Hardwares e sistemas multiagente: um estudo sobre arquiteturas híbridas

Rafhael Rodrigues Cunha [1]
Renata Wotter [1]
Rodrigo Rabenhorst [1]

Resumo: Este artigo apresenta três estudos de caso sobre a aplicabilidade de arquiteturas de hardware reconfiguráveis, como FPGA, voltadas à utilização em sistemas multiagentes. Feita uma análise visando à elucidação dos resultados e das contribuições que os estudos proporcionaram aos autores, observa-se que o desenvolvimento de sistemas inteligentes depende cada vez mais de uma programação que explore o hardware ao máximo. Esse desfecho torna o uso de hardwares reconfiguráveis o mais aconselhável quando problemas computacionais complexos demandam respostas rápidas e eficientes, como nos casos estudados.

***Palavras-chave*:** Estudo de caso. Hardwares reconfiguráveis. Sistemas multiagente.

Abstract: *This paper presents three case studies on the applicability of reconfigurable hardware architectures, such as FPGA, aimed at use in multi-agent systems. Made an analysis in order to elucidate the results and contributions that the studies provided to the authors, it is observed that the development of intelligent systems increasingly rely on a program that exploits the hardware to the fullest. This outcome makes using the most advisable reconfigurable hardware when complex computational problems require quick and efficient responses, as in the cases studied.*

***Keywords*:** *Case study. Multi-agent systems. Reconfigurable hardware.*

1 Introdução

Inicialmente, os agentes foram desenvolvidos com a finalidade de dar dinamismo a aplicações de software. Entretanto, de alguns anos para cá, começaram a ser elaborados estudos que tinham como proposta acrescentar agentes em hardwares capazes de ser reconfigurados conforme a necessidade da aplicação [1].

No mercado atual, os hardwares reconfiguráveis mais usuais são o *Field Programmable Gate Array* (FPGA) e o *Compute Unified Device Architecture* (CUDA). O FPGA é um dispositivo semicondutor utilizado para processamento de informações digitais. A CUDA é uma plataforma de computação paralela utilizada para tirar proveito das unidades de processamento gráficos (GPUs) de placas da NVIDA, objetivando a resolução de problemas computacionais complexos que demorariam maior tempo para serem resolvidos em um processador convencional [1].

Visando ao dinamismo que os sistemas multiagentes disponibilizam aos usuários e ao desempenho que as placas FPGA proporcionam, apresenta-se um estudo de caso sobre alguns trabalhos que mesclam a utilização dessa tecnologia. Para tanto, o artigo está organizado da seguinte forma: o capítulo 2 introduz os conceitos fundamentais para o entendimento do tema. No capítulo 3, são apresentados os trabalhos estudados e as considerações sobre cada um deles. Finalmente, o capítulo 4 expõe as conclusões acerca deste trabalho.

2 Fundamentação teórica

Nesta seção, são apresentados os conceitos teóricos fundamentais para o entendimento deste trabalho, tais como: agentes, arquitetura BDI, sistemas multiagentes (SMA) e hardware programável (FPGA).

[1]Programa de Pós-Graduação em Engenharia de Computação (PPGCOMP), FURG, *Campus* Carreiro - Av. Itália km 8 - Rio Grande (RS) - Brasil.
{rcrafhaelrc,renata.wotter,rodrigoraben@gmail.com}

http://dx.doi.org/10.5335/rbca.2015.4617

2.1 Agentes

Dado um determinado sistema, denomina-se "agente" cada uma de suas entidades ativas. O conjunto de agentes forma uma "sociedade". As entidades passivas serão designadas pelo termo "ambiente". Um agente raciocina sobre o ambiente, sobre os outros agentes e decide, racionalmente, quais objetivos deve perseguir, quais ações devem tomar, entre outras funções. A utilização do termo "agente" pressupõe uma noção subjacente de controle. De fato, o termo "ativo" tem como objetivo sinalizar que o conceito de agente não corresponde a noções estáticas como módulos, conjunto de regras e bases de conhecimento, sem que a estas esteja intimamente relacionado um mecanismo de controle para a sua ativação [2].

Em [2], segundo [3], um agente é uma entidade real ou virtual, imersa num ambiente sobre o qual é capaz de agir, que dispõe de uma capacidade de percepção e de representação parcial desse ambiente, que pode se comunicar com outros agentes e que apresenta um comportamento autônomo, consequência de suas observações, de seu conhecimento e das suas interações com os outros agentes.

De acordo com [4], um agente inteligente apresenta as seguintes propriedades:

- reatividade - responde às mudanças no ambiente em um tempo adequado;
- pró-atividade - toma a iniciativa de tentar atingir seus objetivos;
- habilidade social - interage com outros agentes para satisfazer seus objetivos.

Em [5], foi proposta uma taxonomia na qual um agente é visto como um tipo especial de sistema computacional e que pode ser classificado segundo alguns eixos, tais como:

- eixo de atuação - atuar de forma isolada ou interagindo com outros agentes;
- eixo cognitivo - modelo de representação interna do ambiente e dos outros agentes baseados em estados mentais e em um modelo racional de decisão, ou apenas, em reações provocadas pelo ambiente;
- eixo ambiental - atuar no desktop ou em rede.

2.2 Arquitetura BDI

As mais importantes arquiteturas de agentes deliberativos são baseadas em um modelo de cognição fundamentado em três principais atitudes mentais, que são as crenças, os desejos e as intenções (abreviadas por BDI: beliefs, desires e intentions).

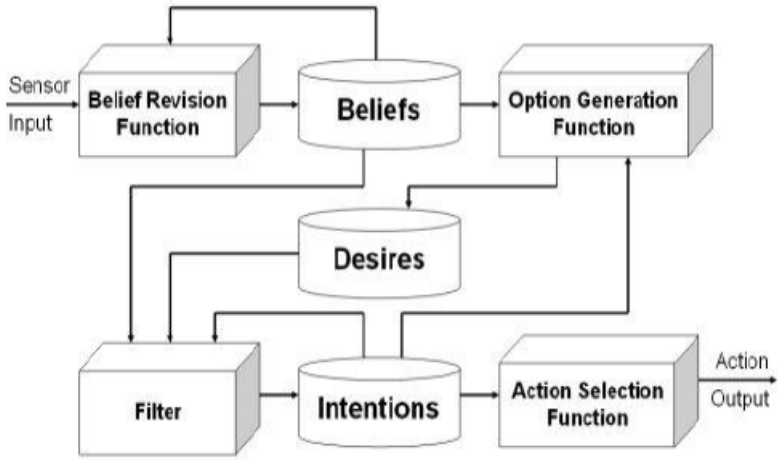
A arquitetura BDI surgiu na década de 1980, no Stanford Research Institute, e é baseada na Teoria do raciocínio prático do filósofo Michael Bratman [6]. Essa arquitetura está representada na Figura 1.

