ARCHITEKTURY ČÍSLICOVÝCH SYSTÉMŮ VYUŽÍVAJÍCÍ PRINCIP SAMOADAPTACE

Lukáš Kohout
Informatika a výpočetní technika, 1. ročník kombinovaného studia (1. semestr)
Školitel: Ing. Martin Daněk, Ph.D.

Ústav teorie informace a automatizace AV ČR, v.v.i.
Pod Vodárenskou věží 4, 182 08 Praha 8

kohoutl@utia.cas.cz

Abstrakt. Článek se zabývá samoadaptací v číslicových obvodech. V textu je uvedena definice konceptu SANE, který samoadaptaci přímo využívá, a základní popis modelu SVP, který řeší reprezentaci výpočtu v konceptu SANE. Jsou zde uvedeny dostupné platformy pro implementaci SANE konceptu. Jednou z nich je UTIA HW platforma, ta byla rozšířena o plánovač úloh, a byly s ní provedeny první experimenty.

Klíčová slova. samoadaptace, pervasive computing, SANE, SVP, micro-thread, FPGA, MicroBlaze, PicoBlaze, UTIA HW platforma, plánovač úloh, částečná runtime rekonfigurace.

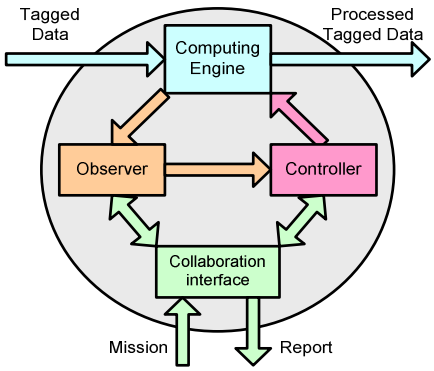
1 Úvod

V několika posledních letech došlo v mnoha oblastech k velkému růstu využívání výpočetní techniky, a ta se tak stala, takříkajíc, všudypřítomnou. Potřeba využívání výpočetních prostředků se mění spolu s individuálními potřebami každého uživatele v závislosti na okolí, ve kterém se uživatel pohybuje (budovy, automobily, atd.). Tyto výpočetní prostředky jsou často jednoúčelové a podporují jen omezený počet funkcí (vestavěné systémy v nejrůznějších přehrávačích multimediálního obsahu, mobilní telefony, atd.), a přesto se stávají stále složitějšími a komplexnějšími.

S neustálým růstem počtu výpočetních prostředků vyvstává otázka, zda by nebylo vhodné využít dané prostředky i k řešení nějakého komplexnějšího problému v okamžiku, kdy nejsou využívány k jejich prvotní funkci. Dostáváme se tak k anglickému termínu **pervasive computing**, který nemá český ekvivalent. Spolu se zavedením pojmu pervasive computing vzniká potřeba najít mechanizmus, který by umožňoval přistupovat k měnícímu se počtu nesourodých zařízeních v nějakém výpočetním prostředí. Možným řešením je princip **samoadaptace** (self-adaptivity), tedy přizpůsobení se novému vzniklému stavu v nějaké výpočetním prostředí.

Text je koncipován jako studium daného problému a popis prvních pokusů. V první části je uveden **SANE koncept** [2], který se snaží odpovědět na to, co samoadaptace je a jak jí použít. Dále je uveden základní popis **SVP modelu** [1], který definuje reprezentaci výpočtu v konceptu SANE. Další část představuje vývojové platformy, které by byly použitelné pro implementaci SANE konceptu. Jedna z nich (**UTIA HW platforma** [6]) byla použita k prvním experimentům, které jsou v další části textu prezentovány. Na závěr jsou shrnuty zatím dosažené výsledky.

Problém někdy neexistující české terminologie je řešen, snad pro čtenáře nejméně násilným způsobem, použitím termínů anglických a jejich běžně používaných českých odvozenin.



Obrázek 1: Základní pohled na SANE [2].

