Arquitetura Heterogênea CPU+FPGA para Análise Formal de Conceitos

Lucas A. Maciel, João Paulo P. Novais, Matheus A. Souza
Mark Alan J. Song, Henrique C. Freitas

Departamento de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte, Brasil

{lamaciel, joao.novais, matheus.alcantara}@sga.pucminas.br

{song, cota}@pucminas.br

Abstract. *Algorithms for formal concept analysis are widely studied to extract computing intelligence patterns and knowledge discovery. However, they require high-performance processing due to their combinatorial characteristics. In this work, we design and evaluate a heterogeneous CPU+FPGA architecture to accelerate the concept extraction with large data sets. The results show a speedup of up to 3,95x with 120,63x more operations per Watt compared to a CPU-based version. Compared to the In-Close-2-BDD software, our architecture is faster (e.g. 4,06x) for several data sets, processing up to 1 million objects.*

Resumo. *Algoritmos para análise formal de conceitos são amplamente estudados para extrair padrões de inteligência computacional e descoberta de conhecimento. No entanto, eles exigem processamento de alto desempenho devido às suas características combinatórias. Neste trabalho, foi projetada e avaliada uma arquitetura heterogênea de CPU+FPGA para acelerar a extração de conceitos em grandes conjuntos de dados. Os resultados encontrados mostram um speedup de até 3,95x com até 120,63x mais operações por Watt em relação a uma versão executada em CPU. Em comparação com o software In-Close-2-BDD, essa arquitetura é mais rápida (e.g. 4,06x) para vários conjuntos de dados, processando até 1 milhão de objetos.*

1. Introdução

O crescente avanço de tecnologias e técnicas para tratar os vários dados heterogêneos gerados em diversos cenários, como redes sociais, plataformas de *streaming* e bioinformática, tem originado conjuntos complexos e volumosos de informações. Juntamente com essa evolução, o conceito de *Big Data* [Neshatpour et al. 2016] ganhou força, ao propor métodos capazes de processar os dados avaliados, de modo rápido e eficaz, permitindo com isso, a obtenção de resultados significativos, que contribuem diretamente na tomada de decisões e interpretação de contextos variados.

A Análise Formal de Conceitos, do inglês *Formal Concept Analysis* (FCA) é um método para representação de dados que pode ser aplicado em várias áreas do

conhecimento para extrair padrões de inteligência computacional, como por exemplo, na recuperação de informações [Codocedo and Napoli 2015]. Devido à importância da descoberta do conhecimento, outras abordagens, como *FcaBedrock* e *InClose* [Andrews and Orphanides 2010], têm a capacidade de reestruturar conjuntos de dados para minimizar comparações desnecessárias em busca de um maior desempenho. Contudo, lidar com grandes bases de dados de maneira oportuna é um grande desafio.

A extração de conceitos formais é feita com base em um contexto formal, que possui duas dimensões denominadas *Objetos* e *Atributos* que representam, respectivamente, os registros e as características de um contexto. Com isso, diversas técnicas que associam objetos e atributos podem ser amplamente estudadas [Lorenzo et al. 2017]. No entanto, com o constante desenvolvimento e geração de grandes conjuntos de dados, o processamento computacional torna-se cada vez mais complexo ou até mesmo inviável [Neto et al. 2018b].

De acordo com o problema de processamento relacionado a grandes quantidades de dados (objetos e atributos), o objetivo deste trabalho é apresentar o projeto e avaliação de uma arquitetura heterogênea para Análise Formal de Conceitos baseada em uma Unidade Central de Processamento, do inglês *Central Processing Unit* (CPU) e um *Field Programmable Gate Array* (FPGA). Assim, nossas contribuições para o estado da arte são as seguintes:

- Arquitetura heterogênea de CPU+FPGA para Análise Formal de Conceitos;
- Maior desempenho (menor tempo de execução) e eficiência energética (mais operações por Watt) em comparação com uma arquitetura homogênea executada em um processador;
- Escalabilidade ao processar grandes conjuntos de dados (por exemplo, 1 milhão de objetos com 25 atributos) quando comparado aos trabalhos relacionados atuais, desenvolvidos na área de FCA.

Este trabalho está organizado da seguinte maneira. A Seção 2 apresenta uma breve explicação da Análise Formal de Conceitos e alguns trabalhos correlatos. A arquitetura proposta e o funcionamento da plataforma heterogênea desenvolvida são detalhados na Seção 3. A metodologia de avaliação adotada é descrita na Seção 4. Os resultados obtidos são apresentados na Seção 5 e as conclusões finais na Seção 6.

