarcado. Dessa forma, um bloco de software ou um bloco de hardware pode usar suas entradas e saídas para contribuir na construção e validação de seu complemento. Este método favorece uma melhor cooperação entre as equipes de desenvolvimento.

3.3. Arquitetura Geral do Framework

O objetivo final deste framework é o envio do projeto criado pelo usuário em uma arquitetura de hardware. Neste caso, após a validação em simulação funcional, existem algumas etapas a serem seguidas. A Figura 2 mostra um fluxo completo usando a estrutura para uma validação completa em FPGA.

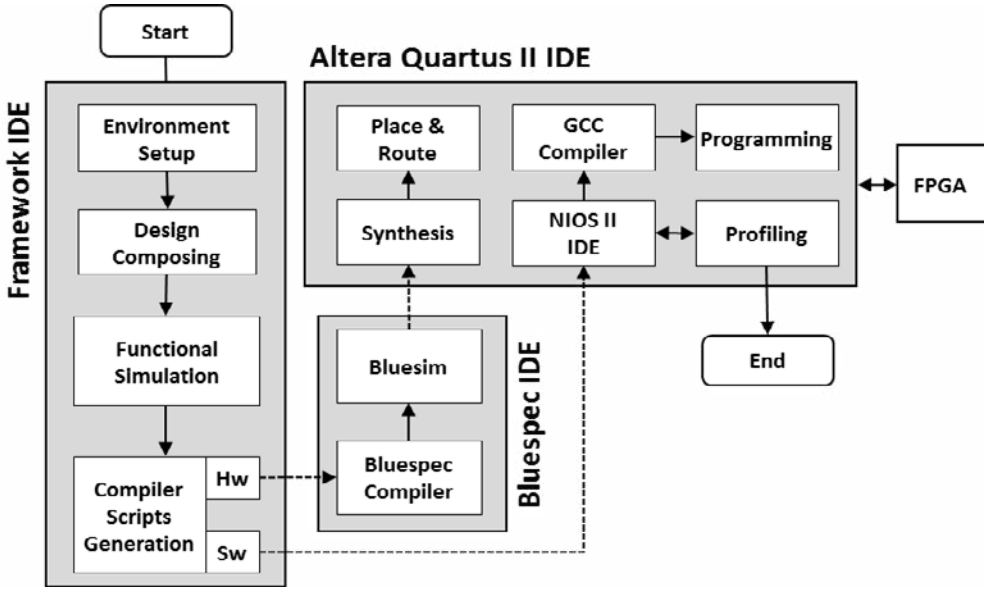

Figura 2. Fluxo completo usando o framework

Configuração do Ambiente: Ao iniciar a ferramenta, o primeiro passo é selecionar a placa FPGA a qual se destina o projeto. Em seguida, o usuário pode iniciar um projeto vazio ou selecionar um modelo base desenvolvido para permitir alterações no projeto. Os modelos são projetos validados que contêm exemplos de uso da ferramenta para vários ADAS.

Composição do Projeto: O sistema fornece uma biblioteca de IPs onde o usuário pode arrastar e soltar para o espaço de trabalho e ligá-los a outros blocos. O IP pode ser hardware ou software, dependendo da escolha do usuário e disponibilidade na biblioteca. Cada IP tem um conjunto de parâmetros que podem ser modificados durante a manutenção do projeto.

Simulação Funcional: Durante a elaboração do projeto, pode-se fazer uma simulação funcional do design. O sistema usa rotinas previamente compiladas de IPs para fazer a visualização de imagens processadas. O usuário também pode escolher vídeos com imagens sequenciais para comparação dos resultados.

Compilação para FPGA: No processo de compilação, o sistema verifica se todos os blocos utilizados na concepção permitem o uso embarcado. Neste caso, para IPs de hardware, o sistema gera scripts de código-fonte em linguagem Bluespec (Bluespec, 2016). Para os IPs de software, o compilador cria scripts código fonte para NIOS II. Esses scripts têm um formato padrão para a comunicação com blocos internos de controle de execução.

Enviar a Hardware: Para uma geração completa de hardware, o usuário deve compilar o projeto gerado pela ferramenta no ambiente de desenvolvimento Bluespec. Caso necessário, outras simulações de hardware utilizando-se a ferramenta BlueSim podem ser executadas. O Bluespec gera arquivos em formato Verilog em um projeto padrão. Este projeto poderá ser sintetizado pelo usuário na ferramenta Quartus II (Altera, 2016) onde poderá ser enviada para placa FPGA. Para concluir a implementação, deve-se compilar o projeto gerado dentro da ferramenta NIOS II (NIOS II, 2016) e enviar o software criado para o processador embarcado no FPGA.