2 SANE koncept

SANE (Self-Adaptive Networked Entity [2], obrázek 1) je nezávislá výpočetní jednotka, která je autonomní, a která má lokální monitorování a řízení vlastního procesu. SANE koncept nespecifikuje implementaci (nezáleží na tom zda bude hardwarová nebo softwarová), pouze zapouzdřuje vlastnost samoadaptace a možnosti spolupráce více jednotek. Unifikuje pohled na jinak heterogenní výpočetní jednotky, což usnadňuje jejich řízení jako celku.

Z pohledu uživatele se přestává mluvit o programu a zavádí se pojem **mise** (mission). Uživatele už nezajímá, jakým způsobem k výsledku řešené úlohy dojde, ale jenom výsledek samotný. Jinými slovy, uživatel řekne jaká jsou data a jaká operace se s nimi má provést, ale už ho nezajímá jak bude požadovaná operace provedena (hardware vs. software).

SANE koncept je založen na třech mechanizmech:

- Vnitřní uzavřené řídicí smyčce, která poskytuje autonomní chování a přizpůsobení se vnějšímu okolí. Tomuto říkáme **vnitřní samoadaptace**.
- Schopnost sdílet informace mezi různými výpočetními jednotkami systému s cílem dosáhnout požadovaného výsledku řešené úlohy. To nazýváme **samoadaptací při spolupráci**.
- Počítání s daty, které si s sebou nesou informaci o tom, co obsahují a jaká operace se s nimi má provést. Zde budeme používat termín **tagované počítání** (tagged computing) a **tagovaná data** (tagged data).

2.1 Vnitřní samoadaptace

Vnitřní samoadaptace slouží k vnitřnímu sledování a řízení výpočetního procesu, nezáleží přitom na způsobu vykonání. S ohledem na co nejlepší vykonání příslušné mise je sledováno také okolí výpočetní jednotky. Jsou dva základní typy adaptace, které odpovídají tomuto modelu [3]:

- Adaptace parametrů je nejjednodušší forma adaptace. Spočívá pouze v přizpůsobení nějakých parametrů výpočetní jednotky bez jakékoliv změny vlastní struktury. Například změna taktovací frekvence nebo napájecího napětí v hardwarové implementaci, změna koeficientů číslicového filtru či změna programu v softwarové implementaci.
- Adaptace struktury je změna struktury výpočetní jednotky, jejích jednotlivých částí a propojení těchto částí. V případě hardwaru je vhodným kandidátem runtime rekonfigurace FPGA obvodu, pro software by to mohla být změna propojení softwarových bloků.

Adaptace se provádí za běhu samoadaptivního systému, jednotlivé konfigurace však mohou být připraveny dopředu. Například několik různých bitstreamů pro FPGA nebo několik různých zkompilovaných programů pro standardní procesor.

2.2 Samoadaptace při spolupráci

Systém postavený ze SANE jako základních stavebních kamenů je v podstatě systém distribuovaný. To znamená, že každý jeden prvek systému je nezávislý a má své lokální chování. Lze ho ale ovlivňovat z venku stavem jeho okolí. Toto umožňuje, aby obecně různé prvky systému (hardware, software, různé technologie, atd.) spolupracovaly na nějaké složitější úloze.

2.3 Tagované počítání

Tagované počítání spočívá v přidání informace k datům o tom, co data znamenají a jaká operace se má nad nimi provést. Koncept tagování není žádná novinka a byl již mnohokrát použit dříve. Například při zpracování velikého množství dat jako jsou databáze, vyhledávače na webu i webové stránky samotné.

