

SoC 設計プラットフォーム ESP を用いた プロセッサ開発環境の検討

四之宮 直輝¹ 土居 拓矢¹ 田中 知成¹ 廖 望¹ 密山 幸男¹

概要：オープンソースプロセッサである RISC-V が注目され、特定用途向けアクセラレータを搭載するプロセッサの設計事例が多数報告されている。本稿では、アクセラレータを搭載する高効率プロセッサを開発するための開発環境として、SoC 設計プラットフォーム ESP (Embedded Scalable Platforms) について検討を行った。コアプロセッサに RISC-V を使用し、積和演算アクセラレータを搭載した SoC を設計した。Xilinx VC707 をターゲットボードとして合成を行い、アクセラレータを搭載した場合と搭載しなかった場合について比較評価を行った。

キーワード：SoC, RISC-V, ESP, 開発環境, アクセラレータ

1. はじめに

近年、画像認識、音声認識、自動運転など様々な用途に機械学習が用いられ、これまでコンピュータが担ってきた単純な信号処理を超えて、認識、予測、判断といった高度な情報処理が可能となりつつある。今後は自動車やスマートフォン、IoT 端末など電力資源の限られたところでの活用が期待されており、高度なエッジコンピューティングの実現が求められている。本研究では、高い性能と低消費電力の両立を実現する特定アプリケーションドメイン向け高効率プロセッサの開発を目指して、アクセラレータを搭載するプロセッサの開発環境を構築する。RISC-V[1]をコアプロセッサとして用い、SoC 設計プラットフォームに ESP (Embedded Scalable Platforms)[2][3]を用いて環境の構築を行った。

2. RISC-V の概要

RISC-V はカリフォルニア大学バークレー校で開発された RISC (Reduced Instruction Set Computer) に基づく命令セットアーキテクチャ (ISA) である。RISC-V は基本命令に加えて拡張命令を取捨選択することができる。基本命令には、算術命令、論理演算命令を含んでいる。レジスタのビット幅は 32, 64, 128 ビットから選択できる。拡張命令は、乗除算命令(M), アトミック命令(A), 単精度(F), 倍精度(D), 4倍精度(Q)の浮動小数点命令, 16bit 短縮命令(C)など 13 種類用意されている。拡張命令から必要な命令のみを基本命令に付け加えることで、実装規模を抑えながら高い性能を実現することが可能である。

3. SoC 設計プラットフォーム ESP の概要

ESP はコロンビア大学が開発したヘテロジニアス SoC 設計プラットフォームであり、プロセッサ、アクセラレータ、メモリなどを搭載する SoC のシステムレベル設計が可能である。プロセッサコアには RISC-V ISA の Ariane[4]や、

SPARC V8 ISA の LEON3[5]を使用することができる。アクセラレータの設計は C/C++, SystemC, SystemVerilog や VerilogHDL を用いて設計することができる。図 1 に ESP の設計フローの概要を示す。

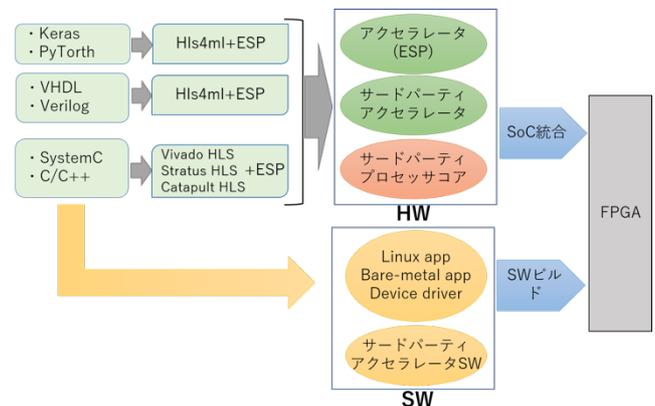


図 1 ESP 設計フロー

ESP を用いたヘテロジニアス SoC の設計は、タイルベースで行う。図 2 に示すように、タイル要素としてプロセッサ、アクセラレータ、メモリインタフェース、I/O の 4 種類がある[2]。それぞれのタイルの個数や配置は自由に変更することが可能であるが、CPU、メモリ、I/O をそれぞれひとつ以上配置する必要がある。

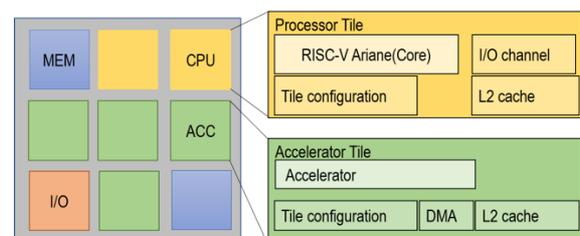


図 2 3x3 インスタンスの SoC 設計例[2]

