USING TRANSPORT TRIGGERED ARCHITECTURES FOR APPLICATION SPECIFIC PROCESSOR DESIGN

Petr Mikušek
Master Degree Programme (2), FIT BUT
E-mail: xmikus01@stud.fit.vutbr.cz

Supervised by: Tomáš Martínek
E-mail: martinto@fit.vutbr.cz

ABSTRACT

This paper deals with searching of processor architecture template suitable for automatized design of application-specific processors for embedded systems. The aim is to design a processor generator which is able to generate processor according to architecture description. Transport Triggered Architectures (TTAs) seams to be regular, flexible and scalable compared to superscalar and VLIW architectures. The programming model of TTAs is mirrored when compared with regular RISC and VLIW architectures; instead of programming operations which cause data transports as side effects, in TTAs the transports are programmed, where transport may trigger an operation if necessary.

1. ÚVOD

Procesory hrají důležitou roli ve všech oblastech lidského života. Můžeme je najít nejen v osobních počítačích, ale především ve vestavěných zařízeních (např. v mobilních telefonech, spotřební elektronice, osobních automobilech nebo semaforech), kterými je člověk v dnešní době doslova obklopen. Ačkoliv zhruba polovina všech vynaložených prostředků na procesory připadá na použití v oblasti osobních počítačů, počet těchto procesorů tvoří pouhý zlomek všech prodaných procesorů. Zbytek připadá právě na procesory ve vestavěných systémech, takže si zasluhují naši pozornost.

Vestavěné systémy jsou navrženy pro vykonávání určitého specifického úkolu. Pro realizaci tohoto úkolu můžeme použít běžně dostupný univerzální procesor, ale ten nemusí vždy vyhovovat požadavkům na výkon, cenu a spotřebu kladených na konkrétní aplikaci. Řešením je použití aplikačně-specifického procesoru, který je po všech stránkách uzpůsoben pro optimální splnění požadovaných kritérií. Problémem však zůstává, jak snadno a rychle daný procesor navrhnout a implementovat, neboť disciplína návrhu procesorů je poměrně komplikovaná a vyžaduje zkušeného návrháře. Zautomatizování tohoto procesu by výrazně zkrátilo dobu návrhu a tím i dobu uvedení výrobku na trh.

Tato práce si klade za cíl najít a popsat vhodnou šablonu architektury procesorů, která je dostatečně pravidelná, flexibilní a škálovatelná, a tím pádem vhodná pro implementaci aplikačně-specifických procesorů. Dále je nutné navrhnout a implementovat generátor procesorů, který na základě popisu funkčních jednotek, jejich propojení a parametrů architektury automaticky vygeneruje optimální implementaci procesoru.

2. SOUČASNÉ ARCHITEKTURY VESTAVĚNÝCH PROCESORŮ

Možnost překrytí zahájení a provedení několika instrukcí v rámci jednoho procesoru se nazývá paralelizmus na úrovni instrukcí (ILP). Využívání ILP se stalo důležitou metodou zvyšování výkonu nejnovějších procesorů.

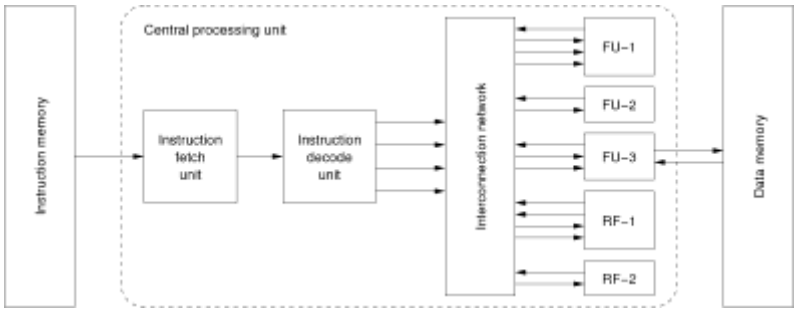
V superskalárních procesorech (např. Intel Pentium) hardware procesoru detekuje a řeší závislosti mezi operacemi v sekvenčním proudu instrukcí za běhu. Tento přístup poskytuje binární kompatibilitu s předchozími generacemi architektur, ale trpí složitostí hardwaru (hlavně použitím asociativních pamětí) a dlouhou dobu návrhu, což dělá superskalární architektury nezajímavé z pohledu vestavěných systémů.

Procesory s velmi dlouhým instrukčním slovem (VLIW) vydávají pouze jednu instrukci za takt. Tato jedna dlouhá instrukce sestává z několika RISCových operací, které se vykonávají současně. Datové závislosti jsou řešeny v době kompilace pomocí sofistikovaného kompilátoru provádějícího plánování instrukcí. Ukazuje se však, že VLIW procesory nejsou dostatečně škálovatelné [2]. Se zapojením každé další funkční jednotky je potřeba přidat potřebné čtecí a zápisové porty do souboru registrů. Navíc je nutné rozšířit předávací síť, která zajišťuje přivedení výsledku jedné jednotky na vstup druhé jednotky v případě datového konfliktu. Obecně jsou propojeny výstupy všech jednotek se všemi vstupy a navíc jsou připojeny výstupy ze souboru registrů a přímé operandy, což vede na rozsáhlou propojovací síť typu křížový přepínač. Většina zdrojů na čipu je tak použita na propojení a nikoliv na užitečný výpočet.

