Univerzita Pardubice

Fakulta ekonomicko-správní

Modelování práce procesorů v multiprocesorovém operačním systému

Bc. Jan Fila

Diplomová práce

2011

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jan FILA**
Osobní číslo:	**E090484**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Modelování práce procesorů v multiprocesorovém operačním systému**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

Osvojení Petri sítě jako modelového nástroje.
Spolupráce procesorů v multiprocesovém operačním systému s pamětí.
Spolupráce jader ve vícejádrovém procesu s pamětí.
Modelování této spolupráce pomocí Petri sítě.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] MARKL, Jaroslav. Petriho sítě [online]. Ostrava : VŠB - Technická univerzita Ostrava , 1998 [cit. 2010-06-24]. Dostupné z WWW: <http://www.cs.vsb.cz/markl/pn/>.

[2] SILBERSCHATZ, Abraham ; GALVIN, Peter Baer; GAGNE, Greg. Operating systems concepts. John Wiley, 2003. 978 s.

[3] STALLINGS, William. Operational Systems. Prentice Hall, 2005. 822 s.

[4] TANENBAUM, Andrew Stuart. Modern operating Systems. Prentice Hall, 1992. 728 s.

[5] VORÁČKOVÁ, Šárka; PĚNIČKA, Martin ; VESELÝ, Jaroslav. Úvod do modelování procesů Petriho sítěmi. Vyd. 1. Praha : ČVUT v Praze, 2008. 126 s.

Vedoucí diplomové práce: **prof. Ing. Jan Čapek, CSc.**
Ústav systémového inženýrství a informatiky

Datum zadání diplomové práce: **4. října 2010**

Termín odevzdání diplomové práce: **6. května 2011**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.D.
vedoucí ústavu

V Pardubicích dne 4. října 2010

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury. Byl jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 2.5.2011

Fila Jan

PODĚKOVÁNÍ

Touto cestou bych rád poděkoval panu prof. Ing. Janu Čapkovi, CSc. za jeho cenné rady poskytnuté při konzultačních hodinách.

SOUHRN

Diplomová práce je zaměřena na popis paralelních systémů a modelování spolupráce SMP se sdílenou pamětí. Součástí práce je osvojení Petriho sítí jako modelovacího jazyka. Její hlavním cílem je namodelovat spolupráci SMP (Symmetric multiprocessing) se sdílenou pamětí pomocí nástroje HPSim. SMP jsou modelovány pomocí propojovacích sítí sběrnice a křížového přepínače. Následně jsou obě architektury porovnány a zjištěny jejich hraniční možnosti.

KLÍČOVÁ SLOVA

Paralelní systémy, SMP, sdílená paměť, HPSim

TITLE

Modeling work of processors in a multiprocessor operating system

ABSTRACT

The thesis focuses on the characterization and modeling of parallel systems co SMP shared memory. A part of the adoption of Petri nets as a modeling language. Its main aim is to model cooperation SMP shared memory using HPSim. SMP (Symmetric multiprocessing) are modeled using interconnection networks bus and crossbar. Subsequently, the two architectures were compared and found their border possibilities.

KEYWORDS

Parallel systems, SMP, shared memory, HPSim

Obsah

Úvod

Od existence číslicových počítačů byla snaha zvyšovat výpočetní kapacity. Zvyšování frekvence procesorů a miniaturizace tranzistorů na čipu, tlačí tento trend k fyzickým limitům výroby. Právě výroba procesorů na křemíkovém základu dosáhne v bližší době svého konce.

„Počet tranzistorů v jednom integrovaném obvodu se zdvojnásobí každý rok při zachování stejné ceny.“

Gordon Moore

Přes neustále zvyšování výkonu procesorů se vždy najde rozsáhlá aplikace, kde počítače s jedním procesorem nestačí na jejich zpracování. Proto už od vzniku výpočetní techniky byly vedeny myšlenky jak problémy s výkonností vyřešit.

Rostoucí trend využívání paralelních systémů je nevyhnutelný v mnoha odvětvích. Příkladem může být první dvoujádrový smartphone od firmy LG, který byl představen v lednu letošního roku.

Cílem práce je namodelovat spolupráci SMP (Symmetric multiprocessing) se sdílenou pamětí pomocí Petriho sítí. SMP jsou modelovány pomocí propojovacích sítí sběrnice a křížového přepínače. Tyto architektury jsou následně analyzovány a zjištěny jejich hraniční možnosti.

V první kapitole je uvedeno rozdělení paralelních systémů podle různých kritérií, pomocí nichž lze na daný paralelismus nahlížet. Tato část je především zaměřena na klasifikaci architektur. Závěrečná část první kapitoly je věnována možnostem odvětví paralelních systémů. Druhá kapitola je zaměřena na Petriho sítě. V této části je popisována grafická reprezentace a charakteristika Petriho sítí s jednotlivými příklady. Třetí část je zaměřena na modelování spolupráce procesorů se sdílenou pamětí v prostředí HPSim. V této části jsou uvedeny dva modely a to spolupráce s využitím sběrnici a křížového přepínače. V předposlední části je zkoumána komunikační režie dle propojovací sítě (sběrnice, křížový přepínač). Poslední kapitola se zaměřuje na hraniční možnosti namodelovaných architektur a následné porovnání.

1 Úvod do paralelních systémů

Na začátku této kapitoly bude vymezeno, co znamená pojem paralelní systém. Paralelní systém označujeme za takový systém, kde současně může běžet několik procesů. [6]

1.1 Úrovně paralelního zpracování

V rámci paralelního zpracování je proces chápan v několika úrovních složitosti. Tyto úrovně jsou označovány jako úrovně granularity (zrnitosti) procesu. Nejjemnější granularita je označena 1, nejhrubší 5. [6]

1. Příkazy, instrukce
2. Cykly, interakce
3. Podprogramy
4. Části úloh a programů
5. Nezávislé úlohy a programy

Na nejnižší úrovni, to znamená na úrovni instrukcí, se jedná převážně o superskalární režim, který umožňuje zpracovat několik instrukcí z paměti pomocí více jednotek ALU (Aritmetic Logic Unit, dále jen ALU), které tvoří jediný procesor. Druhá úroveň (cykly, interakce) se někdy označuje jako vektorizace. Vektorové počítače dokážou nejlépe pracovat s řadou (vektorem) hodnot najednou. Na třetí úrovni granularity (podprogramů) se jedná o rozložení jednotlivých podprogramů, které jsou vzájemně zpracovány ve výpočetních jednotkách. Na nejvyšších úrovních 4 a 5 se jedná o samostatné a nezávislé zpracování jednotlivých úloh ve víceprocesorových systémech. Je nutné dodat, že vývoj postupoval od nejnižší až po nejvyšší úroveň. V této práci je hovořeno pouze o zpracování nezávislých úloh a programů. [6]

1.2 Klasifikace paralelních systémů

Od vzniku číslicových počítačů se zaváděly různé klasifikace pro lepší dorozumění mezi projektanty a uživateli výpočetních systémů. Paralelní systémy nejsou

výjimkou, a proto před 40 lety vznikla tzv. **Flynnova taxonomie**. V této části budou paralelní systémy děleny dle zmíněné taxonomie a dle **uspořádání paměti**.

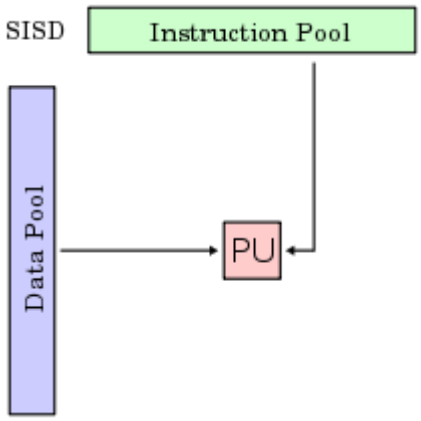
1.2.1 Flynnova taxonomie

Flynnova klasifikace byla navržena v roce 1966 americkým profesorem M.J.Flynnem. Flynnova klasifikace rozlišuje multi-procesorové počítače dle zpracování počtu **instrukčních a datových toků**, které lze zpracovat v jeden okamžik. Instrukce je chápana jako sekvence instrukcí nižší úrovně, kterou vykonává výpočetní jednotka (procesor). Datový proud je chápan jako výměna dat mezi pamětí a procesorem. Flynn definoval kategorie do následujících skupin [1]:

- SISD (Single Instruction Single Data)
- SIMD (Single Instruction Multiple Data)
- MISD (Multiple Instruction Single Data)
- MIMD (Multiple Instruction Multiple Data)

SISD

Počítač kategorie SISD zpracovává data sériově podle jednoho programu. Typickým představitelem je uváděn počítač typu von Neumana (např. IBM 360). Schéma SISD je ilustrován na obr. 1. Architekturu dle von Neumana nelze považovat za paralelní systém, ale podle kompletního dělení systémů ji Flynn zařadil do své klasifikace. [3]

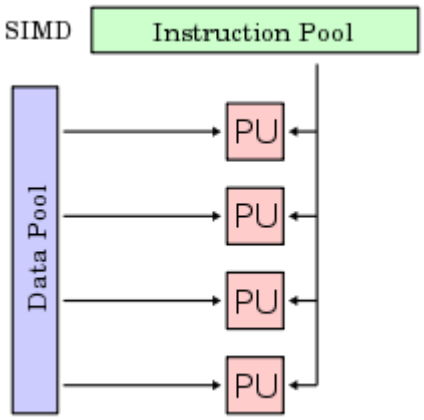


Obrázek 1: Schéma SISD. Zdroj [15].