2. Background

A *Formal Concept Analysis* é um ramo da matemática iniciado por [Davey and Priestley 1990] denominado teoria de reticulados, definidos como conjuntos parcialmente ordenados. Estes conceitos formais também foram estudados por [Grätzer 2002] como a teoria da estrutura geral, e reestruturados em [Wille 2009] como a teoria de estrutura hierárquica. As fundamentações matemáticas para a área de FCA são apresentadas em [Ganter and Wille 2012], e o significado geral deste ramo de pesquisa pode ser descrito como:

- *Analysis*: Observação e manipulação dos dados fornecidos;
- *Concept*: Determina os conjuntos aos quais os dados são estruturados para interpretação humana da realidade;

- *Formal*: Refere-se à fundamentação matemática da geração de conceitos através das análises.

Com objetivo de propor um melhor entendimento das definições apresentadas, esta seção está dividida da seguinte forma: a Seção 2.1 apresenta uma descrição de um contexto formal, posteriormente, na Seção 2.2 o conceito formal é retratado com uma explicação de extração de um conceito através de um contexto. Por fim, uma breve explicação do funcionamento do algoritmo de Força Bruta é descrita na Seção 2.3 e alguns trabalhos relacionados são apresentados na Seção 2.4.

2.1. Contexto formal

Um contexto formal possui a notação (O, A, R), sendo O e A conjuntos contendo elementos denominados objetos e atributos, respectivamente, e sendo R uma relação binária chamada incidência, onde $R \subseteq O \times A$. Todo elemento da incidência tem a notação oRa, em que uma relação entre objeto e atributo, pode ser lida como "o objeto o tem o atributo a". Quaisquer subconjuntos $E \subseteq O$ e $I \subseteq A$, podem ser determinados seus conjuntos derivados (que respectivamente são os atributos comuns aos objetos e os objetos que possuem os atributos), por meio das seguintes operações de derivação:

$$E' = \{a \in A \mid \forall o \in E : \ oRa\}$$

$$I'' = \{o \in O \mid \forall a \in I : \ oRa\}$$

A tabela de incidências é uma maneira adequada para representar um contexto formal, onde cada linha representa os objetos e cada coluna os seus respectivos atributos. Quando o objeto possui um atributo, uma incidência é assinalada através de um símbolo (por exemplo X) indicando que há uma relação entre os objetos e atributos capazes de formar um contexto, como exemplificado na Tabela 1, sendo A e O os atributos e objetos, com m e n referindo-se aos seus índices.

Tabela 1. Tabela de incidências

	A1	**A2**	**...**	**Am**
O1	x	x	...	
O2	x		...	x
...	...	...	...	...
On		x	...	x

Contextos formais representam conjuntos de objetos que possuem ou não atributos, ou seja, a relação entre esses dois elementos é categorizada em verdadeiro ou falso. No entanto, os atributos de um conjunto de dados tendem a possuir diversos valores, sendo desta maneira, necessário definir um contexto formal multi-valorado, ou seja, se o atributo A possui dois valores, é necessário dividir este atributo em dois (A' e A'') de forma que seja possível realizar a análise de maneira binária (verdadeiro ou falso).

2.2. Conceitos formais

Um contexto formal (Seção 2.1) define conceitos formais, e para isto, em cada conjunto de atributos que possuem objetos (após sua derivação), torna-se necessário que os mesmos objetos sejam derivados para encontrar um conjunto de atributos idênticos. Isto é, se

forem derivados $A1$ e $A3$ e encontrados $O1$ e $O3$, para que os mesmos sejam definidos como um conceito é necessário derivar $O1$ e $O3$, encontrando exatamente $A1$ e $A3$ como resultado, semelhante ao exibido no quadro de *Conceito* da Figura 1. Já o quadro de *Não é Conceito*, apresenta valores distintos, pois quando é derivado somente $A1$ são obtidos $O1$ e $O3$, no entanto, quando derivados $O1$ e $O3$ são encontrados $A1$ e $A3$.