3.4. Construção de IP

Para estender a ferramenta com o novos IP que possam sem embarcados, é necessário o desenvolvimento de estruturas de código em um formato padrão. Para garantir que os resultados obtidos na simulação sejam iguais aos resultados práticos, são exigidos alguns cuidados.

Para a validação de IP de software (Figura 3), deve-se usar um padrão pré-estabelecido de construção. O arquivo principal "IP.c" contém o algoritmo proposto e deve ser compatível com as bibliotecas utilizadas pelo NIOS II. Recomenda-se executar o script de teste "TestBench.c", isto permite-lhe ler as entradas de dados de um arquivo de origem "iStream.bin" para ser usado no teste de algoritmo e salvar a saída para um arquivo de verificação "oStream.out ". Finalmente, com o sistema em execução no ambiente embarcado, um profiler pode ser feito para extrair várias características do algoritmo proposto.

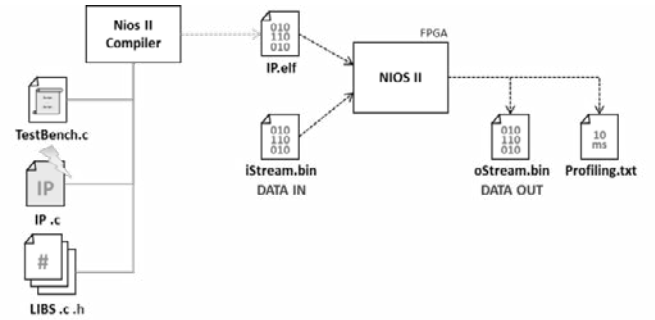

Figura 3. Validação de IP de Software

Para a validação de IP em hardware (Figura 4), o arquivo principal "IP.bsv" contém o algoritmo proposto em linguagem Bluespec. Caso necessário, a linguagem permite a reutilização de estruturas em Verilog usando um encapsulamento para essa finalidade. Uma vez compilado, gera-se um arquivo Verilog "IP.v" que pode ser adicionada a um template de validação no Quartus II. Através de técnicas de profiler, é possível extrair várias características do hardware gerado.

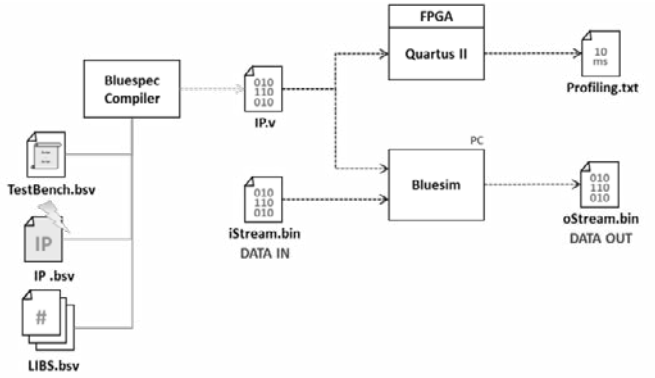

Figura 4. Validação de IP de Hardware

O BlueSim permite a validação do hardware gerado através de um programa escrito na própria linguagem "TestBench.bsv". Assim como os blocos validados em software, as entradas de dados podem ser lidas a partir de uma fonte de arquivo "iStream.bin" e a saída gerada em um arquivo de verificação "oStream.out". Em ambos

os casos, pode-se comparar os arquivos "fStream.bin" com a saída da implementação em software do PC.

4.Detecção de Pedestres

A proposta para a validação da ferramenta é a adaptação de um sistema de detecção de pedestres construído por Martinez (2011) (Figura 5) e baseado no trabalho de Jalalian (2008) e ao final, comparam-se os resultados finais com o trabalho original. Em seguida, modificar o sistema de estabilização de imagem para uma proposta de diferente autoria. No primeiro trabalho, o estabilizador é criado através de um vetor de valores pixels acumulados e o comparados com o mesmo vetor do quadro anterior. A segunda proposta consiste de uma implementação de fluxo óptico em hardware.