Z obrázku je vidět, že procesní jednotka (PU – processor unit) vyjadřuje samostatnou funkční jednotku, která obsahuje řídící a aritmeticko-logickou jednotku. Tato jednotka pak vykonává instrukce nad daty. Například sečtení čísel c=a+b, kde PU provádí c=1+2. [3]

SIMD

Počítač obsahuje více procesorů, který je řízen jedním programem. Každý z těchto procesorů pracuje s různými daty, ale každý procesor provádí stejnou instrukci. Schéma architektury SIMD je vidět na obr. 2. [3]



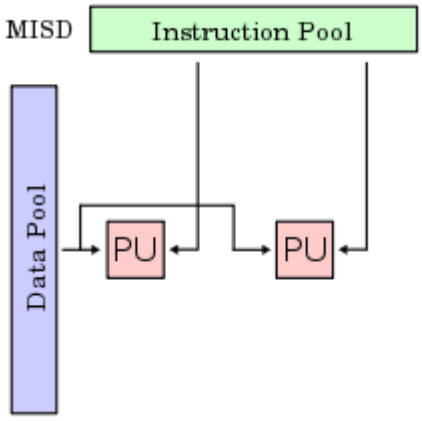
Obrázek 2: Schéma SIMD. Zdroj [16].

Z obrázku je vidět, že výpočetní jednotky (PU) provádí stejnou instrukci na různých datech. Například instrukce sečtení dvou čísel c=a+b. PU_1 bude provádět c= 1+2, PU_2 c=3+5, PU_3 c=4+6, PU_4 c=8+9. Příkladem této architektury jsou superpočítače od firmy Cray, které před desítkami let patřili k nejvýkonnějším počítačům na světě. [3]

MISD

Počítač obsahuje více procesorů jako v předešlé architektuře. Každý z procesorů pracuje se stejnými daty, ale každý procesor provádí jinou instrukci nad těmito daty. Tato architektura vznikla spíše na základě této klasifikace. V běžné praxi tuto architekturu nenajdeme. Schéma architektury MISD je znázorněn na obr. 3. [3]

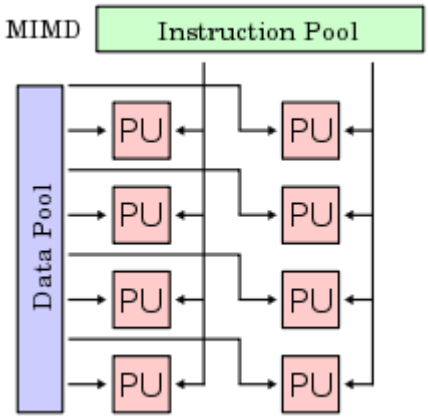
Z obr. 3 je vidět, že výpočetní jednotky pracují se stejnými daty, ale každá jednotka provádí jinou operaci.



Obrázek 3: Schéma MISD. Zdroj [17].

MIMD

Systémy typu MIMD pracují tak, že každý procesor pracuje s různými daty a provádí jiné operace s těmito daty. Schéma MIMD je zobrazen na obr. 4. [3]



Obrázek 4: Schéma MIMD. Zdroj [18].

Z obrázku 4 je vidět, že každý procesor provádí jiné operace na jiných datech. Příkladem může být PU_1 c=3+5, PU_2 a=3*5 apod.

1.2.2 Uspořádání podle paměti

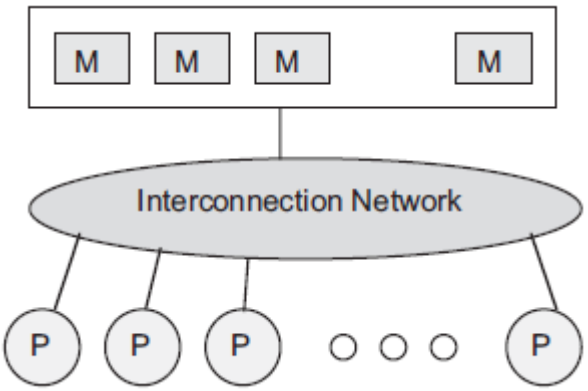
Flynnova taxonomie je velmi používanou klasifikací, ovšem její nevýhoda tkví v tom, že je velice hrubá. Z jejího hlediska se používají pouze systémy typu SIMD a MIMD. Proto se také zavedlo klasifikaci paralelních systémů dle uspořádání paměti. [1]

Systémy se dělí na [1]:

- se sdílenou pamětí,
- s distribuovanou pamětí,
- s distribuovanou sdílenou pamětí.

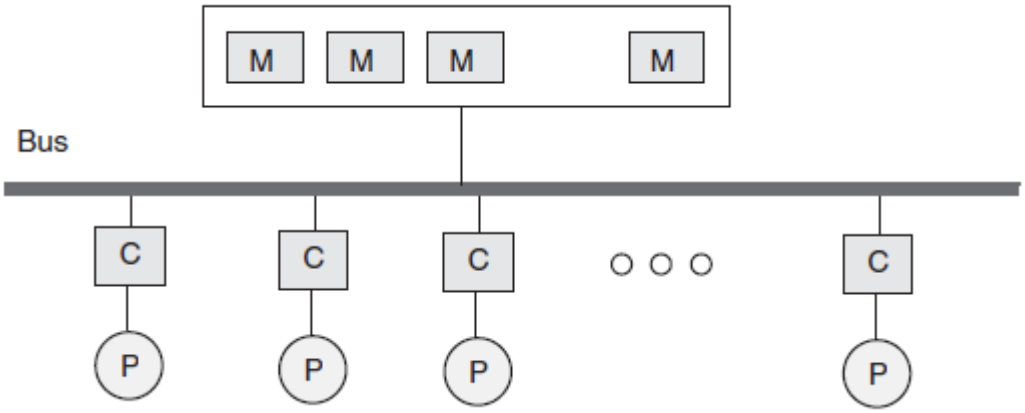
Sdílená paměť

Jedná se o systémy, kdy mají všechny procesory (výpočetní jednotky) přístup do fyzicky sdílené (globální) paměti. Z obr. 5, je vidět, že procesory mají přístup pomocí propojovací sítě k paměťovým modulům, které jsou pro všechny procesory stejné (jakýkoliv procesor má přistup do jakéhokoliv paměťového místa). Procesory spolu komunikují právě přes sdílenou paměť pomocí operacemi **read** a **write**. [1][3][10]



Obrázek 5: Schéma sdílené paměti paralelních systémů. Zdroj [1].

Pokud výpočetní jednotky mají přístup do paměti stejný, označují se jako UMA (Uniform Memory Access, dále jen UMA). UMA systémy se sdílenou pamětí se vyznačují tím, že všechny procesory jsou identické a mají stejný časový přístup do paměti. Jelikož přístup do sdílené paměti je rovnoměrný, tyto systémy se někdy označují jako symetrické multiprocesory (Symmetric multiprocessing, dále jen SMP). Architektura SMP představuje situaci dvou nebo více identických procesorů na jedné základní desce, které mají stejnoměrný přístup do paměti. Architektura UMA je ilustrována na obr. 6. [1][5]



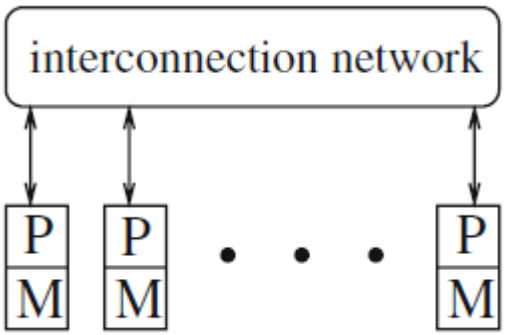
Obrázek 6: Schéma architektury UMA. Zdroj [1].