	A1	A2	A3	A4
O1	x	x	x	
O2		x	x	
O3	x		x	
O4		x		x

Conceito	
a)	A1,A3 => O1, O3
b)	O1, O3 => A1,A3

Não é Conceito	
a)	A1 => O1,O3
b)	O1,O3 => A1, A3

Figura 1. Conceito Formal

2.3. Força Bruta

Existem diversos algoritmos capazes de encontrar os conceitos formais de um contexto formal, como o Close-by-One(CbO), *In-Close* e *In-Close2* [Andrews 2015]. Além desses, um dos métodos usados para extrair conceitos formais é o *Força Bruta*. Este método possui baixa complexidade de desenvolvimento e alto desacoplamento, sendo estas características vantajosas para o desenvolvimento de uma arquitetura em *hardware*. Para gerar um conceito formal, é necessário submeter um grupo de atributos a uma função de derivação, capaz de gerar uma combinação de atributos. Por exemplo, dado um conjunto de atributos (a, b, c), as seguintes combinações devem ser geradas a, b, c, ab, bc, ac e abc, implicando em um grupo de objetos contendo um conjunto inicial de atributos. Esses objetos são submetidos à derivação por implicar um novo conjunto de atributos, de modo a gerarem conceitos, quando forem iguais aos atributos iniciais.

Algoritmo 1: GeradorConceitos

Entrada: $QtdAtributos$
Saída: $ResConceitos$

```
1  início
2      Filtros = GerarCombinacao(QtdAtributos);
3      DataROM = LerBaseDados();
4      para (i = 1 → Filtros.Length) faça
5          RamRegistros ← BC_Go(Filtros[i], DataROM);
6          Aux ← BC_Back(Filtros[i], RamRegistros);
7          se Aux então
8              ResConceitos++;
9          fim
10     fim
11     retorna ResConceitos;
12 fim
```

O algoritmo baseado em FPGA para FCA usa três funções para extração formal de conceitos, são elas: GeradorConceitos, BC_Go e BC_Back. A função principal $GeradorConceitos$, como apresentada no Algoritmo 1, necessita gerar todas as combinações de atributos (e.g. função $GerarCombinacao$) e armazenar os seus resultados na variável $Filtros$. Posteriormente, os dados da base são lidos, em seguida é realizada a derivação de cada filtro em relação a base de dados e por fim, os conceitos encontrados são confirmados.

A função BC_Go apresentada no Algoritmo 2, contêm as entradas $Filtro$ e $DataROM$, sendo que a variável $Filtro$ possui apenas uma das combinações geradas na função anterior. Já o $DataROM$ é passado com todos os objetos para validações. Essa função é responsável por realizar a derivação do filtro para encontrar os objetos relacionados e armazenados na variável $RamRegistros$.

Algoritmo 2: BC_Go

```
Entrada: Filtro, DataROM
Saída: RamRegistros
1  início
2      para (i = 1 → DataROM.Length) faça
3          RamRegistros ← DerivarConceito(Filtro, DataROM[i]);
4      fim
5      retorna RamRegistros;
6  fim
```

Na função BC_Back (Algoritmo 3) é esperado como entrada o $Filtro$ e os resultados da função BC_Go. Contudo, diferentemente da função anterior, o objetivo desta é encontrar os atributos que estão relacionados com os objetos gerados através da derivação. Após as validações, caso o resultado da derivação for o mesmo do filtro inicial, é retornado verdadeiro para confirmar o conceito. Caso contrário, a função retorna falso.

Algoritmo 3: BC_Back

```
Entrada: Filtro, RamRegistros, SaveConcept
Saída: Booleano (Verdadeiro ou Falso)
1  início
2      Aux_Atributos = TodosAtributosSelecionados
3      para (i = 1 → RamRegistros.Length) faça
4          Op ←
             AtributosANDRegistro(RamRegistros[i], Aux_Atributos);
5          se Op = Filtro então
6              retorna Verdadeiro;
7          fim
8      fim
9      retorna Falso;
10 fim
```

2.4. Trabalhos Relacionados

A área de FCA é considerada uma teoria importante para representação de formalização dos conceitos, e deste modo, a extração de conceitos possui uma complexidade combi-

natória, sendo determinada pela sua alta dimensionalidade. A literatura apresenta diversos algoritmos para extração de conceitos com objetivo de alcançar uma alta dimensionalidade, isto é, ter a capacidade de extrair conhecimento de bases de dados cada vez maiores e mais completas.

Em [Andrews 2011] é apresentado o algoritmo *In-Close2*, sendo uma variação do método *Close-by-One* (CbO). Uma aplicação para este algoritmo é dada usando o método de suporte mínimo, para reduzir o tamanho e complexidade de um grande contexto formal. Contudo, esse trabalho compara o desempenho alcançado por dois algoritmos, o *In-Close2* e outra variação do CbO, cujo nome é dito por *Fast Close-by-One* (FCbO). O algoritmo *In-Close* apresenta um melhor desempenho na maioria dos casos, sendo este melhor recomendado para os casos avaliados.