Resumidamente, essa arquitetura de agente está estruturada da seguinte forma: as crenças representam aquilo que o agente sabe sobre o estado do ambiente e dos agentes naquele ambiente (inclusive sobre si mesmo); os desejos representam estados do mundo que o agente quer atingir (dito de outra forma, são representações daquilo que ele quer que passe a ser verdadeiro no ambiente). Em tese, desejos podem ser contraditórios, ou seja, pode-se desejar coisas que são mutuamente exclusivas do ponto de vista de ação prática. Normalmente, são referidos como um subconjunto dos desejos que são todos compatíveis entre si. As intenções representam sequências de ações específicas que um agente se compromete a executar para atingir determinados objetivos.

2.3 Sistemas multiagente

Tomando um ponto de vista bottom-up no desenvolvimento de sistemas computacionais, o objetivo da área de SMA passa a ser a definição de modelos genéricos de agentes, interações e organizações que possam ser instanciados dinamicamente, dado um problema. Definido esse ideal de metodologia de desenvolvimento de sistema, essa abordagem apresenta as seguintes características [2]:

Figura 1: Arquitetura BDI Genérica [4].



- os agentes são concebidos independentemente de um problema particular;
- a interação entre os agentes não é projetada anteriormente, busca-se definir protocolos que possam ser utilizados em situações genéricas;
- a decomposição de tarefas para solucionar um dado problema pode ser feita pelos próprios agentes;
- não existe um controle centralizado da resolução do problema.

2.4 Hardware reconfigurável

Um FPGA é um dispositivo semicondutor largamente utilizado para o processamento de informações digitais. Foi criado pela Xilinx Inc. e teve o seu lançamento no ano de 1985 como um dispositivo que poderia ser programado de acordo com as aplicações do usuário (programador). O FPGA é composto, basicamente, por três tipos de componentes: blocos de entrada e saída (IOB), blocos lógicos configuráveis (CLB) e chaves de interconexão (*switch matrix*). Os blocos lógicos são dispostos de forma bidimensional, e as chaves de interconexão, em formas de trilhas verticais e horizontais entre as linhas e as colunas dos blocos lógicos [7].

3 Estudo de caso

O estudo de caso visa fazer um apanhado dos trabalhos estudados, demonstrando de maneira geral o processo de desenvolvimento das arquiteturas, bem como os conceitos aplicados nos seus desenvolvimentos e os experimentos resultantes dos testes realizados nessas.

3.1 Incorporando sistemas multiagente em um combinado de sistema de hardware/software reconfigurável - uma estrutura geral de arquitetura

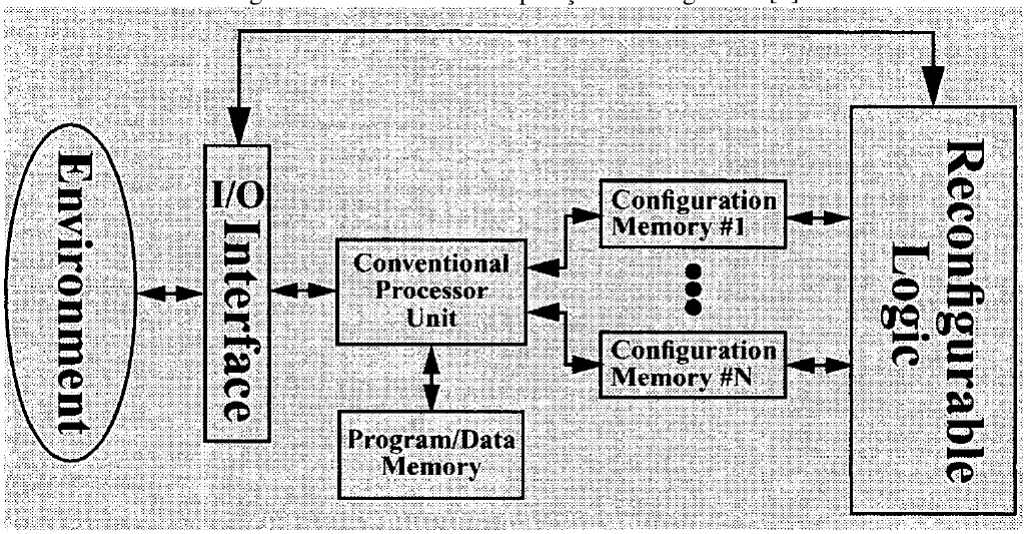
A maioria das pesquisas na área de SMA tem foco nas linhas de sistemas embarcados distribuídos ou em ambientes da internet [1]. Porém, de alguns anos para cá, com o advento da grande escala disponível no mercado de lógica reprogramável, é possível estender esse paradigma para um ambiente mais geral, em que os agentes possam residir tanto no software como no hardware do sistema.

3.1.1 Ambiente de computação genérico reconfigurável proposto

A Figura 2 ilustra um ambiente reconfigurável híbrido genérico de computador que apoia o modelo de hardware e agentes de software integrado apresentados nesse trabalho. Um exemplo desse ambiente pode ser

criado através da incorporação de um protótipo de placa baseada em FPGA com uma placa-filha de processamento de um computador pessoal [8].

Figura 2: Ambiente de computação reconfigurável [1].



Segundo [1], o ambiente inclui um elemento de processamento popular juntamente com seu programa de associados e da memória de dados. Os elementos lógicos reconfiguráveis servem de interface para o elemento de processamento por meio de memória mapeada I/O. Essa memória mapeada E/S permite que o processador escreva diretamente na memória de configuração da lógica reconfigurável, dando ao processador o poder de mudar dinamicamente a funcionalidade do hardware mediante a introdução de novos agentes. A lógica reconfigurável é usada para suportar reconfiguração parcial, na medida em que as partes da sua lógica podem ser alteradas sem afetar o restante. A interação com o ambiente externo é suportada por conexões I/O dos sensores e atuadores que estão interligados diretamente com os periféricos do processador e dos elementos da própria lógica reconfigurável, permitindo, assim, que as operações de sensores e atuadores de alta velocidade sejam controladas diretamente pela lógica reconfigurável sem passar pelo processador.

Nota-se que a quantidade de memória do programa é considerada muito maior em comparação com a quantidade de memória de configuração. Isso significa, segundo [1], que a porção lógica reconfigurável é vista como a parte mais cara de concepção do sistema híbrido de agentes. Em geral, os agentes simples e mais bem estruturados que abrangem operações altamente repetitivas são bons candidatos para as operações de hardware reconfiguráveis, enquanto que os agentes mais complexos e irregularmente estruturados devem ser colocados em software (na memória do programa do processador). Agentes de hardware servem para atuar como uma interface entre o meio ambiente e o sistema reconfigurável. Esses agentes são colocados na memória de configuração do sistema reconfigurável. Em ambos os casos de hardware e software, vários agentes podem estar presentes ao mesmo tempo. Para alguns agentes, é possível ocorrer a migração entre o hardware e o software, a fim de encontrar um equilíbrio entre desempenho e melhor uso dos recursos limitados associados à implementação particular.