U této architektury se procesory do paměti dostávají pomocí instrukcí **read/write**. Problémem u této architektury je, aby dva procesory nemohli zapisovat v jeden okamžik paměťovou buňku paměti. [10]

Distribuovaná paměť

Paralelní systémy s distribuovanou pamětí jsou charakterizovány tím, že každý procesor má svoji fyzickou lokální paměť a procesory (výpočetní uzel) jsou propojeny pomocí propojovací sítě a poskytují přenos dat mezi těmito uzly. [10]

Procesor do své paměti může přistoupit přímo, ale pokud žádá přístup do vzdálené paměti jiného výpočetního uzlu, musí požádat tento uzel o přístup do jeho paměti pomocí zpráv **send/receive.** [10]

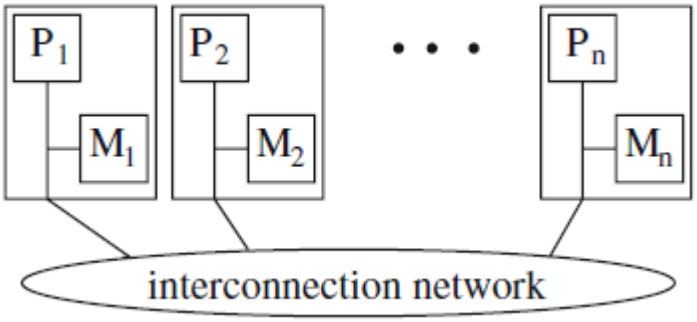


Obrázek 7: Schéma paralelního systému s distribuovanou pamětí. Zdroj [10].

Distribuovaná sdílená pamět

Hybridní systémy s distribuovanou sdílenou pamětí jsou charakteristické tím, že fyzickou paměť mají distribuovanou, ale pomocí virtualizace vidí procesory společný sdílený adresový prostor. To znamená, že procesor zapisuje data, ale fyzicky neví, kde jsou data uložena. [10]

Přístup do paměti je závislý na tom, zda se jedná o lokální nebo vzdálený přistup do paměti. Jelikož tento čas do paměti není rovnoměrný, ale odlišný, nazývá se tato architektura NUMA (Non Uniform Memory Access, dále jen NUMA) viz. obr. 8. [10]



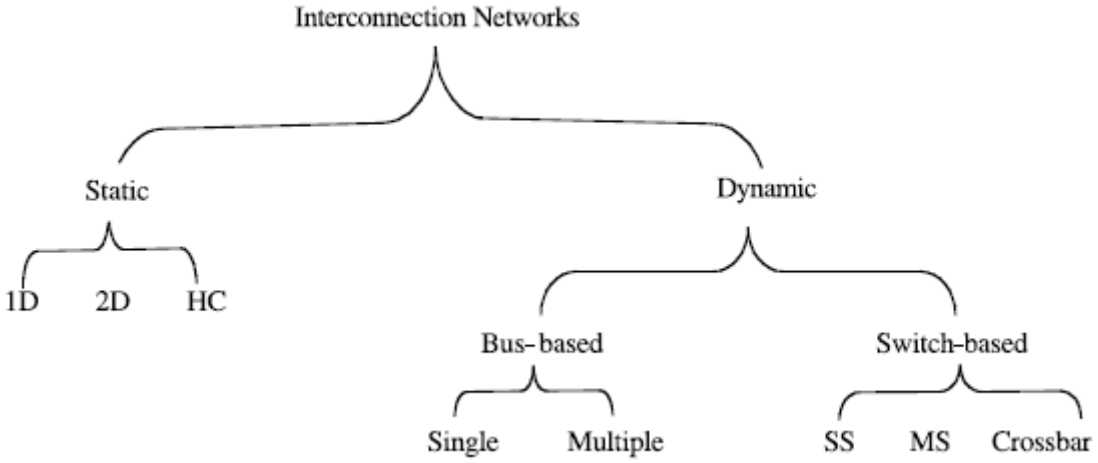
Obrázek 8: Architektura NUMA. Zdroj [10].

1.3 Propojovací sítě

Důležitou součástí paralelních systémů je samotná komunikace mezi výpočetními uzly, která probíhá přes propojovací síť. Pomocí propojovací sítě si mohou procesory vyměňovat informace. [1][6]

Jedno ze základních dělení propojovacích sítí je klasifikace na **statické** a **dynamické**. Statické propojovací sítě spojují uzly (procesor, paměť) pomocí přímé cesty, která je mezi nimi vytvořena a je neměnná. Statické sítě jsou někdy označovány jako přímé nebo point-to-point sítě. Tyto sítě jsou využívány především v systémech s distribuovanou pamětí (distribuovaný adresový prostor). Dynamické sítě oproti statickým mohou spojovací cesty měnit. To znamená, že spojovací cesty mohou vznikat i zanikat. Dynamické sítě jsou označovány jako nepřímé sítě. Vhodnost použití dynamických sítí je především v systémech se sdílenou pamětí (sdíleným adresový prostor) nebo distribuovanou pamětí. [10]

Hlubší dělení statických a dynamických sítí znázorňuje obr. 9, kde se statické sítě dělí na 1D (cesta), 2D (kružnice, strom,…), 3D (hyperkostka) a dynamické se dále dělí na **sběrnicové** a **přepínací.** O jednotlivých topologiích bude řečeno v následujících podkapitolách. [1]

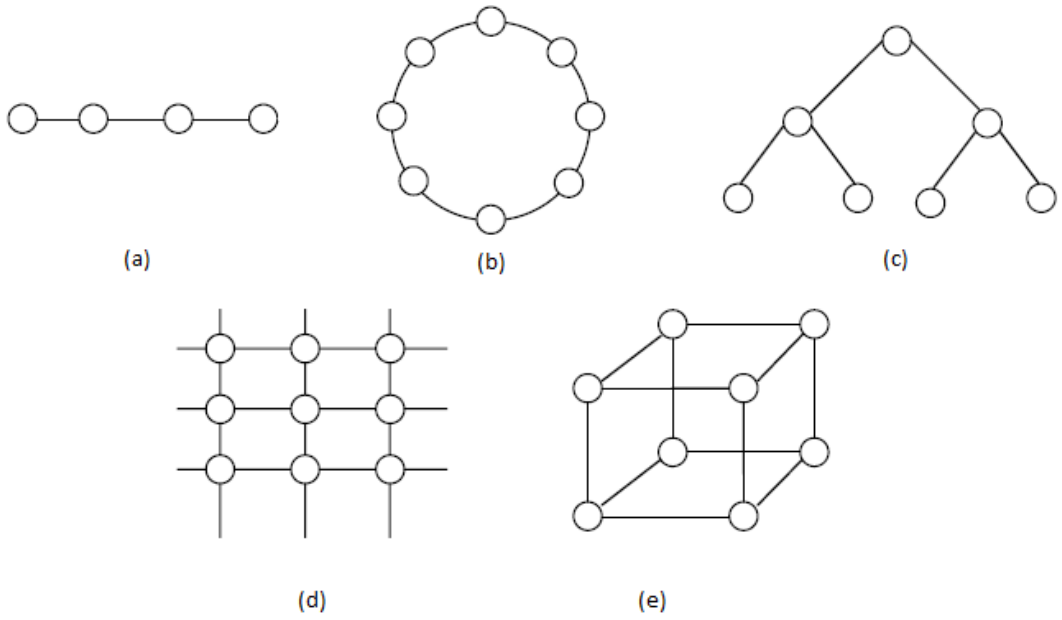


Obrázek 9: Klasifikace propojovacích sítí. Zdroj [1].

1.3.1 Statické sítě

Statické sítě jsou definovány tak, že při své činnosti nemohou měnit svoji strukturu, jak už bylo řečeno výše. Z tohoto důvodu se od počátku přizpůsobují činnosti, které budou po síti probíhat. Přehled statických struktur jsou zobrazeny na obrázku č. 10. Každý z vrcholů, které jsou na obrázku, představuje uzel sítě. Tedy pokud chce komunikovat jeden procesor s n-tým prvkem sítě, musí poslat informaci skrz síť, kde mezilehlé uzly sítě tvoří přestupnou stanici k cílovému modulu. Pokud je graf acyklický, je cesta k cílovému uzlu jednoznačná. Toto řešení velmi zjednodušuje adresaci. Ovšem nevýhodou statických sítí je, že dojde-li k poruše uzlu sítě, celý přenos nemůže být uskutečněn. U cyklických grafů tento problém nenastává. Pokud selže jedna cesta, může komunikace probíhat jinou cestou. [1][6]

Důležitou vlastností sítě je průměr grafu a stupeň vrcholu. Průměr grafu určuje nejdelší vzdálenost, která je mezi dvěma libovolnými uzly. Průměr grafu tedy stanovuje rychlost pro přenos zprávy od uzlu A k uzlu B a má vliv na celkovou rychlost přenosu. [11]



Obrázek 10: Příklady statický struktur cesta (a), kružnice (b), binární strom (c), mříž (d), krychle (e). Zdroj upraven dle [1].