Já no trabalho retratado em [Andrews 2015], são estudados pontos de fixação *Galois Connections* que formam padrões em dados relacionais binários, podendo ser tratados na área de pesquisa de FCA. Com isso, no trabalho apresentado foi verificado que os melhores algoritmos para este contexto, são derivados do algoritmo CbO. Logo, o artigo mostra uma comparação gráfica do desempenho de diversos algoritmos, sendo eles: CbO, $FCbO$, *In-Close*, *In-Close2* e *In-Close3*. Os algoritmos utilizaram as mesmas estruturas e técnicas para fornecer um campo de atuação nivelado para uma melhor avaliação.

A pesquisa de novos algoritmos para a área de FCA que alcançam um processamento com alta dimensionalidade ainda é o grande foco. No trabalho apresentado por [Santos et al. 2018], é desenvolvido um novo algoritmo para contextos de alta dimensionalidade, cujo nome é descrito por *ImplicPBDD* (baseado no algoritmo *PropIm*) que utiliza uma estrutura BDD, sendo esta uma estrutura simplificada de representação do contexto formal para aprimorar a extração de conhecimento. No trabalho apresentado, é criada uma base sintética com variações de dimensão e objetos, obtendo um desempenho de 80% a mais que o seu antecessor, ao avaliar a eficiência do algoritmo.

As pesquisas em FCA possuem grande importância em várias aplicações, como por exemplo em mineração de dados, robótica, medicina, etc. Sendo assim, como apresentado em [Neelima and Sarma 2019], uma nova abordagem baseada em FCA e *Fuzzy Logic Controller* (FLC) é estudada, para obter respostas de uma ação humana, em que, a parte de FCA analisa os dados de uma base que possuem relações entre o conjunto de objetos e atributos prévios de ideias de pensamentos humanos, e gera conceitos/regras para serem aplicados na FLC, construindo regras *Fuzzy*, que permitem a obtenção de respostas de uma ação posteriormente. A área de FCA também é uma boa alternativa para identificar estruturas conceituais em rede social, como pode ser visto em [Neto et al. 2018a]. Este trabalho mostra modelos computacionais baseados em implicações que representam e analisam subestruturas das redes sociais. Sendo assim, foi possível descobrir padrões de acesso e comportamento em redes sociais específicos.

A busca por extração de conhecimento em uma base de dados com alta dimensionalidade e conjunto de dados é um dos maiores focos de pesquisa. No entanto, há uma grande necessidade de recursos computacionais para processar um alto volume de informações. Contudo, estudos com FPGA mostram ser bastante promissores para mineração de dados e tratamento de um grande volume de dados. Um levantamento de diversos trabalhos que mostraram o uso de arquiteturas heterogêneas, referindo-se ao

uso de diversos dispositivos em conjunto, como CPU, GPU, FPGA, foi realizado em [Andrade and Crnkovic 2019]. Sendo assim, no trabalho é apresentado o estudo de 28 pesquisas relacionadas à computação heterogênea, de forma à realizar críticas construtivas da relevância e problemas que esta área de pesquisa pode oferecer.

Diante da importância das diversas pesquisas para extração de conceitos formais, e também as vantagens que a computação heterogênea oferece, este trabalho visa analisar os ganhos em FCA ao aplicar uma abordagem de computação heterogênea, ainda não explorada pelos trabalhos correlatos. Assim, é apresentada na Seção 3 a proposta do projeto de uma arquitetura heterogênea (CPU+FPGA) para extração de conceitos formais.

3. Arquitetura Heterogênea CPU+FPGA

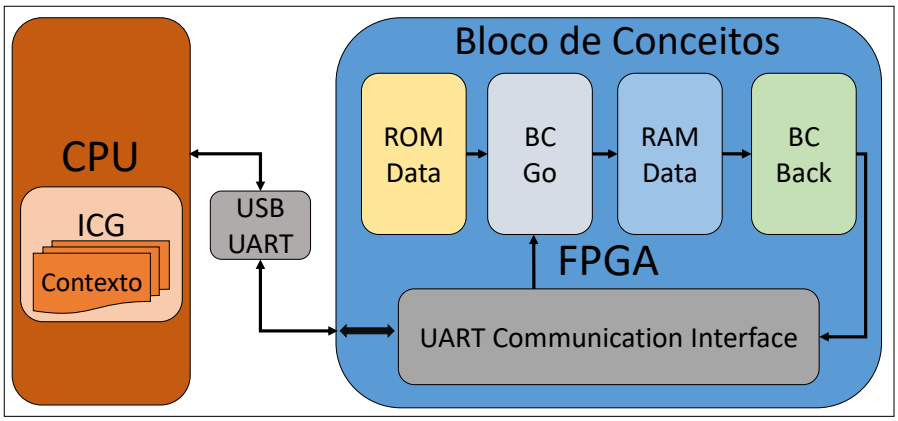

Figura 2. Arquitetura do Formal Concept Generator (FCG)