3.1.2 Exemplo de uso simples

Nesse modelo, o problema é dividido em subproblemas, sendo cada uma de suas partes resolvida por um agente. Desse modo, cada agente pode resolver sua respectiva porção do problema da forma que melhor atenda à sua própria agenda.

Foi considerado o problema de escolher, de um conjunto de nove números de 1 a 9, três números cuja soma acrescenta-se ao número 15 [1]. Com um cálculo simples, descobre-se que existem 84 combinações diferentes que possibilitam selecionar três números de nove números e que apenas oito combinações diferentes terão a soma de 15.

Foram utilizados quatro agentes para resolver esse problema. As crenças de cada agente são o conjunto de números inteiros de 1 a 9. O plano é uma função que extrai três números com tal propriedade, e o objetivo é que a soma seja igual ao número 15. Assume-se como premissa que os agentes recebem o índice do primeiro número do ambiente ou de outro agente. Cada agente tem uma memória local para armazenar valores temporários e resultados e uma camada de comunicação para enviar e receber informação para/do ambiente ou de outros agentes, que proporciona a capacidade de comunicação e de cooperação para cada agente. Estes agentes também podem ser implementados em software ou hardware, dependendo da necessidade de uma operação de alta velocidade. Em [1], existe um exemplo de implementação em VHDL.

3.2 Sistema de arquitetura aberta baseada na plataforma multiagente para maquinas CNC que possuem hardware/software reconfiguráveis

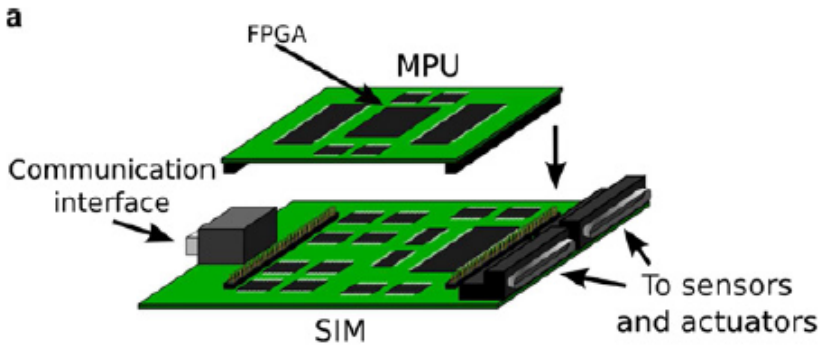
O trabalho [9] propõe uma arquitetura aberta para um sistema com uma plataforma de hardware e software baseado na filosofia multiagente distribuída. Segundo [9]. Essa arquitetura de sistema aberto, conhecida como controlador multiagente distribuído (Madcon), destina-se a preencher os requisitos de reconfigurabilidade para a próxima geração de máquinas inteligentes em relação ao desenvolvimento de drivers inteligentes.

O sistema proposto integra componentes de hardware e software que compartilham um protocolo de comunicação. Esse sistema é projetado para permitir a integração de novas unidades de processamento de ambos os módulos de hardware e software. Madcon abrange três elementos: a plataforma de hardware, de software e um protocolo de conexão.

3.2.1 Plataforma de hardware

A plataforma de hardware integra os controladores distribuídos reconfiguráveis (DRC), que contêm dois elementos: a unidade de microprocessamento (MPU) e o sistema de interconexão dos meios de comunicação (SIM), conforme visto na Figura 3. A interface com o mundo real é o SIM, o qual mantém a conversão de tensão para analógico e IO digital. Ele se concentra em condicionamento de sinal para controlar equipamentos externos e protocolos de comunicação.

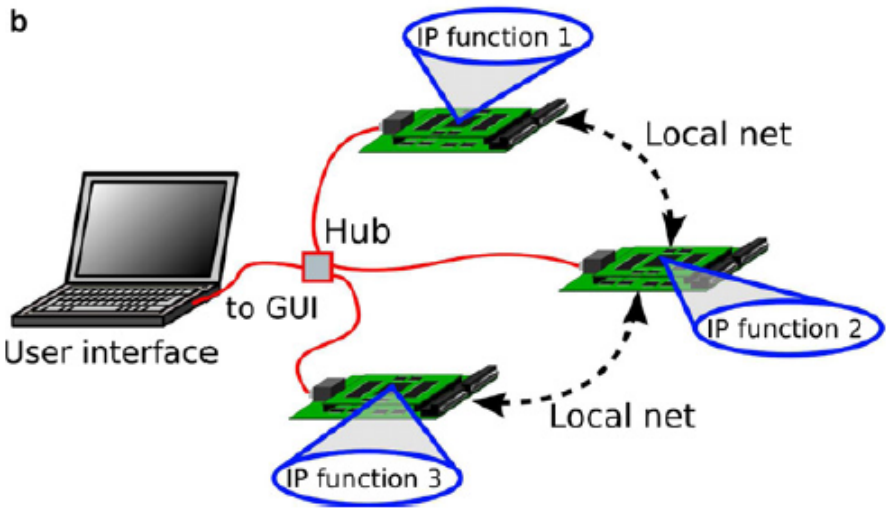
Figura 3: Controlador Reconfigurável Distribuído (DRC), mostrando seus elementos MPU e SIM [9]



O MPU não se restringe a ser um FPGA. Entretanto, um FPGA baseado em DRC possibilita que novas funções em hardware possam ser incluídas em um DRC existente em uma máquina sem afetar o desempenho do sistema. Cada DRC pode ser conectado à interface de usuário, e uma rede DRC local opcional pode ser usada para comunicá-los de forma direta, conforme Figura 4.

No sistema Madcon, um processador integrado específico também foi desenvolvido. Esse é um processador especializado para a transferência de ficheiros de chamada πRTP, que se trata de um pequeno processador com algumas instruções básicas e potência mínima de computação, mas com recursos avançados para registrar transferências e interrupções [9]. Essa plataforma visa extrair o máximo de proveito do processamento paralelo, deixando as tarefas de comunicação para o processador. A abordagem SoC é atingida pela incorporação de todas as unidades dentro do FPGA.

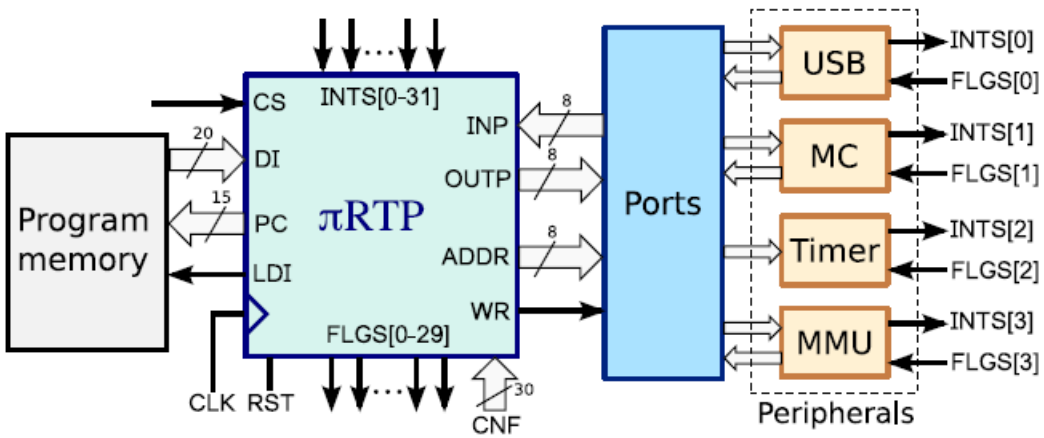
Figura 4: Interface de usuário com as redes locais DRC opcionais [9]



Segundo [9], o πRTP é um pequeno microprocessador RISC em uma arquitetura Harvard, com apenas dezoito instruções. O conjunto de instruções está disponível na íntegra no artigo citado.