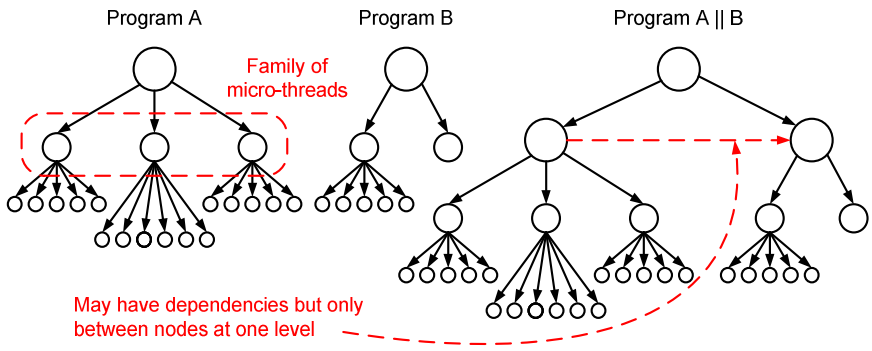
Tagování se zdá být pro samoadaptaci velmi důležité, obzvláště pokud v souvislosti s ní mluvíme o pervasive computing. Přidáním informace o významu dat mluvíme o **samo-identifikovatelných datech** (self-identifying data, [4]). Přirozeným důsledkem je, že pokud výpočetní systém zná význam dat, pak s nimy může pracovat efektivněji a po zpracování dat tag změnit s ohledem na to, která data byla zpracována a případně co se ještě s daty má provést (jiná výpočetní jednotka). Tag vlastně přidává k datům informaci o kontextu, což výpočetnímu systému může pomoci k adaptaci a rychlejšímu dosažení požadovaného výsledku.

Spojení všech těchto myšlenek vede k tomu, že každá výpočetní jednotka umí nějakou sadu operací. Podle informace z tagu data buď zpracuje nebo jen propustí. Pokud by nastala situace kdyby po delší dobu přicházela data, která neumí zpracovat (výpočetní jednotka by byla nevyužita), byla by možnost tuto sadu operací nahradit jinou sadou operací. V nejjednodušším případě by se mohlo jednat o změnu programu v klasickém procesoru. Další možnosti nabízí částečná runtime rekonfigurace v FPGA obvodech.

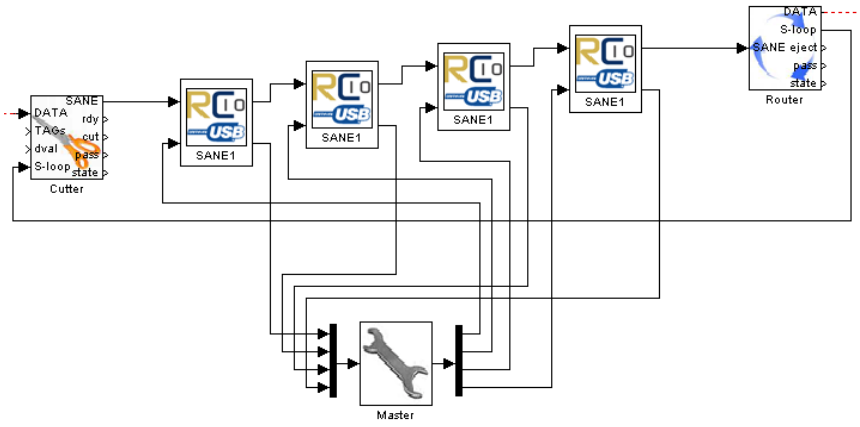
3 SVP model

Koncept SANE říká jak mají vypadat výpočetní jednotky, aby systém z nich postavený byl samoadaptivní. Je zřejmé, že i když samotné výpočetní jednotky mohou být implementovány libovolnou technologií, tak jejich komunikační rozhraní, které zajišťuje spolupráci, musí být nějakým způsobem unifikované. Takové rozhraní, a způsob reprezentace výpočtu v takovém systému, specifikuje **SVP model** (SANE Virtual Processor, [1]).

SVP model pracuje na principu rozbalování smyček algoritmu na **micro-thready (**micro-threads**)**. Je modelem souběžného výpočtu (concurrent computation) a je definovaný jako sada instrukcí nebo operací ve výpočetní jednotce nad micro-thready (operace *create*, *kill*, *squeeze*, *break* a *synch*). Micro-thready mohou být dále seskupována do rodin micro-threadů (families of micro-threads), jedné



Obrázek 2: SVP model [1].