Stupeň grafu určuje počet hran, které do uzlu vstupují nebo vystupují. Stupeň grafu tedy stanovuje, kolik přenosových modulů je potřeba. Z tohoto parametru se pak určuje cena celé sítě. V následující tabulce je výpis průměru grafu a stupně vrcholu sítí z obr. 10. [6][13]

Tabulka 1: Parametry statických sítí. Zdroj [6].

typ sítě	průměr grafu	stupeň vrcholu
cesta	N-1	2
kružnice	N/2	2
binární strom	$2[\log_2(N+1)-1]$	3
mříž	$2*N^{1/2}-2$	4
krychle	$\log_2 N$	$\log_2 N$

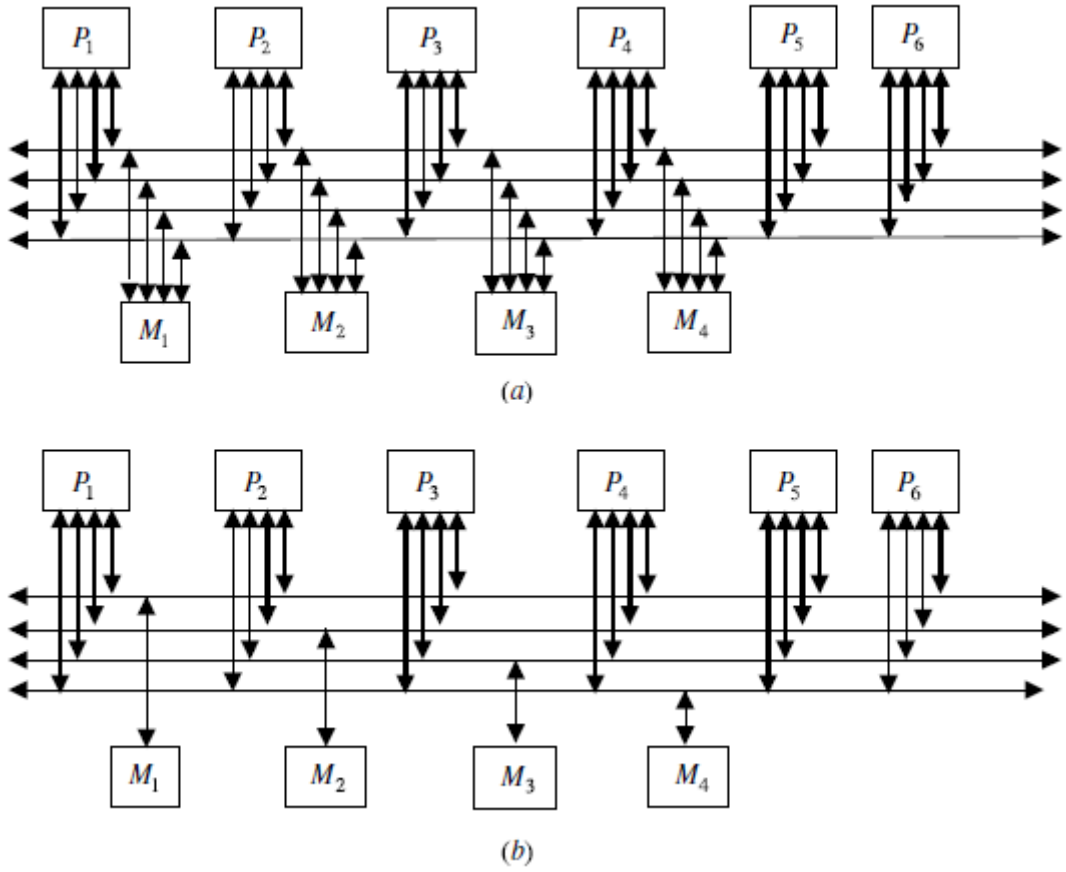
Při zvyšování stupně vrcholu u statických sítí může docházet k technickým problémům, které nelze překonat. V těchto případech se dává přednost dynamickým systémům, které dovolují přímé spojení s více moduly (procesor, paměť). [6]

1.3.2 Dynamické sítě

Dynamické sítě jsou charakterizované tím, že spojovací cesty vznikají nebo zanikají podle jejich potřeby. Součástí dynamických systémů jsou sběrnice (bus), křížové přepínače (crossbar) a víceúrovňové sítě (MINs). [1][6]

Sběrnice (Bus)

Sdílená sběrnice patří mezi nejjednodušší a nejlevnější realizace paralelních systémů. Realizace pomocí sběrnice má své výhody i nevýhody. Výhodou jak už bylo řečeno, je její jednoduchá realizace, jelikož všechny moduly jsou propojeny jednou sdílenou sběrnicí. Nevýhodou společné sběrnice je, že pokud chce procesor komunikovat s n-tým procesorem pomocí sdílené paměti je dovolena komunikace pouze jednoho procesoru. Ostatní procesory mohou vykonávat vlastní instrukce, které si z paměti načetly nebo naslouchají na sběrnici a čekají, dokud sběrnice nebude volná. Příklad společné sběrnice je zobrazen na Obrázek 6 v kapitole 1.2.2 sdílená paměť. [6]



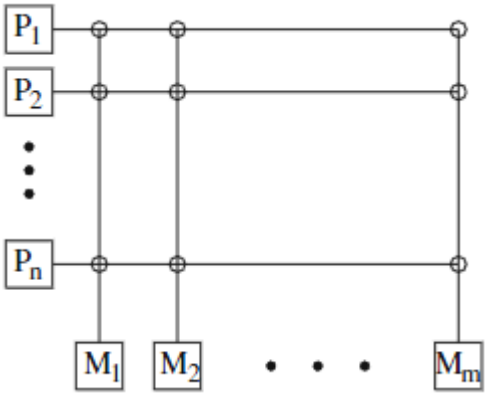
Obrázek 11: Vícesběrnicová propojovací síť – úplné propojení s paměťovými moduly (a), částečně propojení s paměťovými moduly (b). Zdroj upraven dle [1].

Problémy s propustností sítě může řešit více samostatných sběrnic, které jsou připojeny ke každému modulu paměti. Možnosti propojení sběrnice je znázorněn na obr. 11, kde každá paměť je připojena úplně (a) nebo každá paměť bude náležet jedné

sběrnici (b). U topologie úplného propojení dochází ke zvyšování průchodnosti i spolehlivosti oproti typu topologie částečné. Celkově však zvyšuje průchodnost a spolehlivost při komunikaci oproti společné sběrnici. [1]

Křížové přepínače (Crossbar)

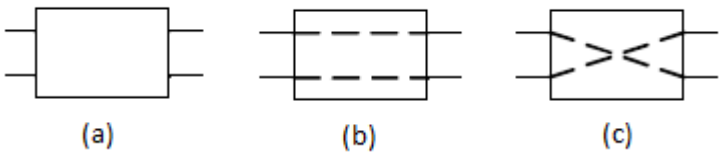
Křížové přepínače tvoří síť, která spojuje *n* vstupů s *m* výstupy (n x m). V systémech se sdíleným adresovým prostorem představují procesory vstupy a paměti výstupy. Výhodou křížových přepínačů je variabilita propojovacích cest, která lze vytvořit mezi *n* vstupy a *m* výstupy. Nevýhodou křížových přepínačů je vysoká cena, a proto se křížové přepínač používají pro malé počty procesorů. [6][12]



Obrázek 12: Dynamická síť s přepínači o n procesorů, m pamětí. Zdroj upraven dle [10].