A Figura 5 mostra uma aplicação simples, na qual o processador desenvolvido controla a troca de informações entre os periféricos. A principal característica de um aplicativo baseado em πRTP é que todos os periféricos estão ativos ao mesmo tempo, enquanto o processador coordena a interação de dados entre eles, sem afetar o seu desempenho. Uma vez que os cálculos são realizados pelas unidades periféricas, o poder computacional necessário do πRTP é mínimo.

Figura 5: Exemplo de aplicação para o πRTP onde os periféricos são executados em paralelo enquanto o processador lida com as transferências de dados entre eles [9].

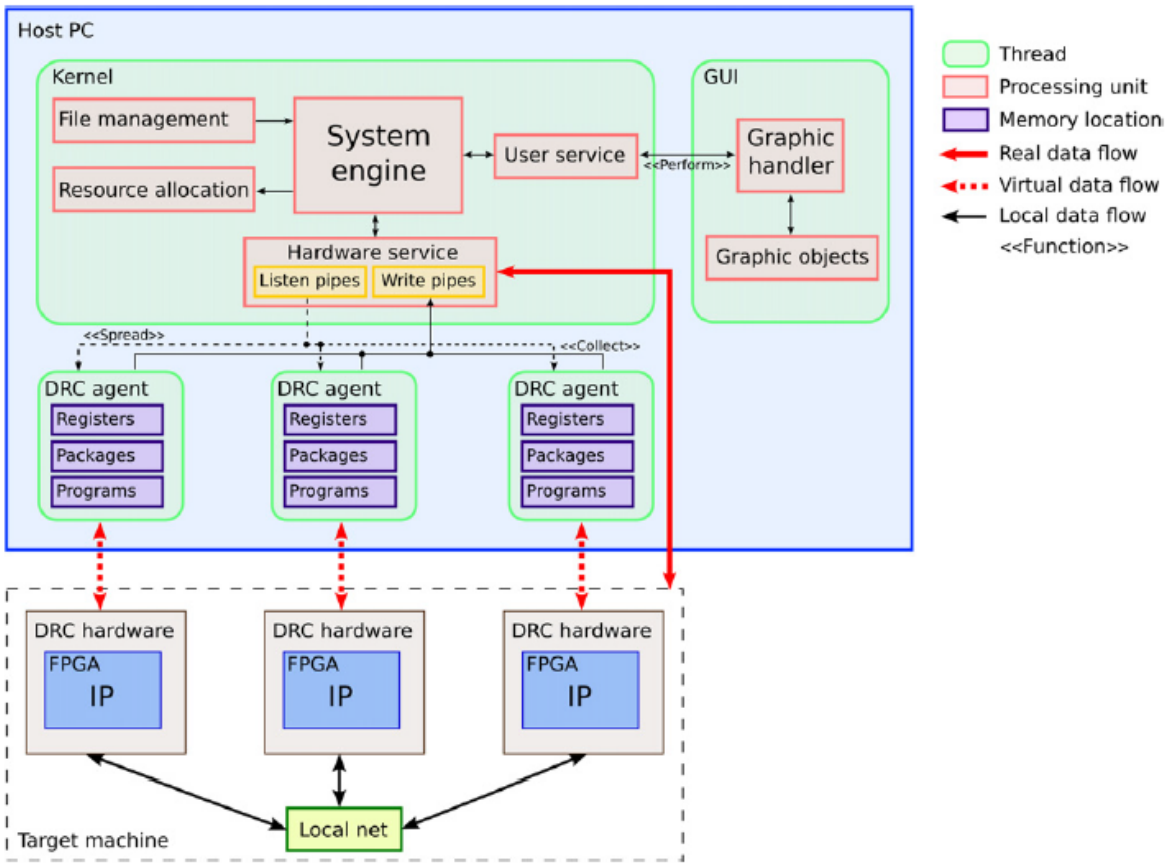


3.2.2 Plataforma de software Madcon

A arquitetura do software é exibida na Figura 6, onde se ilustra um diagrama de blocos que contém o Host PC e a máquina CNC com seus componentes correspondentes. O Host PC mantém a execução do software e a GUI, ao passo que a máquina CNC detém os módulos da DRC em tempo real e os mecanismos associados controlados pelos DRCS. Os elementos de software do sistema são agrupados em três módulos: kernel, agente DRC e GUI. O

núcleo contém as principais funções do sistema. Os agentes de DRC são elementos de software que representam as unidades de tempo real da DRC. Cada agente DRC contém registros, pacotes e programas exigidos pelas unidades da DRC e é executado em uma thread separada. O núcleo e os agentes de DRC são necessários para a execução do sistema.

Figura 6: Arquitetura do sistema Madcon [9].