¹ 高知工科大学

アクセラレータとプロセッサの結合は、Tightly-Coupled Accelerators (TCA)と Loosely-Coupled Accelerators (LCA)の2通り用意されている。TCAではアクセラレータはプロセッサコアに内蔵される。一方、LCAではアクセラレータはプロセッサコアの外にあり、バスを介してプロセッサと接続される。

4. アクセラレータ搭載 SoC 設計の概要

実験的に、アクセラレータを搭載する SoC の設計を行う。プロセッサとの結合に LCA を採用する場合、ESP を用いたアクセラレータタイトルの設計は図3に示す設計フローに従って行う。

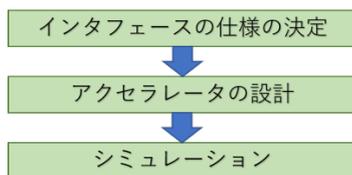


図3 アクセラレータタイトルの設計フロー

インタフェース仕様の設定とアクセラレータの記述は ESP 上で行う。インタフェース仕様として、データビット幅やバッチ処理係数などを設定する。

アクセラレータは C 言語で記述して、高位合成を適用して生成する。図4に示すように積和演算アクセラレータを C 言語で記述した。ESP では、積和演算アクセラレータはチュートリアルとして用意されているため、変数や関数があらかじめ記述されたフォーマットを用いて計算カーネルのみを記述した。

```

for (int i = 0; i < in_len; i += 2) {
    acc += _inbuff[i] * _inbuff[i+1];
    vector_index += 2;
    if (vector_index == mac_len) {
        _outbuff[vector_number] = acc;
        acc = 0;
        vector_index = 0;
        vector_number++;
    }
}
  
```

図4 積和演算アクセラレータの記述例

5. 実装結果

Xilinx 社 VC707 評価キットをターゲット FPGA ボードとして、SoC を合成した。アクセラレータ搭載 SoC と非搭載

SoC で FPGA 使用リソースの比較を行った結果を表1に示す。アクセラレータを搭載した場合、非搭載の場合と比較して LUT の使用数は約 46,000 個 (55.3%) 増加した。また、DSP は約 2 倍、FF と BRAM は約 1.2 倍となった。

SoC のタイトルごとの LUT 使用数を表2に示す。アクセラレータを搭載した場合、アクセラレータタイトルだけでなく、他のすべてのタイトルで LUT 使用数が増加した。

表1 使用リソースとその使用率

	LUT	FF	BRAM	DSP
Acc 搭載 SoC	129,191 (42.5%)	102,580 (16.8%)	85 (8.2%)	48 (1.7%)
Acc 非搭載 SoC	83,179 (27.3%)	83,906 (13.8%)	68 (6.6%)	27 (0.9%)

表2 タイル毎の LUT 使用数

	CPU	ACC	MEM	I/O	empty
Acc 搭載 SoC	62,163	22,681	20,840	23,507	-
Acc 非搭載 SoC	51,136	-	11,364	13,344	7,363

6. まとめ

本稿では、SoC 設計プラットフォーム ESP を用いたプロセッサ開発環境の構築と、それを用いて積和演算アクセラレータ搭載 SoC の設計を行った。Xilinx VC707 をターゲットとして合成を行い、アクセラレータを搭載した場合と搭載しなかった場合でリソース使用量の比較評価を行った。実験を通して、ESP を用いることにより SoC のシステムレベル設計が容易に行えることを確認した。今後はターゲットボードへの実装と動作検証を行う。さらに、アクセラレータを搭載した場合と搭載しなかった場合で処理時間の比較評価も行う。

参考文献

- [1] <https://riscv.org/>
- [2] P. Mantovani, D. Giri, G. D. Guglielmo, L. Piccolboni, J. Zuckerman, E. G. Cota, M. Petracca, C. Pilato, and L. P. Carloni, "Agile SoC Development with Open ESP," in Proc. International Conference on Computer Aided Design (ICCAD), Sept. 2020
- [3] D. Giri, P. Mantovani, and L. P. Carloni, "Runtime Reconfigurable Memory Hierarchy in Embedded Scalable Platforms," in Proc. Asia and South Pacific Design Automation Conference (ASPAC), Jan. 2019, https://sld.cs.columbia.edu/pubs/giri_aspdac19.slides.pdf.
- [4] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 11, pp. 2629-2640, Nov. 2019.
- [5] <https://gaisler.com/>