Obrázek 3: Koncept SANE postavený z vývojových desek RC10 v prostředí Matlab/Simulink.

výpočetní jednotce náleží jedna rodina micro-threadů. Při souběžném výpočtu je potřeba, aby u programů, které jsou vykonávány souběžně, nedošlo k uváznutí (aby byly *deadlock free*). Tento požadavek SVP model splňuje [1]. Uvedené vlastnosti SVP modelu zobrazuje obrázek 2, detailnější popis SVP modelu přesahuje rámec tohoto textu a nebude dále rozebírán.

4 Přehled platforem pro implementaci SANE konceptu

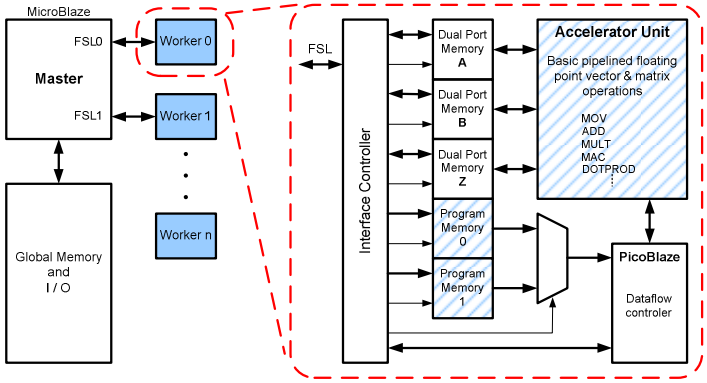
Zatím se neví, jak SANE koncept implementovat ani jak má vypadat topologie takového systému. Zde budou ve zkratce popsány dvě základní platformy, které vznikly na půdě ÚTIA AV ČR, v.v.i, Oddělení zpracování signálu, a ze kterých lze vyjít.

4.1 SANE model v prostředí Matlab/Simulink

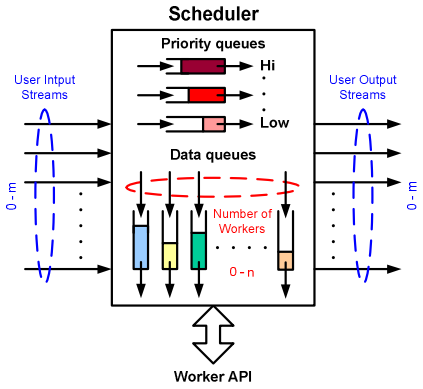
První pokusy byly provedeny s topologií logický kruh [5], systém byl postaven z několika vývojových desek RC10 firmy Celoxica Ltd [7] propojených pomocí USB. Každá jedna deska RC10 obsahovala jeden SANE. Systém byl vyvinut v prostředí Matlab/Simulink firmy MathWorks, Inc, pomocí kterého je potom i ovládán. Obrázek 3 zobrazuje blokové schéma systému, ze kterého je zřejmé, že Matlab/Simulink je použit k přípravě vstupních dat a sběru výsledků (data jsou tagovaná). Každý jeden SANE může být softwarový blok v prostředí Matlab/Simulink, ale i reálný hardware. Pokud se jedná o hardware (vývojová deska RC10), tak je z prostředí Matlab/Simulink spustitelný a krokovatelný (*hardware in the loop*).

4.2 UTIA HW platforma

UTIA HW platforma [6], je systém typu *master-worker*. *Master* je procesor MicroBlaze [9], *worker* je nezávislá řídící jednotka (procesor PicoBlaze [10]) a akcelerátor (například jednotka pro výpočet v pohyblivé řádové čárce). Řídící jednotka má k dispozici dvě paměti programu, mezi kterými lze přepínat. Obsah pamětí programu lze kdykoliv za běhu přepsat novým programem. Akcelerátor je připojen přes tři nezávislé dvouportové paměti, dvě jsou typicky použity pro dva operandy, a třetí na výsledky. Dvouportovosti těchto pamětí je využíváno k zajištění nepřetržitého přísunu nových dat a *worker* tak mohl pracovat efektivněji. Paměť je logicky rozdělena na dva bloky, jeden logický blok je plněn daty z vnějšku a z druhého logického bloku se čtou aktuální data. Mezi těmito logickými bloky se přepíná (*ping pong buffer*). Přenos dat se provádí dávkově po blocích. *Worker* jako celek je připojen k *masteru* pomocí FSL (Fast Simplex Link [9]). Obrázek 4 zobrazuje blokové schéma základní konfigurace UTIA HW platformy, vyšrafované části poskytují vnitřní samoadaptaci. Bloky *Program Memory 0* a *Program Memory 1* nabízejí samoadaptaci parametrů, blok *Accelerator Unit* samoadaptaci struktury (částečná runtime rekonfigurace). Složitější konfigurace UTIA HW platformy



Obrázek 4: Blokové schéma UTIA HW platformy.