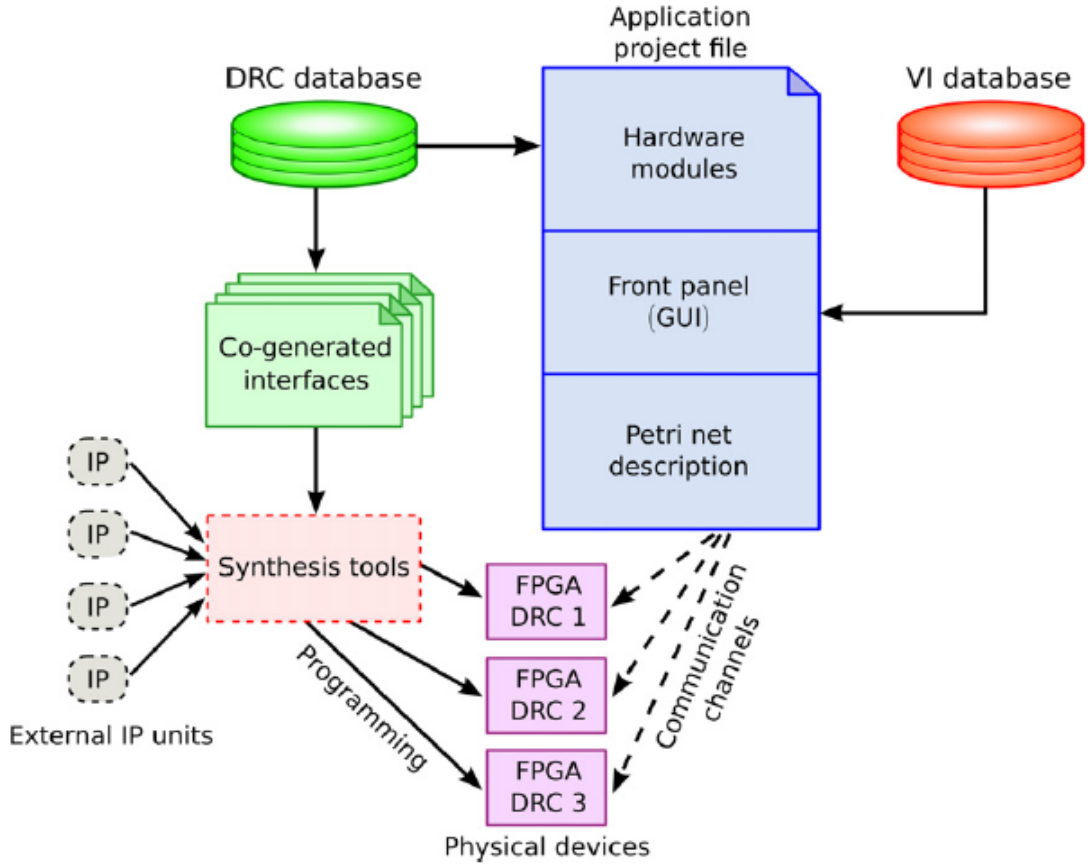
O sistema de gerenciamento de arquivos MADCON, apresentado na Figura 7, incorpora um arquivo de projeto, dois bancos de dados, arquivos DRC cogerados com VHDL e arquivos do módulo de IP externo. O banco de dados DRC contém informações sobre os módulos de hardware que podem ser incluídos em um aplicativo, e o banco de dados VI contém informações sobre a instrumentação virtual disponível para fins de exibição gráfica.

Para os módulos de DRC baseados no sistema embarcado πRTP, uma unidade VHDL de nível superior é gerada pelas ferramentas de sistema para facilitar a comunicação entre os módulos de hardware e da aplicação. Uma vez que os dispositivos físicos são programados, podem comunicar-se com o sistema por meio da interface cogerada sem a necessidade de desenvolvimento de um controlador de porta especial para cada módulo de hardware [9].

O requerimento para a definição do projeto abrange todas as informações do sistema em um único arquivo, que é usado para identificar o software e o hardware conectados ao host. Esse arquivo inclui a definição do painel frontal que contém a instrumentação virtual. Além disso, o projeto tem uma rotina de execução expressa como uma rede de Petri para fazer um ciclo automático para todo o processo industrial.

O Host PC abrange dois níveis de execução do sistema. A execução de baixo nível, que corresponde à sequência de pacotes executados pelo sistema de software nos quais afetam apenas um DRC que está diretamente relacionado a um fluxograma. O fluxograma está expresso como uma linguagem de descrição de fluxograma do roteiro (FDL). Já a execução de alto nível corresponde ao controle de um processo inteiro. Esse controle é realizado

Figura 7: Arquitetura de arquivos do sistema Madcon [9].



pela rede de Petri, que comanda toda a sequência de operações. A rede de Petri está contida no arquivo do projeto e é gerada usando uma ferramenta gráfica [9].

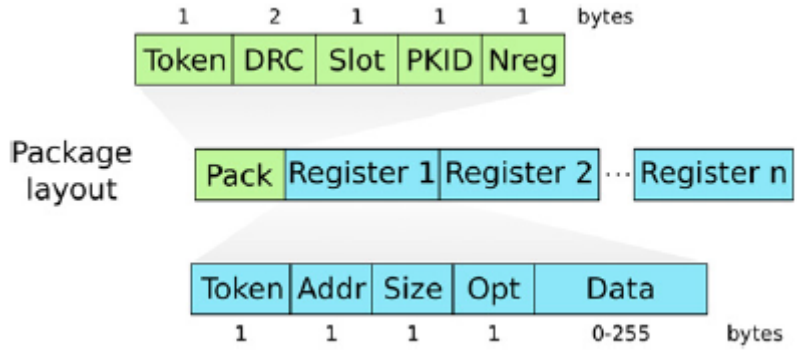
3.2.3 Protocolo de conexão Madcon

A Figura 8 demonstra a disposição do *package layout*. Ele incorpora a informação do pacote *pack* no início do fluxo seguido da informação de cada um dos registros incluídos no pacote [9]. O *package layout* contém cinco campos: um *byte* para o *token* que identifica a seção *pack*, dois *bytes* para o número de identificação do DRC, um *byte* para o número do slot, um *byte* para o número de identificação do pacote e um *byte* para o número de registros incluído no pacote. A seção de registro contém cinco campos: um *byte* para o *token* de registo, um *byte* para o endereço base do registro, um *byte* para o tamanho do registro em *bytes*, um *byte* para as opções de registro e um campo de tamanho variável para dados. A quantidade de dados varia de 0 a 255 *bytes* e deve ser consistente com o campo de tamanho.

3.2.4 Estudo de caso

Para efeitos de teste, [9] desenvolveu um controlador e monitor em um torno mecânico CNC. Imagens do experimento podem ser visualizadas no artigo estudado. O incremento integra um controlador de movimento, um PLC para controlar os eventos discretos (bombas, conversor de fuso e interruptores de segurança) e um monitor atual de vibração para realizar uma vigilância contínua da condição da ferramenta e do nível de vibração. O hardware foi implementado em um lowcost Spartan3E-1600 FPGA da Xilinx.

Figura 8: Apresentação do package layout para trocar informações na plataforma de hardware e software [8].



3.2.5 Resultados

A plataforma desenvolvida foi utilizada para implementar uma aplicação CNC integrada por três unidades de DRC. Esse experimento reúne recursos de monitoramento de controle em uma abordagem IMS usada no sistema de Madcon. Em virtude de os recursos de processamento das unidades da DRC dependerem dos núcleos IP, funções complexas em tempo real como controlador de alta velocidade, a dinâmica do movimento de controle, o monitoramento sem sensor foram incluídas no teste.

Em [9], é explicado que o desenvolvimento dos núcleos IP consome mais tempo se comparado ao projeto de criação de uma rotina de software para um DSP ou microcontrolador. Entretanto, os benefícios no desempenho do sistema são superiores segundo o autor. Como conclusão, ele menciona que o sistema criado não é um IMS, mas que o trabalho demostra o grau de integração entre hardware e software por meio de um cartão.

3.3 Uma arquitetura de sistema ligado em chip baseada em agentes para um sistema de tempo real

Em [10], é proposta uma arquitetura reconfigurável *multi-core system-on-chip* (RMCSoC). Também são propostas uma metodologia codesign de hardware/software baseada em agentes unificados para otimizar o desempenho do sistema neste RMCSoC e uma plataforma autorreconfigurável para permitir que o FPGA reconfigure o hardware dinâmico em tempo de execução por um processador embutido.