Jednoúrovňové sítě (SS)

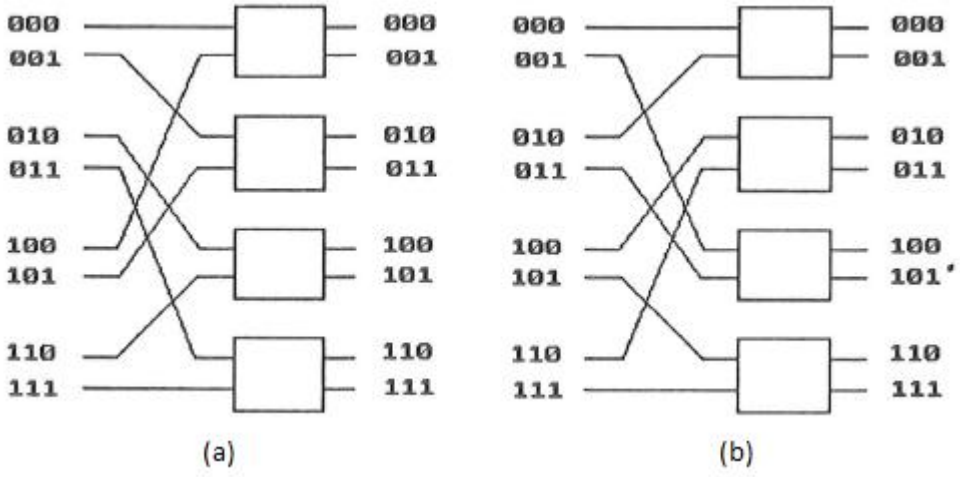
Jednoúrovňové sítě patří do skupiny propojovacích sítí typu promíchání s výměnou. Základním konstrukčním prvek sítě je přepínač 2x2, který je vidět na obr. č. 13. [6][10]



Obrázek 13: Přepínač 2x2 (a), stav - identita (b), stav - výměna (c). Zdroj upraven dle [10].

Přepínač 2x2 je základním prvkem propojovací sítě s dokonalým promícháním (a) a dokonalým oddělením (b), které jsou znázorněny na obr. 14. Způsob propojení dokonalého promíchání je dáno, že každému vstupu a výstupu je přiřazena číslo dvojkové soustavy. Výstupní kód získáme, že u vstupu posuneme vstupní kód cyklicky

doleva. U dokonalého oddělení je to právě naopak, kdy kódy přiřazené vstupům posuneme cyklicky doprava. Mezi další jednoúrovňové sítě lze označit síť nazvanou motýlek, která spočívá záměnou prvního a posledního bitu. [6][10]



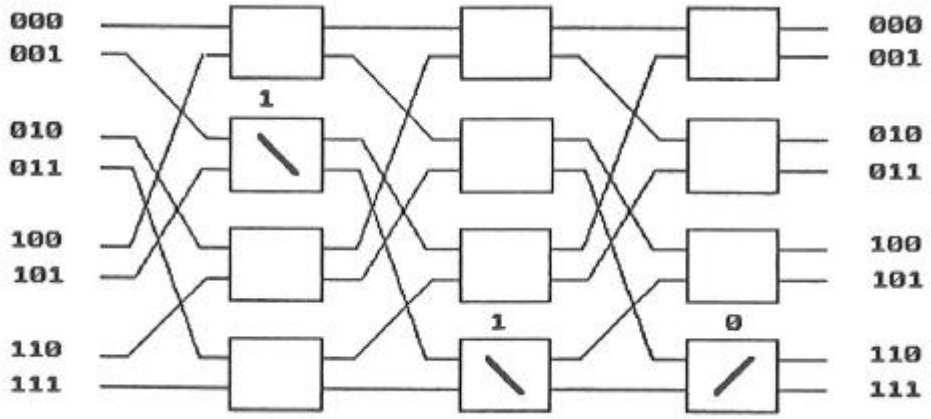
Obrázek 14: Dokonalé promíchání (a), dokonalé oddělení (b). Zdroj [6].

Víceúrovňové sítě (MS)

Víceúrovňové sítě jsou také označované jako MINs (Multiple Interconnection Networks). Liší se od jednoúrovňových tím, že jednotlivé vstupy lze propojit s jakýmkoliv výstupem. Toto platí pouze tehdy, jestliže všechny vstupy nejsou využity. MINs se dělí do tří skupin [6]:

- blokující sítě,
- přestavitelné sítě,
- neblokující sítě.

Propojení blokující sítě může být v okamžiku, kdy procesor naváže spojení s pamětí, blokováno existujícím spojením jiného procesoru s pamětí. Mezi nejznámější patří síť OMEGA, která je znázorněna na obr. 15. Přestavitelné sítě mohou uskutečnit všechny možnosti propojení procesoru s pamětí i za předpokladu, že bude muset přestavit dosud vytvořená spojení. Neblokující sítě jak už z názvu vyplývá, při vytvoření libovolného spojení mezi procesorem a pamětí, neblokují žádné spojení. [6]



Obrázek 15: Síť OMEGA 8x8. Zdroj [6].

1.4 Aplikace paralelních systémů

Vzrůstající tendence výpočetní náročnosti získali paralelní systémy důležitou roli v řadě oblastí. Využívání paralelních systémů má široké spektrum využití ve vědecké činnosti i v komerční sféře. Díky paralelním systémům došlo v minulosti k velkým pokrokům např. v těchto oblastech [3]:

- astrofyzika – aplikace zkoumání vývoje galaxie, analýza dat z vesmírných dalekohledů, sledování pohybu planet,
- počasí – sledování vývoje počasí, aplikace tání ledovců, modely oceánů,
- fyzika – modelování jaderné reakce,
- chemie,
- biotechnika,
- geologie, seismologie apod.

S rostoucími požadavky dynamického a statického obsahu, který webové stránky obsahují, jsou potřeba efektivní servery. Dále např. velký objem dat, z kterých je třeba rychle vyvodit analýzy pro obchodní a marketingové rozhodnutí jako např. využití superpočítačů velkých makléřských společností, kde je nutné rychle zvládnout až několik stovek tisíc souběžných objednávek v jeden okamžik. V komerční sféře se můžeme setkat s těmito aplikacemi [3]:

- databáze, datamining,
- zdravotní a diagnostické modely,

- letectví, kosmonautika,
- zpracování videa,
- virtuální realita,
- finanční a ekonomické modely,
- vyhledávací systémy (Google),
- 3D animace apod.

Z výše uvedeného vyplývá, že zastoupení paralelních systémů hraje důležitou roli v komerčních i vědeckých oblastí. Mnohé vědecké výzkumy nepochybně vyžadují vysoké nároky na výpočetní výkon. Právě díky paralelním systémům se tyto vědecké výzkumy urychlily.

2 Petriho sítě

Petriho síť (dále jen PS) byla vynalezena německým matematikem Carlem Adamem Petrim v roce 1962. PS kombinují matematickou teorii s grafickou reprezentací a jsou velmi silným nástrojem pro modelování ***dynamického chování systému, komunikace a synchronizace***. [7]

PS jsou velice silným nástrojem pro modelování spolupráce procesorů a následnou analýzu sítě. Modely, které poskytuje UML, nelze 100% využít, jelikož je třeba zkoumat dynamické vlastnosti chování modelu a to UML neumožňuje.

2.1 Definice PS

PS je neformální druh grafu, který definuje tři objekty – ***place (místo), transitions (přechod) a direct arc (orientovaná hrana).*** Formální stránka PS je pak definována jako $N=(P,T,I,O,M_0)$ kde [7]:

$P = \{p_1,p_2 \ldots p_m\}$ *je konečná množina míst,* (1)

$T = \{p_1,p_2 \ldots p_t\}$ *je konečná množina přechodů,* (2)

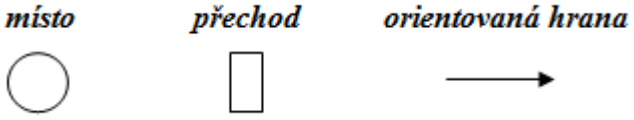
$I: P \times T \rightarrow$ *je vstupní funkce, která vede z místa do přechodu kde N je nezáporné číslo,* (3)

$O: T \times P \rightarrow$ *je výstupní funkce, která vede z přechodu do místa,* (4)

$M_0: P \rightarrow N$ *je počáteční ohodnocení.* (5)

2.2 Grafická reprezentace

PS definují grafickou reprezentaci pro zmíněné objekty: místo - kruh, přechod - obdélník a orientovaná hrana - šipku. Grafická reprezentace objektů vypadá následovně [4][8]:



Obrázek 16: Grafická reprezentace místa, přechodu a orientované hrany. Zdroj vlastní.