Obrázek 5: Plánovač úluh v UTIA HW Platformě.

mohou být typu SIMD, mohou obsahovat více bloků *Accelerator Unit* (2, 4, 8), čehož by šlo využít při implementaci SVP modelu. Jeden *worker* by pak mohl představovat jednu rodinu micro-vláken.

5 Samoadaptace v závislosti na počtu vstupních toků dat

Mějme systém, ve kterém je počet výpočetních jednotek proměnný, a který má zároveň zpracovat proměnný počet nezávislých vstupních toků dat. Takovým systémem může být například UTIA HW platforma (kapitola 4.2). Automatické plánování úloh tohoto systému, by pak také bylo určitou formou samoadaptace.

UTIA HW platforma byla použita v její jednodušší podobě, systém využívá čtyři *workery*, kde každý provádí výpočty v pohyblivé řádové čárce. Implementace byla provedena na vývojové desce ML402 [8] firmy Xilinx, Inc. Platforma byla dále rozšířena o plánovač úloh, který mapuje proměnný počet vstupních toků dat na proměnný počet výpočetních jednotek. Tím vznikla další úroveň abstrakce, která před uživatelem schovává reálný počet výpočetních jednotek. Plánovač je doplněn o podporu priorit vstupních toků dat, a lze ho proto využít například k zajištění požadované kvality služby (QoS - Quality of Service). Princip plánovače úloh v UTIA HW platformě je naznačen na obrázku 5. Proměnný počet výpočetních jednotek je simulován DIP přepínačem na desce ML402.

Implementovaná funkce je FIR filtr řádu 250 v aritmetice s pohyblivou řádovou čárkou (single precision). Pokud jsou za běhu odebírány nebo naopak přidávány jednotlivé výpočetní jednotky (DIP přepínačem), pak je ovlivněn čas výpočtu. Tabulka 1 shrnuje dosažené zrychlení systému při taktovací frekvenci 100 MHz vůči samotnému procesoru MicroBlaze s připojenou hardwarovou jednotkou pro výpočty v pohyblivé řádové čárce. Zrychlení je změřeno pro všechny konfigurace systému, tedy postupně pro jeden, pro dva, pro tři a nakonec pro čtyři *workery*. Zrychlení je dosaženo dávkovým zpracováním, data jsou držena lokálně uvnitř *workera* a lze tak výrazně efektivněji využít proudově pracující jednotku v pohyblivé řádové čárce.

Tabulka 1: Zrychlení systému pro všechny konfigurace vůči samotnému procesoru MicroBlaze s hardwarovou jednotkou pro výpočty v pohyblivé řádové čárce (FPU – Floating Point Unit).

	Čas výpočtu [µs]	**Počet FP operací [-]**	**Propustnost [FP op/s]**	**Zrychlení [-]**
MicroBlaze+HW FPU	41 787 715	250 000 000	5 982 619	1,00
1 *worker* + plánovač	1 922 296	250 000 000	130 052 811	21,74
2 *workeři* + plánovač	965 955	250 000 000	258 811 228	43,31
3 *workeři* + plánovač	725 525	250 000 000	344 578 064	57,60
4 *workeři* + plánovač	486 340	250 000 000	514 043 673	85,92