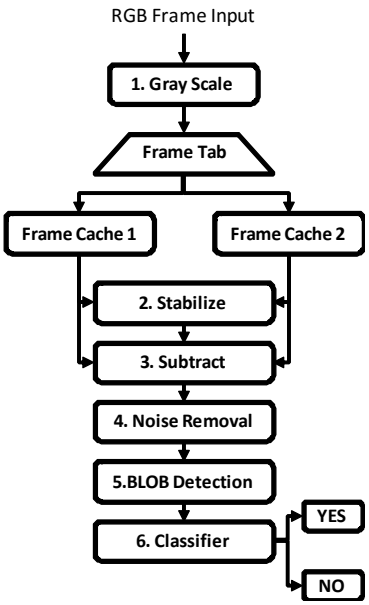

Figura 5. Modelo proposto para deteção de pedestres

4.1. Estabilização de Imagem com Vetor

Para comparar dois frames, as imagens devem estar alinhadas, dessa forma o IP de estabilização é essencial para o bom funcionamento do sistema. A detecção de objetos em movimento em um ambiente real é uma tarefa complexa pois, a imagem de fundo está mudando constantemente. Para isso, este bloco dimensiona o tamanho do vetor com a correspondente altura da imagem e armazena a soma dos valores de intensidade de pixel para cada linha (Figura 6). Para estabilizar, o vetor atual é comparado com o anterior trazendo como resultado o deslocamento do eixo Y. Utilizando-se apenas vetores, esta aplicação torna-se simples para ser construida em um sistema de hardware. Existem diversas abordagens para a estabilização de imagem, no entanto, muitas exigem grande quantidade de elementos lógicos para a sua construção.

Esta abordagem demonstrou ser eficiente, contudo foi somente implementada para identificar os deslocamentos verticais, de modo que o sistema não está preparado para detecções com o veículo em curva, permite apenas com o veículo movendo-se para frente.

Figura 6. Estabilização de Imagem com Vetor

4.2. Fluxo Optico

Uma outra solução proposta para resolver o problema do alinhamento entre os quadros é a criação de vectores de movimento da imagem. Assim, para minimizar o efeito de trepidação da câmera a imagem é dividida igualmente em quatro regiões e calcula-se os quatro vectores que representam o vector geral de movimento. A principal desvantagem desta abordagem é o custo computacional quando comparado à estabilização por vetor. A fim de avaliar o sistema de estabilização de imagem, foi utilizado como uma base, o trabalho de Lobo (2013), que foi baseado na arquitectura de MIT (2011), o qual é construído um sistema de fluxo óptico (Figura 7).

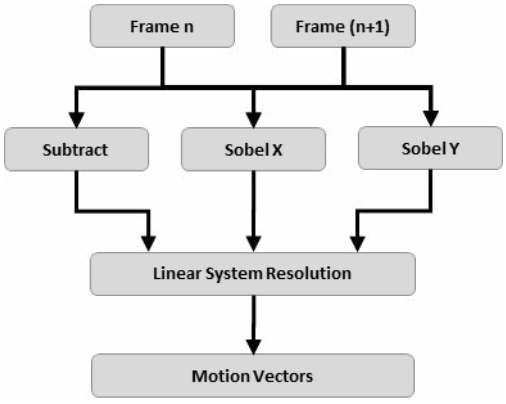

Figura 7. Fluxo Optico Proposto

Usando convolução com filtro de Sobel, derivadas espaciais são calculadas. A derivada temporal é o resultado da diferença entre as dois frames. Com estes dados, um sistema de equação é resolvido utilizando múltiplos produtos por pixel e o resultado é soma da vizinhança de pixels (3 x 3), gerando o campo de fluxo óptico.

4.3. Remoção de Ruidos

Após a diferenciação de quadros de imagem subsequentes (Figura 8 esquerda) é possível detectar objectos que se moviam no processo de transição, contudo, o movimento irregular da câmara pode gerar ruído quando existe diferenciação. Para resolver este problema, um passo de pré-processamento é necessário para minimizar regiões de ruído. Assim, vários algoritmos de redução de ruído podem ser utilizados,

por exemplo, o filtro de média não linear, o que suaviza a imagem sem diminuir a resolução.