O protótipo *Formal Concept Generator* (FCG) é baseado em um algoritmo de força bruta (apresentado na Seção 2.3) e ilustrado pela Figura 2. O FCG é um conjunto de blocos lógicos descritos em *VHSIC Hardware Description Language* (VHDL) e implementado em um FPGA. A arquitetura do Bloco de Conceitos (BC) é interconectada via USB (*Universal Serial Bus*) UART (*Universal Asynchronous Receiver-Transmitter*) em uma CPU, sendo responsável pelo processamento do *Incidence Combination Generator* (ICG), descrito em linguagem C.

Com relação às entradas de cada bloco principal, o ICG lê a quantidade e o conjunto de valores de atributos que existem no contexto. Para isso, a arquitetura do BC possui as entradas de *clock*, *reset*, número de objetos e atributos e uma combinação de atributos de incidência gerados pelo ICG. As saídas do FCG correspondem aos conceitos gerados compostos por objetos e atributos. Para a implementação no FPGA avaliado, foram utilizadas entradas de dados de 16 bits, onde cada bit representa um atributo específico do objeto. A arquitetura proposta (Figura 2) pode ser explicada em três partes principais: CPU, FPGA e comunicação.

A primeira parte é composta pela CPU e por um algoritmo (ICG) implementado na linguagem C, responsável por gerar todas as combinações possíveis de um conjunto de dados de até 16 atributos, representado por $A[n]$. O algoritmo desenvolvido faz chamadas recursivas para produzir cada combinação ($A[i]$) que deve ser enviada para o FPGA, responsável por processar as informações e permitir uma próxima iteração.

Em relação ao FPGA (segunda parte), o protótipo desenvolvido tem quatro blocos funcionais, dois deles representando as memórias internas do *hardware* (implementadas pela ferramenta *IP Catalog* do *software Quartus Prime 16.0*) e os outros dois, as unidades de processamento de conceito (BC). Para armazenar a tabela de contexto original, uma memória ROM (*Read Only Memory*) de 65536 posições, foi utilizada. Cada posição representa um objeto com até 16 bits. Um bit definido como 1 representa um atributo presente no objeto analisado.

A partir do armazenamento do contexto na memória ROM e do número de objetos e atributos informados pelo usuário, o bloco BC_Go é acionado recebendo um $input$ $A[i]$ da CPU. Ele identifica todos os objetos que possuem uma combinação de atributos ($A[i]$) definida pela CPU e executa em paralelo uma comparação lógica AND dos atributos de um objeto, realizando o primeiro processo de derivação. O FCG é uma arquitetura com *pipeline* de três estágios e, por esse motivo, um objeto pode ser lido na ROM, enquanto outro é processado e um terceiro é enviado para o bloco da memória RAM (*Random Access Memory*). Essa memória, possui canais de entrada e saída que permitem o armazenamento de todos os objetos identificados pelo bloco BC_Go e a leitura dos atributos pelo próximo bloco de processamento (BC_Back).

O último bloco é o BC_Back, responsável por executar o segundo processo de derivação a partir de uma operação lógica paralela AND com todos os objetos armazenados na RAM, para encontrar seus atributos em comum. De uma maneira iterativa, cada objeto é lido e processado com seu predecessor. Este procedimento é repetido até que todos os dados sejam analisados. Finalmente, o conjunto de atributos encontrados (A') é comparado com a combinação gerada pela CPU ($input$). Se eles forem iguais, os objetos e atributos analisados compõem um novo conceito, que é retornado para a CPU, caso contrário, o FPGA limpa seus registros e aguarda uma nova chamada.

A comunicação entre FPGA e CPU (terceira parte) é realizada via cabo USB UART. Com isso, é necessário instalar o driver FT232R na CPU e configurar a comunicação serial para definir a porta USB e a taxa de transferência de dados. Do lado do FPGA, é necessário implementar os pinos RX e TX na arquitetura, além da realizar a configuração da ferramenta Intel Nios II.