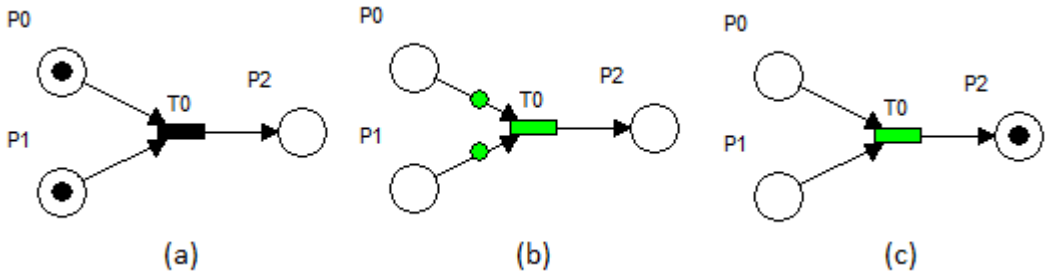
Označení v PS znamená přiřazení tzv. ***tokenů*** každému místu. Tyto tokeny představují dynamickou složku sítě. Místo může obsahovat žádný, jeden nebo více tokenů. Místo, které obsahuje jeden token, znázorňuje obr. 17. [7]

Obrázek 17: Označení tokenu v místě. Zdroj vlastní.

Spojování přechodu s místem nebo místo s přechodem se provádí pomocí orientované hrany. Není však možné spojovat dvě místa nebo dva přechody navzájem. Každá hrana je ohodnocena váhou, která určuje násobnost hrany. To znamená, kolik tokenů z místa p_1 bude odebráno do místa p_2. Pokud hrana nemá určenou hodnotu váhy, je tato váha rovna jedné. [4][8]

2.3 Umožnění a odpálení přechodu

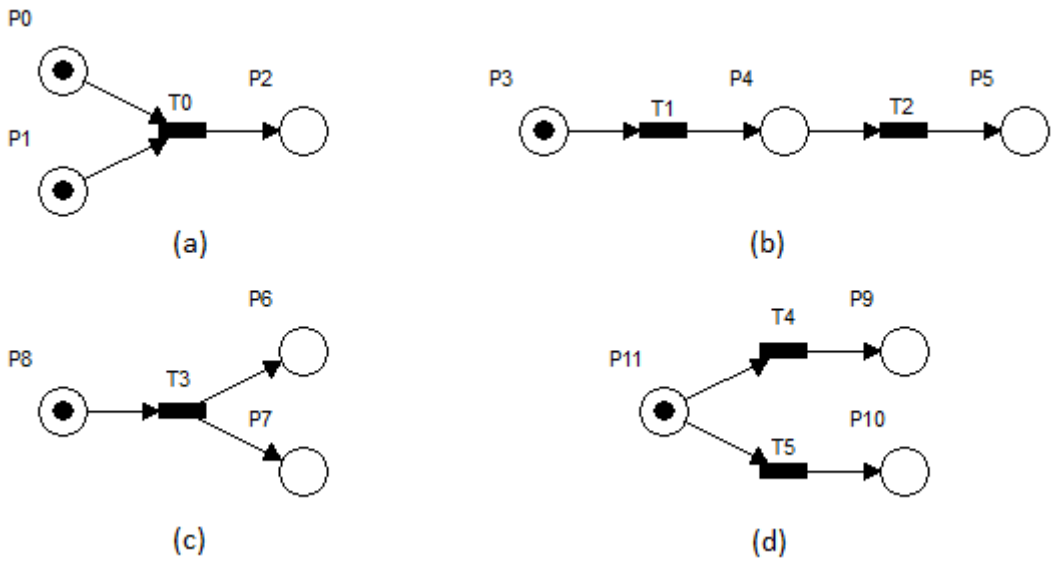
PS se během své činnosti může dostat do několika stavů. Aby mohla PS být činná, musí být ***přechod umožněn (enable)***. Přechod *t* je umožněn pouze tehdy, pokud vstupní místo *p* obsahuje tolik tokenů, kolik je požadovaná váha na orientované hraně. Je-li přechod *p* umožněn, může být ***odpálen (firing)***. Následně je ze všech vstupních míst počet tokenů odstraněn (dle vah na orientované hraně) a přesunut do výstupního místa. Umožnění a odpálení přechodu znázorňuje obr. 18. [7]



Obrázek 18: Petriho síť po provedení přechodu. Zdroj vlastní.

Z obr. 18 je vidět počáteční stav sítě (a). Přechod T0 je uskutečněn pouze tehdy, pokud vstupní místa mají tolik tokenů, kolik požaduje váha hrany. Tato podmínka je splněna, proto může být přechod uskutečněn (zelený) a odpálen (b). Po uskutečnění přechodu T0 se síť dostane do cílového stavu (c).

Modelování v PS dovoluje využít několik typů přechodů. Tyto typy zobrazuje obr. 19. [4][8]



Obrázek 19: Typy přechodů – synchronizace (a), sekvence (b), souběžnost (c), konkurence (d). Zdroj upraven dle [9].

Při umožnění a odpálení přechodu je možné nastavit **časové vyjádření přechodu**. V tomto případě se jedná o časované PS (dále jen ČPS). Pomocí ČPS můžeme měnit chování sítě. Přechody v ČPS mají dvojí časové vyjádření [8]:

- nulové doby trvání (immediate),
- nenulové doby trvání (deterministic),

U nulové doby trvání je odpálení provedeno ihned. Nenulová doba stanovuje čas, kdy po uplynutí doby, bude přechod umožněn a odpálen. V tomto případě se přechody, které mohou být z místa umožněny, nejprve **zbarví žlutě** a po aktivaci přechodu je zbarven zeleně a následně odpálen. [8]

2.4 Základní vlastnosti

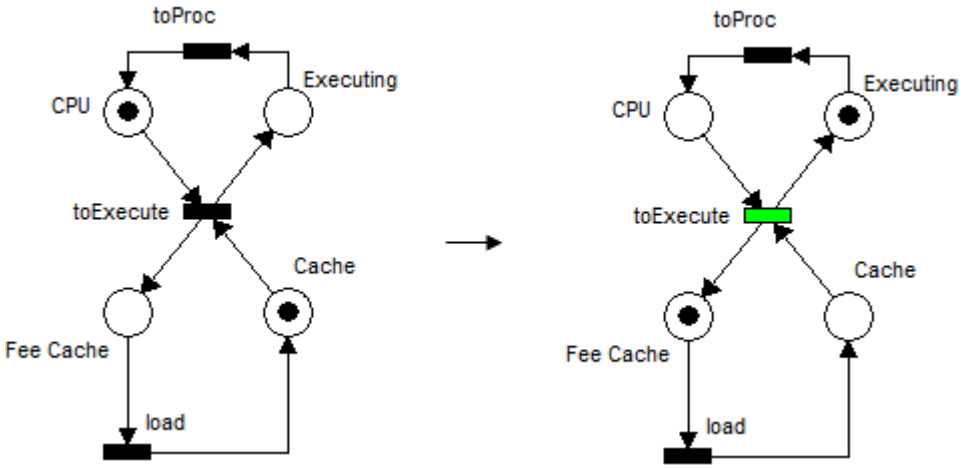
2.4.1 Ohraničenost a bezpečnost

PS je ohraničená (*k*-omezená) tehdy, jestliže pro každé dosažitelné značení, není počet tokenů v místě vyšší než hodnota *k*. Pokud se *k=1*, pak se říká, že PS je bezpečná. To znamená, že v každém stavu a místě PS je jeden token. [4][8]

Bezpečnost (safe) zaručuje, že místa v PS nepřesáhnou hodnotu 1. Bezpečnost lze explicitně zajistit tím, že nastavíme v místech kapacitu na hodnotu 1. Bezpečnost lze považovat za obecnější vlastnost. [4][8]

2.4.2 Konzervativnost

Konzervativní PS zaručuje, že všechny tokeny v síti (reprezentující prostředky) nebudou vytvářeny ani ničeny. To znamená, že že počet tokenů se při uskutečňování přechodů nezmění. Konzervativní síť je znázorněna na obr. 20. [4][8]