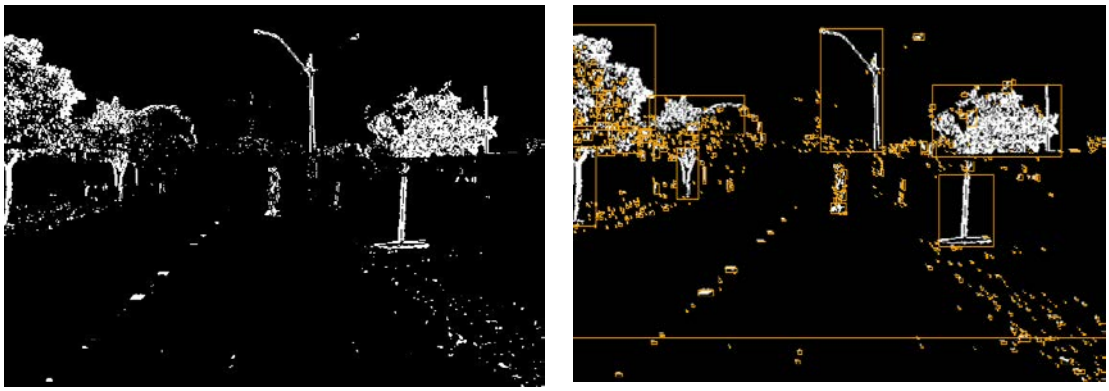

Figura 8. Resultado da diferença de dois frames (esquerda) e Componentes Conectados e Classificados (direita)

4.4. Identificação de componentes conectados e Classificador

O algoritmo de identificação de componentes conectados é usado para a detecção de objectos da cena. Este algoritmo é utilizado em visão computacional para detectar regiões conectadas em imagens digitais binárias.

Para classificação, adotou-se o uso de limitadores parametrizados dos objetos extraidos de componentes conectados. No projeto original, com o uso de uma placa de FPGA com recursos mais limitados, objetivou-se minimizar a utilização de recursos para atender ao requisito de execução em tempo real segundo a especificação de Gavrila (2006).

4.5. Resultados da Detecção

Usando a mesma fonte de dados de imagem e com os algoritmos implementados segundo as especificações do projeto origional, o resultado das detecções foram idênticas aos resultados do trabalho de base. Durante a síntese, foi necessário fazer alguns ajustamentos de temporização. Como esperado, o fluxo óptico para a estabilização de imagem tem boa vantagem, a Tabela 1 mostra a comparação entre estabilizadores de imagem:

Tabela 1. Detecção com diferentes estabilizadores

Estabilizador	Acertos	Falsos Positivos	Falsos Negativos
N/A	40%	20%	40%
Vetor	83%	9%	6%
Fluxo Optico	91%	2%	7%

A Figura 9 mostra um esboço do hardware gerado pelo framework.

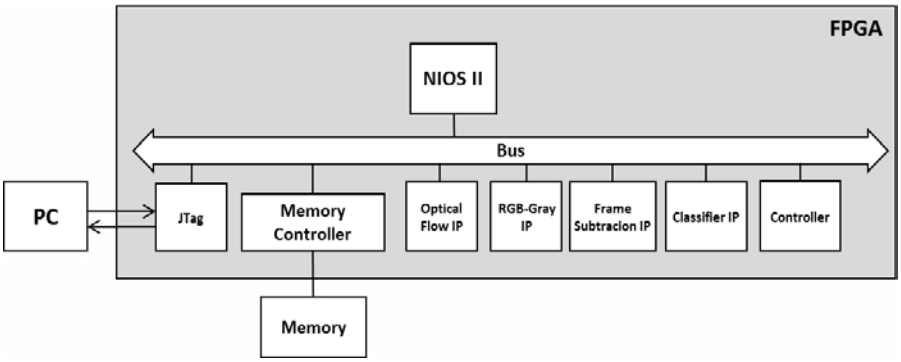

Figura 9. Esboço do hardware gerado

4.6. Plataforma de desenvolvimento do Hardware

A plataforma utilizada para a validação hardware gerado é a placa FPGA DE2i-150 da Terasic (Terasic, 2016). As principais especificações são: Dispositivo Cyclone IV EP4CGX150DF31; 149.760 Elementos Logicos; 720 M9K Blocos de Memória; 6.480 Kbits memória embarcada; 8 PLLs; 128MB (32Mx32bit) SDRAM; 4MB (1Mx32) SSRAM; 64MB (4Mx16) Flash modo 16-bit; Três osciladores de clock de 50MHz.