Klasické architektury procesorů se dají zařadit mezi tzv. *operací spouštěné architektury* (OTA, Operation Triggered Architectures). Tyto architektury se programují uvedením požadovaných operací, na základě kterých dojde k přenosům dat jako vedlejší efekt. Na opačném principu pracují tzv. *přenosem spouštěné architektury* (TTA, Transport Triggered Architectures) [1], tzn. že se programují uvedením přenosů dat, které spouští operace jako vedlejší efekt. V [2] je ukázáno, že TTA architektury redukují složitost předávací sítě a požadavky na počet portů souboru registrů. Složitost hardwaru je vyměněna za větší úsilí kompilátoru, který vidí veškeré architekturní registry a může tak provádět komplexní plánování instrukcí a optimalizace staticky v době kompilace.

3. PŘENOSEM SPOUŠTĚNÉ ARCHITEKTURY

Na obrázku 1 vidíme architekturu TTA procesoru. Skládá se z libovolného množství funkčních jednotek (FU) a souboru registrů (RF), které mohou mít různý počet vstupních a výstupních portů. Jednotky jsou propojeny mezi sebou pomocí propojovací sítě. Načítání instrukcí z instrukční paměti provádí jednotka načítání instrukcí. Dekódování načtené instrukce má na starosti jednotka dekódování instrukce, která zároveň řídí propojovací síť.



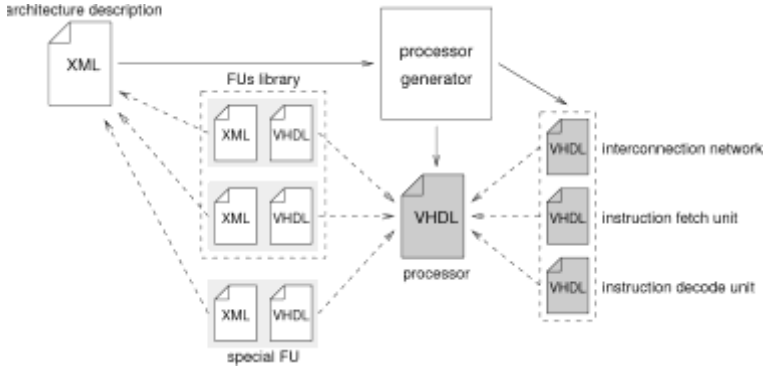
Obrázek 1: Architektura TTA procesoru

Propojovací síť je tvořena několika sběrnicemi a umožňuje tak provádět několik přenosů dat zároveň. Funkční jednotky a soubory registrů se připojují k propojovací síti pomocí jedné nebo více vstupních nebo výstupních zásuvek. Vstupní zásuvky obsahují multiplexory, výstupní demultiplexory. Propojovací síť může být plně propojená nebo částečně propojená.

Každá funkční jednotka a soubor registrů obsahuje na svém vstupu a výstupu registry. Tyto registry se dělí do třech kategorií: *operandový registr* slouží k nahrání požadovaných operandů, *spouštěcí registr* nahrává poslední operand a spouští danou operaci, a konečně *výsledkový registr* obsahuje po dokončení operace výsledek.

4. GENERÁTOR PROCESORU

Na obrázku 2 je naznačen postup při generování procesoru. Popis každé FU je rozdělen do dvou souborů. Vlastní HW realizace je popsána ve VHDL souboru, ke kterému přísluší XML soubor popisující rozhraní a časování jednotky. Soubor XML popisující architekturu udává, jaké jednotky jsou v procesoru použity a jakým způsobem jsou propojeny. Tento soubor je vstupem do generátoru procesorů, který na jeho základě vygeneruje VHDL soubory s implementací propojovací sítě, jednotek načítání a dekódování instrukcí a také hlavní VHDL soubor popisující procesor, ve kterém jsou instancovány funkční jednotky a soubory registrů z knihovny a vygenerované jednotky.



Obrázek 2: Postup při generování TTA procesoru

5. ZÁVĚR

V tomto článku byly prezentovány přenosem spouštěné architektury jako vhodná šablona architektury pro aplikačně-specifické procesory. Tato šablona je díky své pravidelnosti, flexibilitě a škálovatelnosti vhodná pro automatizovaný návrh. K tomu slouží představený generátor procesorů, který na základě popisu funkčních jednotek, jejich propojení a parametrů architektury automaticky vygeneruje optimální implementaci procesoru.

LITERATURA

[1] Corporaal, H., Mulder, H.: MOVE: a framework for high-performance processor design. In: Proceedings of the 1991 ACM/IEEE conference of Supercomputing. ACM Press, New York 1991, s. 692-701

[2] Corporaal, H.: TTAs: Missing the ILP Complexity Wall. Journal of System Architecture, 1999, sv. 45, č. 12/13, s. 949-973