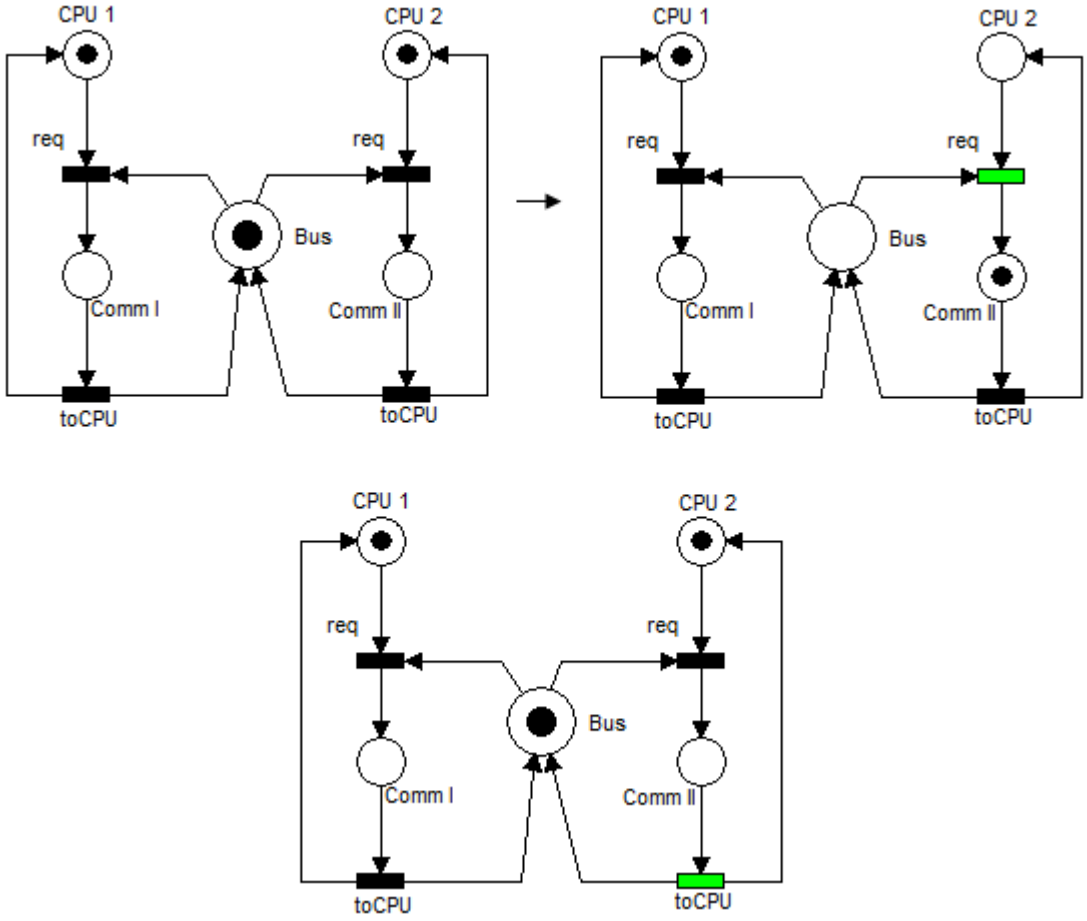
Obrázek 20: Konzervativní síť. Zdroj upraven podle [4].

Z obrázku 20 je patrné, že po uskutečnění a odpálení přechodu (*toExecute*) do dalšího stavu, se počet tokenů v celé síti nezmění. Po dalším provedení a uskutečnění přechodu (*toProc*) se síť dostane do svého počátečního stavu a počet tokenů v síti se opět nezmění. Jak je vidět, síť splňuje podmínku neměnného počtu tokenů po provedení přechodu, a tudíž splňuje vlastnost konzervativní sítě. [4]

PS se používají pro modelování přidělování sdíleného zdroje (např. přidělení sběrnice procesoru pro zápis/čtení do/z paměti). Operace spojené právě s přidělování zdroje se modelují pomocí odebírání a přidělování tokenu např. procesoru o zdroj a následnému navrácení do místa zdroje. [4][8]

Velmi silnou vlastností právě výše uvedeného, kdy modelujeme přiřazení nějakého zdroje, je **striktně konzervativní PS vzhledem k váhovému vektoru**. To znamená, že počáteční počet tokenů musí být po provedení roven váhovému součtu

tokenů v místě. Striktně konzervativní PS vzhledem k váhovému vektoru je znázorněna na obr. 21. [4]



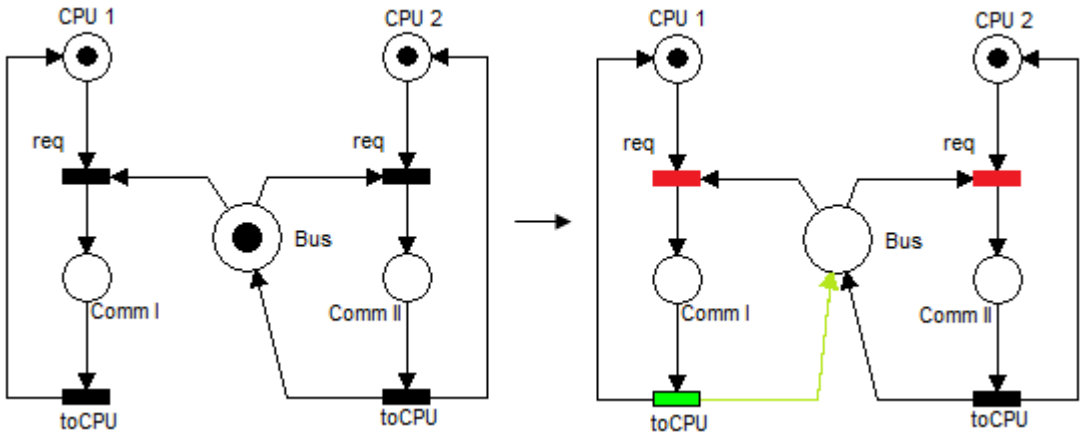
Obrázek 21: Konzervativní Petriho síť vzhledem k váhovému vektoru. Zdroj upraven podle [4].

Obrázek 21 ukazuje konzervativní síť vzhledem k váhovému vektoru w = [1,1,1,2,2] pro označení míst [*CPU 1, CPU 2, Bus, Comm I, Comm II]*. Zdroj, v tomto případě místo *Bus*, může být přidělen pouze jednomu procesoru (místo *CPU 1* nebo *CPU 2*). Z obrázku je sběrnice přidělena procesoru po aktivaci přechodu *req*. V tuto chvíli CPU 1 nemůže komunikovat a musí čekat, až bude sběrnice opět volná. Zároveň token v místě *Comm II* má váhu 2. Jak je vidět, po provedení a uskutečnění přechodu *toCPU* se počet tokenů nezmění. [4]

2.4.3 Živost a uváznutí

Důležitá vlastnost PS je živost sítě (liveness). K tomu, aby síť byla živá, musí být každý přechod uskutečněn nespočetně krát. To znamená, že každý přechod musí být

proveditelný. S živostí je spojeno i uváznutí (deadlock) sítě. Pokud při daném stavu sítě neexistuje přechod, který by mohl být uskutečněn, dojde k uváznutí sítě. Uváznutí sítě znázorňuje obr. 22. [4][9]



Obrázek 22: Uváznutí sítě. Zdroj upraven podle [4].

Z obrázku 22 je vidět, že sběrnice (*Bus*) může být přidělena CPU 1 nebo CPU 2. Pokud je zdroj (sběrnice) přidělen CPU 2, po provedení přechodu navrátí token zdroji. V případě přidělení CPU 1, po uskutečnění přechodu (zelený), procesor neodevzdá token zpět zdroji. Z toho vyplývá, že další přechody *req* (červený) nemůžou být provedeny a dochází k uváznutí sítě. K tomu, aby tato síť byla živá, musela by se přidat hrana (světle zelená) z přechodu *req* do místa *Bus*. [4]

3 Spolupráce SMP se sdílenou pamětí

Využití SMP systémů pro zvýšení výkonnosti je jednou z nejběžnějších realizací, s kterou se lze běžně setkat. Typickým přestavitelem je série procesorů Intel Xeon, které obsahují 2, 4, 6 nebo 8 CPU. [14]

Vlastnosti, které jsou spojeny s použitím SMP architektury pro různé aplikace, hrají důležitou roli. Především se jedná o tyto vlastnosti [11]:

- **škálovatelnost** - možnost přidání dalších prvků (procesor, paměť) do sítě,
- **dostupnost** - schopnost vyrovnání výpadku jednoho uzlu v síti,
- **latence** - doba přístupu procesoru do paměti.