5. Conclusão

Este trabalho apresenta um framework para co-projeto de hardware e software para ADAS com visão computacional, sendo uma aplicação de código aberto baseada em componentes. As principais vantagens do aplicativo foram demonstradas através de um caso de uso para a detecção de pedestres. Apesar de ainda não ser uma ferramenta completa, o framework apresenta características promissoras através de validações de IPs e da execução de testes com diversos algoritmos. A partir dos dados apresentados na Tabela 1, nota-se que no esquema de detecção proposto, a estabilização de imagem usando o fluxo óptico é um fator crítico para melhores resultados. Observa-se também, que o vector para o algoritmo de estabilização, apesar da sua simplicidade, traz bons resultados em relação a resultados sem estabilização de imagem. Neste trabalho, utilizando-se de componentes baseados em outros projetos e com as mesmas amostras de fontes de dados foi possível obter os mesmos resultados originais, mostrando assim, que é possível migrar projetos externos para a ferramenta sem criar interferência nos resultados e permitindo a exploração do espaço de projetos.

Referências

Altera (2016). Introduction to the Quartus II Software. web site. [Online] Acessado em abril 2016. http://www.altera.com/literature/manual/intro_to_quartus2.pdf

Baselabs (2016). Baselabs Flexible ADAS Development web site. [Online] Acessado em abril 2016. http://baselabs.de/flexible-adas-development-kopie.html

Bengler, K., Dietmayer, K., Färber, B., Maurer, M., Stiller, C., Winner, H. (2014). "Three Decades of Driver Assistance Systems: Review and Future Perspectives," IEEE Intell. Transp. Syst. Mag., vol. 6, no. 4, pp. 6 – 22

Blender (2016). Website. [Online] Acessado em abril 2016. https://www.blender.org/

Bluespec (2016). Bluespec SystemVerilog User Guide, [Online]: http://wiki.bluespec.com/Home/BSV-Documentation

Boeing, A.; Braunl, T. (2008) "ImprovCV: Open component based automotive vision," Intelligent Vehicles Symposium, 2008 IEEE , vol., no., p.297,302,4-6.

Comemso (2016). Comemso web site. [Online] Acessado em abril 2016. http://www.comemso.com/index.php/software

Gavrila D., Munder S. (2006). "Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle" International Journal of Computer Vision Springer Science.

Intempora (2016). Intempora web site. [Online] Acessado em abril 2016. http://www.intempora.com

ITRS (2011), International Technology Roadmap for semiconductors. [Online] Acessado em abril 2016. http://www.itrs.net/Links/2011ITRS/2011Chapters/2011SysDrivers.pdf

Jalalian, A., Fathy, M. (2008). Pedestrian Detection from a Moving Camera with an Advanced Camera-Motion Estimator. Third International IEEE Conference on Signal-Image Technologies and Internet-Based System.

Kisačanin, B., Gelautz, M. (2014). Advances in Embedded Computer Vision, Springer.

LabVIEW (2016). LabVIEW web site. [Online] Acessado em abril, 2016. http://www.ni.com/labview

Lobo, T. M. (2009). Co-projeto hardware/software para cálculo de fluxo ótico. [Online] Acessado em abril 2016. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-28082013-094816/en.php

Martinez, L. A. (2011). Projeto de um sistema embarcado de predição de colisão a pedestres baseado em computação reconfigurável. [Online] Acessado em abril 2016. http://www.teses.usp.br/teses/disponiveis/55/55134/tde-27022012-110356/pt-br.php

MIT (2011). MIT 6.375 web site. [Online] Acessado em abril 2016, http://csg.csail.mit.edu/6.375/6_375_2011_www/projects.html.

NIOS II (2016). Altera NIOS II Hardware Tutorial web site. [Online] Acessado em abril 2016. http://www.altera.com/literature/tt/tt_nios2_hardware_tutorial.pdf

MATLAB (2016). MATLAB HDL Coder web site. [Online] Acessado em abril 2016. http://www.mathworks.com/fpga-design/hardware-software-codesign.html

MontiVision (2016). MontiVision Development Kit (SDK) web site. [Online] Acessado em abril 2016. http://www.montivision.com

OpenCV (2016). web site. [Online] Acessado em abril 2016. http://www.opencv.org

Stratix V (2013). Stratix V Development Board Reference Manual. [Online] Acessado em abril. http://www.altera.com/literature/manual/rm_svgx_fpga_dev_board.pdf

Terasic (2016). Terasic DE2i-150 FGA Development Kit. web site. [Online] Acessado em abril 2016. http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=11&No=529&PartNo=1