4. Metodologia de Avaliação

De acordo com as seções anteriores, o *Formal Concept Generator* (FCG) é um projeto heterogêneo baseado nas linguagens C (CPU) e VHDL (FPGA). A versão homogênea e sequencial em *software* (CPU-*only*) foi projetada em C e executada em uma máquina com processador AMD Bulldozer com 2.10 GHz e 32 GB de RAM. A proposta heterogênea CPU+FPGA foi avaliada na mesma máquina, interconectada via USB UART à placa DE2-115 com o FPGA Cyclone IV-E EP4CE115F29C7 da Intel/Altera, operando a 200 MHz. O *software Quartus Prime 16.0* foi utilizado para síntese do projeto da arquitetura.

No decorrer do trabalho, as estratégias de avaliação foram divididas em etapas. Na primeira, o *software* SCGaz [Rimsa et al. 2013] foi utilizado para criar 7 contextos de dados, variando o número de atributos de 10 a 16 e de objetos de 1000 a 65000, para serem aplicados como bases de entrada do algoritmo avaliado. A segunda etapa, compreendeu em analisar o tempo de execução da proposta heterogênea CPU+FPGA, comparado com o obtido na abordagem CPU-*only*, desde a geração das combinações, até a validação dos

conceitos. Já na terceira etapa, a eficiência energética foi avaliada com a métrica Milhões de Operações por Segundo (MOPS) por Watt, ou seja, MOPS/Watt. A quantidade total de operações avaliadas nessa métrica, levam em consideração o comportamento do algoritmo, realizando o produto entre objetos, atributos e comparações lógicas. O consumo de energia para o FPGA foi avaliado utilizando o Intel *PowerPlay* EPE. Para o processador AMD, foi utilizada a ferramenta PowerTOP [Intel 2007]. Vale a pena destacar, que as etapas 2 e 3, utilizam como entrada de dados, os contextos gerados na etapa 1.

Por fim, na última etapa, foram gerados 8 contextos, contendo 25 atributos e uma densidade máxima (referente à quantidade de atributos selecionados em cada objeto) permitida para cada um deles, variando o número de objetos entre 1000 e 1000000. Com os contextos gerados, foi realizada uma análise, para avaliar a escalabilidade da arquitetura heterogênea, aumentando o seu número de atributos suportados de 16 para 25, e sintetizando-a no Intel FPGA Arria 10 GX 1150 executando à 800 MHz, a fim de compará-la com uma versão em *software* de última geração chamada *In-Close-2* com *Binary Decision Diagramas* (BDD). Essa versão foi desenvolvida no trabalho relacionado [Neto et al. 2018b], que apresentou resultados de alto desempenho executando os mesmos conjuntos de dados em uma arquitetura com um processador de nove núcleos e 112 GB de RAM, fornecidos pela plataforma em nuvem Microsoft Azure.

5. Resultados

No decorrer desta seção, são discutidos os resultados obtidos no trabalho. A Tabela 2 apresenta a ocupação de recursos do FPGA, considerando o número total de pinos de entrada, registradores, elementos lógicos e bits de memória do *hardware* sintetizado. Para realizar o processamento correto dos conceitos, o *Formal Concept Generator* precisa armazenar o contexto, objetos e atributos que serão analisados. Assim, há um grande número de bits de memória utilizados nos FPGAs escolhidos.

Tabela 2. Ocupação de recursos do FPGA

Elementos	Disponível		Utilizado	
	Cyclone IV	**Arria 10**	**Cyclone IV**	**Arria 10**
Pinos	529	768	200 (38%)	279 (36%)
Registradores	114480	1708800	157 (0,14%)	204 (0,01%)
Lógicos	114480	1150000	532 (0,46%)	532 (0,05%)
Bits Mem.	3981312	67244312	2097152 (53%)	50000152 (74%)

A Tabela 3 mostra o tempo de execução do algoritmo de geração de conceitos nas arquiteturas CPU-*only* (2.10 GHz) e CPU+FPGA (200 MHz). Os resultados mostraram que a proposta heterogênea de CPU+FPGA pode identificar os conceitos em tempos menores que sua versão CPU-*only*, sendo até 3,95x e 3,14x mais rápido com 1000 e 65000 objetos, respectivamente, devido a paralelização das operações lógicas *AND* dos atributos, que permite ao FPGA processar os registros avaliados com uma maior velocidade.