6 Závěr

Tato práce prezentuje koncept SANE a SVP model jako formální modely pro samoadaptaci výpočetního systému, kde přestává být důležitý, z pohledu uživatele, způsob zpracování dat a začíná být upřednostňován výsledek samotný bez ohledu na to, jak se k němu došlo. V textu jsou popsány první pokusy o samoadaptaci s využitím FPGA technologie. K experimentu byla použita UTIA HW platforma, která je implementací SANE konceptu. UTIA HW platforma byla dále rozšířená o plánovač úloh, který mapuje proměnný počet vstupních toků dat na proměnný počet výpočetních jednotek, čímž byly možnosti samoadaptace ještě více rozšířeny. Testovací úlohou byl FIR filtr řádu 250 v pohyblivé řádové čárce a bylo při ní dosaženo výsledku zhruba 85 krát rychleji, než kdyby stejnou úlohu počítal procesor MicroBlaze spolu s jednotkou pro výpočet v pohyblivé řádové čárce (platí pro taktovací frekvenci 100 MHz a čtyři *workery*).

Budoucí práce se bude zabývat implementací hardwarové podpory SVP modelu, který byl zatím ověřen simulací pomocí pthreadů na standardním PC. Půjde o rozšíření instrukční sady procesoru Leon 3 [11] o operace specifikované v SVP modelu (kapitola 3, [1]).

Poděkování

Tento text vznikl za podpory projektu Evropské komise, projekt ÆTHER číslo FP6-IST-027611 a projektu CAK II číslo 1M0567 na pracovišti ÚTIA AV ČR, v.v.i, Oddělení zpracování signálu.

Literatura

[1] Jesshope, C. Diguet, J-P. Paulsson, K.: ÆTHER Deliverable D2.1.1: First annual report on abstract SANE processors and system environment, 2007.

[2] Philippe, J-M. Paulsson, K. Becker, J. Hübner, M. Danek, M. Bartosinski, R. Honzik, P. Kadlec, J. Pohl, Z. Moreno, J, M. Madrenas, J. Cabestany, J. Casas, J, A. Pozzi, L. Koutsomitsos, S.: ÆTHER Deliverable D1.1.1: First annual report on SANE hardware architecture, 2007.

[3] Bisdounis, L. Ferrante, A. Taddeo, A. Derin, O. Bonnot, P. Koutsomitsos, S. Paulsson, K. Philippe, J-M. Danek, M.: ÆTHER Deliverable D4.1.2 : Applicative Scenarios and Basic Requirements. Metrice Definition.

[4] Feustel, E. A. "On the Advantages of Tagged Architectures," IEEE Transactions on Computers, vol. 22, pp. 644-652, 1973.

[5] Bartosinski, R. Daněk, M. Honzík, P. Kadlec, J.: Modelling Self-Adaptive Networked Entities in Matlab/Simulink. In Technical Computing Prague 2007. Praha : Humusoft, 2007. S. 1-8. ISBN 978-80-7080-658-6. Technical Computing Prague 2007, Praha, 14.11.2007-14.11.2007, CZ.

[6] Kadlec, J. Bartosinski, R. Danek, M.: "Accelerating MicroBlaze floating point operations," Field Programmable Logic and Applications, 2007. FPL 2007. pp. 621–624, 27-29 Aug.

[7] Celoxica Ltd.: RC10 Manual, Platform Developer's Kit. www.celoxica.com [online]. www28.cs.kobe-u.ac.jp/~kawapy/class/proj/proj07/RC10Manual.pdf.

[8] Xilinx, Inc.: ML401/ML402/ML403 Evaluation Platform, User Guide, v 2.5. www.xilinx.com [online]. www.xilinx.com/support/documentation/boards_and_kits/ug080.pdf, May 24, 2006.

[9] Xilinx, Inc.: MicroBlaze Processor Reference Guide, Embedded Development Kit 9.1i, www.xilinx.com [online]. www.xilinx.com/support/documentation/sw_manuals/edk91i_mb_ref_guide.pdf, September 15, 2006.

[10] Chapman, K.: PicoBlaze 8-bit Embedded Microcontroller User Guide, Xilinx UG129, v 1.1.1, November 21, 2005.

[11] Gaisler, J. Habinc, J. Catovic, E.: GRLIB IP Library User's Manual, v 1.0.18. www.gaisler.com/cms [online]. www.gaisler.com/products/grlib/grlib.pdf.