Segundo [10], convencionalmente, todas as tarefas de um sistema complexo de tempo real são implementadas em um processador geral em software devido à sua grande flexibilidade. Entretanto, seguidamente, os sensores que compõem esses sistemas devem funcionar em paralelo e de forma independente, o que, seria, no mínimo, desafiador para um engenheiro de software conseguir implementar. Por essa questão, surgem alternativas, como, por exemplo, hardware específico.

Hardware paralelo não é tão afetado por questões como troca de tarefas, agendamento, serviço de interrupção, e as seções críticas, que complicam as soluções de software de tempo real [10]. Com isso, surgem novas tecnologias, como o FPGA, o qual permite que um arquiteto possa desenvolver uma única plataforma reconfigurável para instanciar ambos os processadores e as unidades lógicas necessárias, viabilizando a arquitetura reconfigurável *multi-core system-on-chip* (RMCSoC).

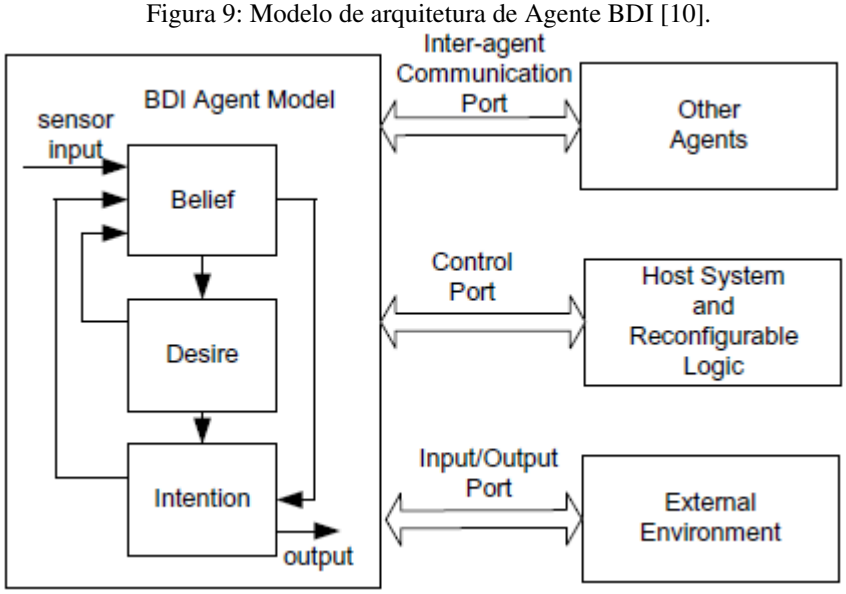
Para acelerar a integração de sistemas e o processo codesign de hardware/software em uma plataforma RMCSoC, uma interface de hardware/software unificado é necessária. Este trabalho [10] apresenta uma metodologia baseada em codesign de agentes executados em uma plataforma RMCSoC, onde múltiplos processadores soft-core são instanciados em um FPGA. O sistema é decomposto em vários agentes com base nas especificações do sistema, onde estes agentes podem realizar algumas tarefas específicas de forma independente e podem se comunicar uns com os outros. Estes agentes são separados em agentes de software e de hardware. Uma vez que todos os agentes partilham o mesmo mecanismo, é possível proporcionar um mecanismo de comunicação unificado entre as peças de hardware e de software. Um novo protocolo de passagem de mensagens sob demanda (ODMP) de

comunicação é também proposto.

3.3.1 Arquitetura de agente BDI

A arquitetura de agentes BDI, explicada na seção 2.2 deste artigo, sofreu algumas modificações para apoiar o desenvolvimento deste trabalho. A Figura 9 exibe três portas externas: uma de comunicação interagente, outra de controle e a última de entrada/saída. Segundo [10], a porta de controle tem três funções: (a) para o sistema ativar/desativar os agentes; (b) para o agente informar ao sistema quando termina o seu trabalho; (c) para o agente sincronizar com o sistema. A porta de entrada/saída é usada para enviar e receber informações a partir do meio de acolhimento. A porta de comunicação interagente permite que os agentes enviem/recebam informações de outros agentes e também cooperem com eles. Uma vez que todos os agentes trabalhem em um modo assíncrono, as mensagens somente serão emitidas na demanda, fato que leva ao desenvolvimento de um novo protocolo de comunicação de mensagens sob demanda (ODMP).

Figura 9: Modelo de arquitetura de Agente BDI [10].