Uma análise da eficiência energética é mostrada na Tabela 4 em número de milhões de operações (MOPS) por Watt. Essa métrica mostra que esta arquitetura heterogênea baseada em FPGA executa até 120,63x (1000 objetos) e 104,74x (65000 objetos) MOPS/Watt a mais do que a arquitetura CPU-*only*. Além disso, na avaliação do

Tabela 3. Tempo de execução (segundos)

Obj	Atr	CPU-*only*	CPU+FPGA	Obj	Atr	CPU-*only*	CPU+FPGA
1000	**10**	0,0821	0,0208	**16000**	**14**	15,9991	5,1211
2000	**11**	0,2669	0,0808	**32000**	**15**	64,2584	20,4814
4000	**12**	1,0242	0,3209	**65000**	**16**	265,5377	84,5021
8000	**13**	4,0606	1,2809	*	*	*	*

CPU (AMD Bulldozer) + FPGA (Cyclone IV)
CPU-only: Força Bruta — CPU+FPGA: Força Bruta

modelo baseado em FPGA, é apresentado um consumo médio de potência de 1,99 Watts, enquanto o *thermal design power* (TDP) do AMD Bulldozer é em média 60,74 Watts. Os resultados também apresentam uma redução esperada de MOPS/Watt ao longo das comparações, devido ao aumento do tempo de execução em cada carga de trabalho, a medida que ocorre o crescimento do número de objetos e atributos.

Tabela 4. MOPS/Watt

Obj	Atr	CPU-*only*	CPU+FPGA	Obj	Atr	CPU-*only*	CPU+FPGA
1000	**10**	4999	603044	**16000**	**14**	525	50299
2000	**11**	3328	326805	**32000**	**15**	272	26722
4000	**12**	1845	176014	**65000**	**16**	133	13930
8000	**13**	995	94305	*	*	*	*

CPU (AMD Bulldozer) + FPGA (Cyclone IV)
CPU-only: Força Bruta — CPU+FPGA: Força Bruta

A fim de satisfazer um cenário com alta densidade e grande conjunto de dados discutido por trabalhos do estado da arte, foi realizada a síntese da arquitetura proposta baseada no algoritmo de *força bruta*, no modelo de FPGA Intel Arria 10. Com isso, fez-se a comparação de seus tempos de execução com contextos possuindo densidade máxima (Tabela 5), com resultados encontrados pelo *In-Close-2-BDD* (CPU-*only*).

Tabela 5. Tempo de geração de conceitos em segundos (25 Atributos)

Obj	CPU-*only*	CPU+FPGA	Obj	CPU-*only*	CPU+FPGA
1000	84,96	0,0057	**100000**	202,812	50,0012
5000	124,84	0,1250	**250000**	229,878	312,5020
10000	135,59	0,5008	**500000**	240,597	1250,0032
50000	177,39	12,5010	**1000000**	*	5000,0057

CPU (AMD Bulldozer) + FPGA (Arria 10)
CPU-only: In-Close-2-BDD — CPU+FPGA: Força Bruta

A versão CPU+FPGA executa a geração de contextos por *força bruta* em um tempo menor que o *In-Close-2-BDD* [Neto et al. 2018b], e.g., sendo 4,06x mais rápido com 100000 objetos. No entanto, a abordagem do trabalho relacionado apresenta tempos

melhores para 250000 e 500000 objetos. O aumento no tempo de execução desta proposta é diretamente proporcional ao crescimento de objetos. Deste modo, quando ocorre uma diferença de crescimento maior (e.g., de 10000 a 50000), o aumento no tempo será consequentemente maior, levando-se em consideração as cargas de trabalho avaliadas. No entanto, esta abordagem heterogênea é mais escalável e pode executar contextos com 1 milhão de objetos e 25 atributos, o que não foi alcançado pelo trabalho do estado da arte relacionado, devido a falta de poder computacional da arquitetura CPU-*only*. Os resultados mostraram a eficiência do uso da plataforma heterogênea baseada no algoritmo *força bruta*. Ela apresentou um desempenho superior em várias avaliações, embora a força bruta seja menos inteligentes que o algoritmo *In-Close-2-BDD*. Desta forma, aponta-se que a arquitetura heterogênea baseada em FPGA é responsável por melhorar consideravelmente a extração de conceitos formais, devido a sua capacidade reconfigurável e escalável.

6. Conclusão e trabalhos futuros

Aplicações executadas no campo de *Big Data*, exigem constantemente um alto poder computacional para gerar resultados satisfatórios. Considerando este cenário, foi apresentada neste trabalho uma arquitetura heterogênea CPU+FPGA que provou ser mais eficiente em comparação a uma abordagem CPU-*only*, utilizando o algoritmo de *força bruta* para geração de conceitos, geralmente aplicado na área de Análise Formal de Conceitos (FCA). A arquitetura desenvolvida também mostrou sua eficiência em operar grandes bases de dados, devido a flexibilidade de ser implementada em dispositivos FPGA distintos e robustos, permitindo o processamento de uma maior quantidade de objetos e atributos.

Os resultados encontrados mostraram a eficácia da arquitetura proposta, ao ser desenvolvida com uma baixa complexidade, utilizando poucos recursos do FPGA. Esta abordagem heterogênea provou ser mais rápida do que uma versão CPU-*only*, obtendo um *speedup* de até 3,95x e 120,63x ao comparar os tempos de execução e eficiência energética (MOPS/Watt), respectivamente. Além disso, é retratado que essa proposta baseada em FPGA foi capaz de processar um contexto de até 1 milhão de objetos e 25 atributos, o que não foi alcançado por um dos trabalhos relacionados, considerado como um estado da arte em FCA. Durante essa comparação, em alguns cenários a proposta apresentada obteve desempenhos inferiores a versão *In-Close-2-BDD*, porém conseguiu mostrar ser escalável em termos de dispositivos suportados e de objetos processados. A contribuição deste trabalho é a elaboração de uma arquitetura heterogênea e sua implementação em *C* para CPU e VHDL para FPGA, do algoritmo *força bruta* de FCA, apresentando sua avaliação de desempenho, energia e escalabilidade.

Como trabalhos futuros, será realizado o desenvolvimento de um projeto paralelo baseado no OpenCL para diferentes algoritmos, como o *força bruta* e o *In-Close-2-BDD*. Além disso, eles serão avaliados em outras plataformas heterogêneas de CPU+FPGA (e.g, o Intel HARP) mantendo a mesma estratégia descrita neste artigo, de modo a alcançar uma maior escalabilidade que poderá ser aplicada nas áreas de FCA e *Big Data*.

Agradecimentos

Este trabalho foi financiado, em partes, pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Agradecemos também ao CNPq, FAPEMIG, PUC Minas e *Intel Hardware Accelerator Research Program* (HARP) pelo suporte no desenvolvimento do trabalho.

Referências

Andrade, H. and Crnkovic, I. (2019). A review on software architectures for heterogeneous platforms. *arXiv preprint arXiv:1905.01695*.

Andrews, S. (2011). In-close2, a high performance formal concept miner. In *International Conference on Conceptual Structures*, pages 50–62.

Andrews, S. (2015). A 'best-of-breed'approach for designing a fast algorithm for computing fixpoints of galois connections. *Information Sciences*, 295:633–649.

Andrews, S. and Orphanides, C. (2010). Analysis of large data sets using formal concept lattices. *Proceedings of the 7th Inter. Conf. on Concept Lattices and their Applications*, pages 104–115.

Codocedo, V. and Napoli, A. (2015). Formal concept analysis and information retrieval–a survey. In *International Conference on Formal Concept Analysis*, pages 61–77.

Davey, B. and Priestley, H. (1990). Introduction to lattices and order cambridge univ. *Press, Cambridge*.

Ganter, B. and Wille, R. (2012). *Formal concept analysis: mathematical foundations*. Springer Science & Business Media.

Grätzer, G. (2002). *General lattice theory*. Springer Science & Business Media.

Intel (2007). Powertop. Disponível em: https://01.org/powertop/.

Lorenzo, E. R., Cordero, P., Enciso, M., Missaoui, R., and Mora, A. (2017). An axiomatic system for conditional attribute implications in triadic concept analysis. *Int. J. Intell. Syst.*, 32:760–777.

Neelima, C. and Sarma, S. S. (2019). Blended intelligence of fca with flc for knowledge representation from clustered data in medical analysis. *International Journal of Electrical and Computer Engineering*, 9(1):635.

Neshatpour, K., Sasan, A., and Homayoun, H. (2016). Big data analytics on heterogeneous accelerator architectures. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–3.

Neto, S. M., Dias, S., Missaoui, R., Zárate, L., and Song, M. (2018a). Identification of substructures in complex networks using formal concept analysis. *International Journal of Web Information Systems*, 14(3):281–298.

Neto, S. M., Zárate, L. E., and Song, M. A. (2018b). Handling high dimensionality contexts in formal concept analysis via binary decision diagrams. *Information Sciences*, 429:361 – 376.

Rimsa, A., Song, M. A. J., and Zárate, L. E. (2013). Scgaz - a synthetic formal context generator with density control for test and evaluation of fca algorithms. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3464–3470.

Santos, P., Ruas, P., Neves, J., Silva, P., Dias, S., Zárate, L., and Song, M. (2018). Implicpbdd: A new approach to extract proper implications set from high-dimension formal contexts using a binary decision diagram. *Information*, 9(11):266.

Wille, R. (2009). Restructuring lattice theory: an approach based on hierarchies of concepts. In *International Conference on Formal Concept Analysis*, pages 314–339.