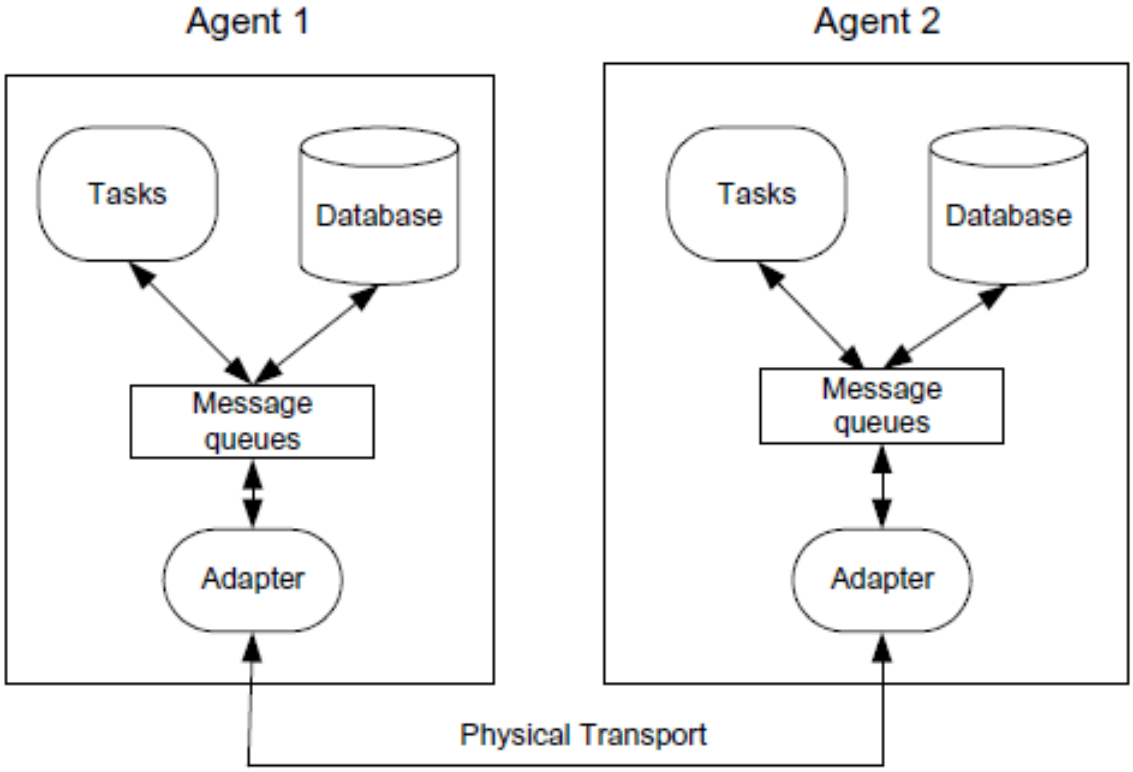


A Figura 10 exibe um exemplo simples de comunicação entre dois agentes. Nesse mecanismo, cada agente cria uma fila de mensagens, em uma ordem FIFO, na qual as mensagens marcadas como “urgentes” estão no começo da fila, seguidas das marcadas como “normais”, estabelecendo-se, assim, prioridades. Cada agente apresenta seu próprio banco de dados, favorecendo a troca de mensagens, visto que, antes de ocorrer a comunicação entre os agentes, o agente emissor consulta o banco de dados de seu receptor para verificar se ele está disponível para receber a mensagem. A fim de proporcionar um mecanismo de comunicação entre os agentes independente de meios de comunicação, os adaptadores são necessários, pois estes proporcionam uma interface uniforme de comunicação, seja qual for o transporte particular utilizado [5].

3.3.2 A autoconfiguração da plataforma

[10] escolheu o FPGA Vertex Pro II como plataforma do RMCSoc. [9] citou dois trabalhos que utilizam uma quantidade de memória limitada em um chip para armazenar os dados de reconfiguração, entretanto, essa proposta não é adequada para um sistema de tempo real. Por essa razão, neste trabalho, foi proposta a utilização de memórias externas DDR para corrigir esse problema. A Figura 11 exibe a plataforma proposta. A lógica fixa inclui circuitos lógicos fixos comuns, ICAP, o controlador de configuração, vários blocos de RAMs de porta dupla (BRAMs), processadores embarcados e memória externa. O processador embutido no FPGA oferece um controle inteligente da reconfiguração do dispositivo em tempo de execução. O processador incorporado comunica com a lógica FPGA por meio do barramento CoreConnect. Por fim, a memória externa DDR é ligada ao processador pelo processo de omnibus local (barramento PLB).

Figura 10: Estrutura de comunicação multiagente [10].



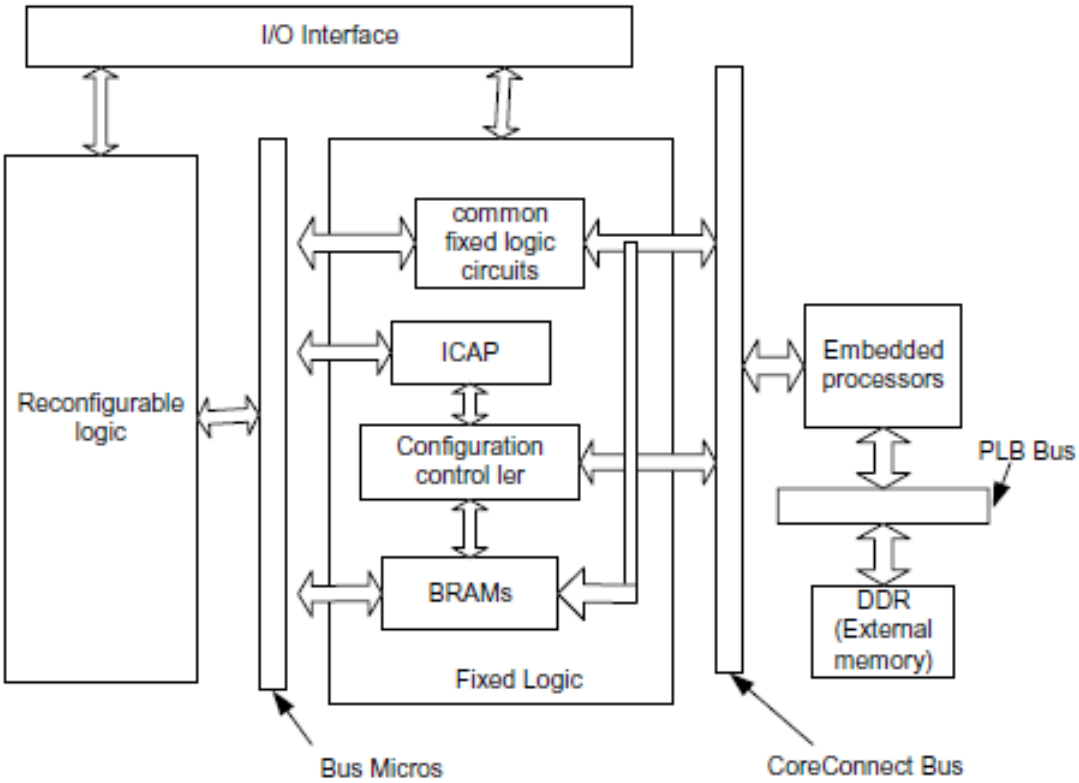
O controlador de configuração utiliza o ICAP para reconfigurar as partes lógicas reconfiguráveis. As características de porta dupla permitem que a BRAM seja acessada de diferentes domínios de relógio de cada lado. Uma porta é acessada pelo controlador de configuração, enquanto a outra porta é acessada pela parte lógica de reconfiguração. Os circuitos lógicos fixos comuns podem implementar alguns agentes específicos de hardware, bem como a lógica de geração de interrupção. Em virtude de o ambiente ser dinâmico em sistemas de tempo real, as partes lógicas reconfiguráveis necessitam ser reconfiguradas, em resposta ao novo ambiente. Há duas condições de disparos para essa reconfiguração. Uma é chamada bottom-up, e outra top-down. A primeira a se manifestar por um evento invocado pelos agentes do sensor. No caso da segunda, o próprio processador toma a decisão, devido à informação de entrada do agente gerenciador do sensor demostrar uma variação dramática em comparação com a informação de dados anteriormente coletada.

3.3.3 Codesign Hardware/Software

Duas questões foram abordadas na seção de codesign: a primeira é o particionamento de hardware/software; a segunda, a implementação da arquitetura baseada em agentes de hardware/software. Sobre a questão de particionamento, os autores adotaram uma heurística segundo a qual os agentes que apresentam operações demoradas altamente repetitivas e extensas são adequados para implementação em hardware reconfigurável com mais regularidade estruturada, enquanto que os agentes mais complexos e estruturados de forma irregular devem ser programados em software. Outro critério adotado é que os agentes com prazos rígidos são distribuídos para hardware, enquanto que os agentes com prazos flexíveis são implementados em software.

A segunda questão se assemelha a referenciada em [10]. Os agentes utilizados seguem a arquitetura BDI. A aplicação do agente de hardware é semelhante ao de software, exceto que, no software, todos os parâmetros dos agentes podem ser transmitidos por chamadas de funções na linguagem de programação C/C++, enquanto que em hardware é necessário definir, especificamente, todas as entidades de agentes de hardware, as suas portas associadas e parâmetros em VHDL.

Figura 11: Auto reconfiguração da plataforma [10].



3.3.4 Teste do experimento

Para testar, colocou-se o robô em um cenário com duas peças e diversos obstáculos, sendo uma peça iluminada e a outra não. A partir de um ponto de saída, o objetivo do robô era deslocar-se até o ponto de chegada sem esbarrar nos obstáculos impostos aleatoriamente no caminho. Para realizar tal ação, o robô contava com diversos sensores, entre os quais uma luz noturna que facilitava sua movimentação na peça sem luz. Para [10], a experiência demonstra que a arquitetura reconfigurável baseada em agentes proposta é viável e pode trabalhar de forma eficiente para um sistema em tempo real complexo em ambientes altamente dinâmicos. O robô ainda foi submetido a outros testes, em geral variando a quantidade de obstáculos, podendo mais detalhes ser visualizados em [10].

3.4 Considerações sobre os estudos de caso

O trabalho [1] aborda a aplicação do paradigma de vários agentes em um sistema HW/SW híbrido reconfigurável. A utilização de tal paradigma tem o potencial de aumentar significativamente a flexibilidade, a eficiência, a capacidade de expansão e a facilidade de manutenção desses sistemas e de fornecer uma alternativa atraente para o atual conjunto de abordagens disjuntas que são atualmente aplicadas a esse domínio do problema. O autor, ainda, referencia que o próximo passo para a evolução desse trabalho é fazer um comparativo da abordagem multiagente apresentada com as técnicas ad hoc atuais que estão sendo aplicadas nos sistemas embarcados distribuídos e domínios de computação reconfigurável.

No trabalho [9], foi apresentada uma plataforma para desenvolver os drivers inteligentes, com uma abordagem codesign de hardware e software de uma forma flexível. Segundo [9], os resultados obtidos a partir de uma máquina CNC mostraram que é possível e fácil reunir controle e recursos de monitoramento utilizando o sistema multiagente Madcon. Como complementa o trabalho, o sistema foi desenvolvido com o intuito de simplificar a interação entre as funções de hardware e software em aplicações industriais, tornando simples a customização do

sistema, gastando menos tempo no seu desenvolvimento e na sua manutenção. Pelo estudo, foi possível concluir que o Madcon proporciona a produção de sistemas altamente reconfiguráveis, graças à modularidade oferecida pelos DRCS, e a reconfiguração de dispositivos FPGA.

O trabalho [10] propôs uma plataforma autorreconfigurável de agentes distribuídos para sistemas de tempo real. Segundo [10], as principais contribuições do trabalho desenvolvido são a arquitetura baseada em multiagente, pela qual um sistema unificado de metodologia codesign de hardware/software é aplicado para melhorar o desempenho do sistema e simplificar o projeto do sistema; a elaboração de um mecanismo unificado para realizar a comunicação independente entre os agentes e o estabelecimento de uma plataforma autorreconfigurável em que a lógica de controle de configuração está localizada no interior da matriz lógica FPGA.

4 Conclusão

Além de proporcionar um estudo sobre agentes e sistemas multiagente, este trabalho oportunizou sistematizar um conhecimento acerca de arquiteturas híbridas aplicadas a esses sistemas, com base em três diferentes estudos de caso sobre as arquiteturas FPGA.

Por meio dos trabalhos analisados, conclui-se que os agentes foram desenvolvidos, inicialmente, sob a perspectiva de dar dinamismo a aplicações que envolvessem somente software. Contudo, hoje, esse paradigma se estendeu a aplicações que englobam hardwares. O fato é uma inovação e ocorreu em razão de hardwares reconfiguráveis terem várias vantagens em relação a hardwares convencionais, como o aumento de flexibilidade, eficiência, capacidade de expansão e facilidade de manutenção de tais sistemas.

Futuramente, pretende-se expandir o estudo para acrescentar novas pesquisas que abordem a miscigenação de agentes com hardwares reconfiguráveis, assim como implementar agentes no software que poderá controlá-los em hardware. Com isso, objetiva-se a prevenção de falhas ou travamento de sistema causado por algum agente que, durante seu funcionamento, possa sofrer algum processo que o desative. Ainda tem-se como proposta realizar um comparativo para descobrir quais são as características semelhantes desse conjunto de trabalhos.

Agradecimentos

Os autores agradecem à Universidade Federal do Rio Grande (Furg), à Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (Fapergs) e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) pelo suporte financeiro na realização do presente trabalho. Agradecem, especialmente, aos professores Dr. José Rodrigo Furlanetto Azambuja e Dr. Denis Teixeira Franco, docentes no Programa de Pós-graduação em Computação do Centro de Ciências Computacionais da Furg pela orientação na realização deste trabalho, e à professora Dra. Diana F. Adamatti, docente do mesmo programa, pelas revisões e correções sugeridas.

Referências

[1] NAJI, H. R.; WELLS, B. E. On incorporating multi-agents in combined hardware/software based reconfigurable systems-a general architectural framework. In: IEEE. *System Theory, 2002. Proceedings of the Thirty-Fourth Southeastern Symposium on*. [S.l.], 2002. p. 344–348.

[2] ALVARES, L. O. *Introduçao aos sistemas multiagentes*. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 1997.

[3] FERBER, J. L'intelligence artificielle distribuée. *Le Courrier du CNRS*, CNRS, n. 80, p. 87–88, 1993.

[4] WOOLDRIDGE, M.; JENNINGS, N. R. Agent theories, architectures, and languages: a survey. In: *Intelligent agents*. [S.l.]: Springer, 1995. p. 1–39.

[5] GARCIA, A. C. B.; SICHMAN, J. S. Agentes e sistemas multiagentes. *Sistemas Inteligentes: fundamentos e aplicações*, 2003.

[6] SICHMAN, J. S. *Raciocínio social e organizacional em sistemas multiagentes: avanços e perspectivas.* Tese (Doutorado) — Universidade de São Paulo, 2003.

[7] BONNOT, P. et al. Definition and simd implementation of a multi-processing architecture approach on fpga. In: ACM. *Proceedings of the conference on Design, automation and test in Europe*. [S.l.], 2008. p. 610–615.

[8] LOO, S. et al. Handel-c for rapid prototyping of vlsi coprocessors for real time systems. In: IEEE. *System Theory, 2002. Proceedings of the Thirty-Fourth Southeastern Symposium on*. [S.l.], 2002. p. 6–10.

[9] MORALES-VELAZQUEZ, L. et al. Open-architecture system based on a reconfigurable hardware–software multi-agent platform for cnc machines. *Journal of Systems Architecture*, Elsevier, v. 56, n. 9, p. 407–418, 2010.

[10] MENG, Y. An agent-based reconfigurable system-on-chip architecture for real-time systems. In: IEEE. *Embedded Software and Systems, 2005. Second International Conference on*. [S.l.], 2005. p. 8–pp.