Důležitou vlastností SMP architektur je škálovatelnost. Přidáváním procesorů dochází k zvyšování výkonu, ale otázkou je, do jaké míry je propustnost sítě dostatečná. S přidáváním procesorů tak může dojít ke zvýšení latence a tím pádem k snížení celého výkonu. [11]

V této části bude modelována spolupráce SMP se sdílenou pamětí, kde paměťové modely budou organizovány v jedné části systému a procesory k ní budou přistupovat pomocí těchto propojovacích sítí:

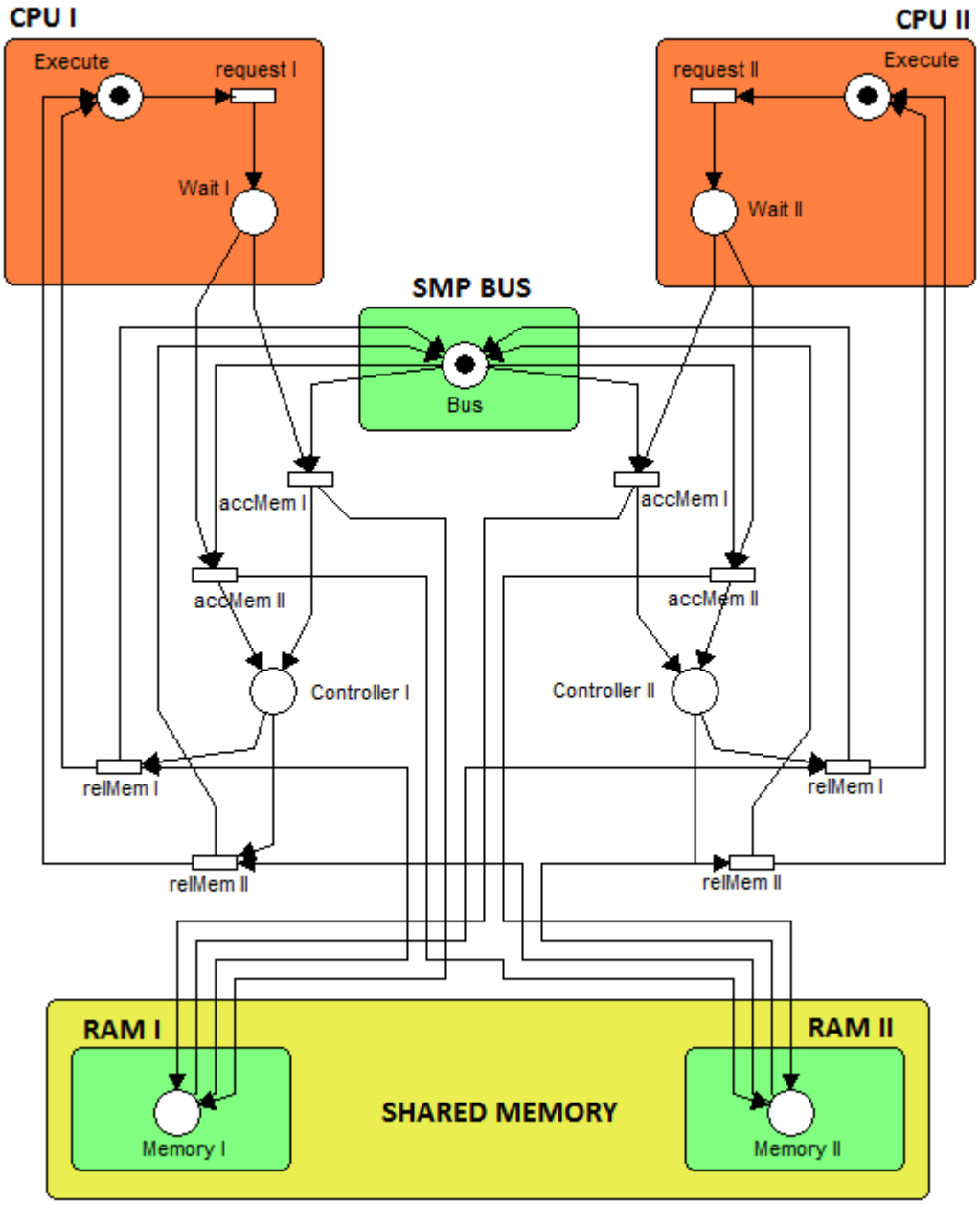
- sběrnice,
- křížového přepínače.

Jak už bylo řečeno v první kapitole, spolupráce procesorů bude probíhat pomocí sdílené paměti. Modely byly vytvořeny v programu HPSim, který je přiložen na CD. V navrhnutém modelu není modelována (pouze uvažována) cache paměť z důvodu zkoumání propustnosti propojovací sítě. Dalším předpokladem je, že procesor načítá všechna potřebná data z paměti pro danou úlohu.

3.1 SMP 2x2 – sběrnice

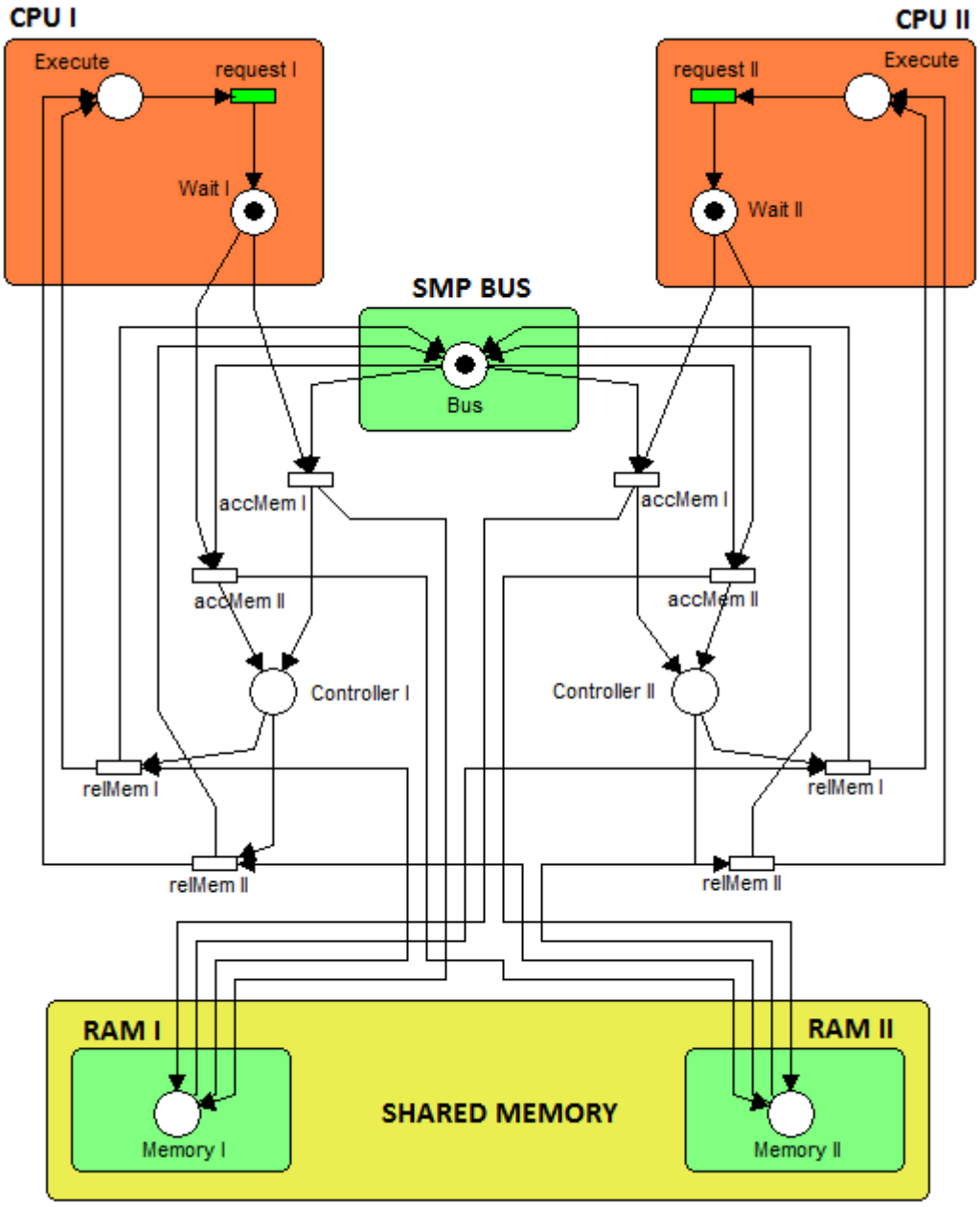
Vytvoření Petriho modelu SMP 2x2 (2 procesory, 2 paměti) znázorňuje obr. 23, kde je model ve svém počátečním stavu. V počátečním stavu jsou oba procesory (CPU I, CPU II) v místě *Execute*. To znamená, že procesor vykonává úlohu (jobs). V tu chvíli

je sběrnice volná a připravena k přidělení jednomu z procesorů a tím komunikovat se sdílenou pamětí (SHARED MEMORY).



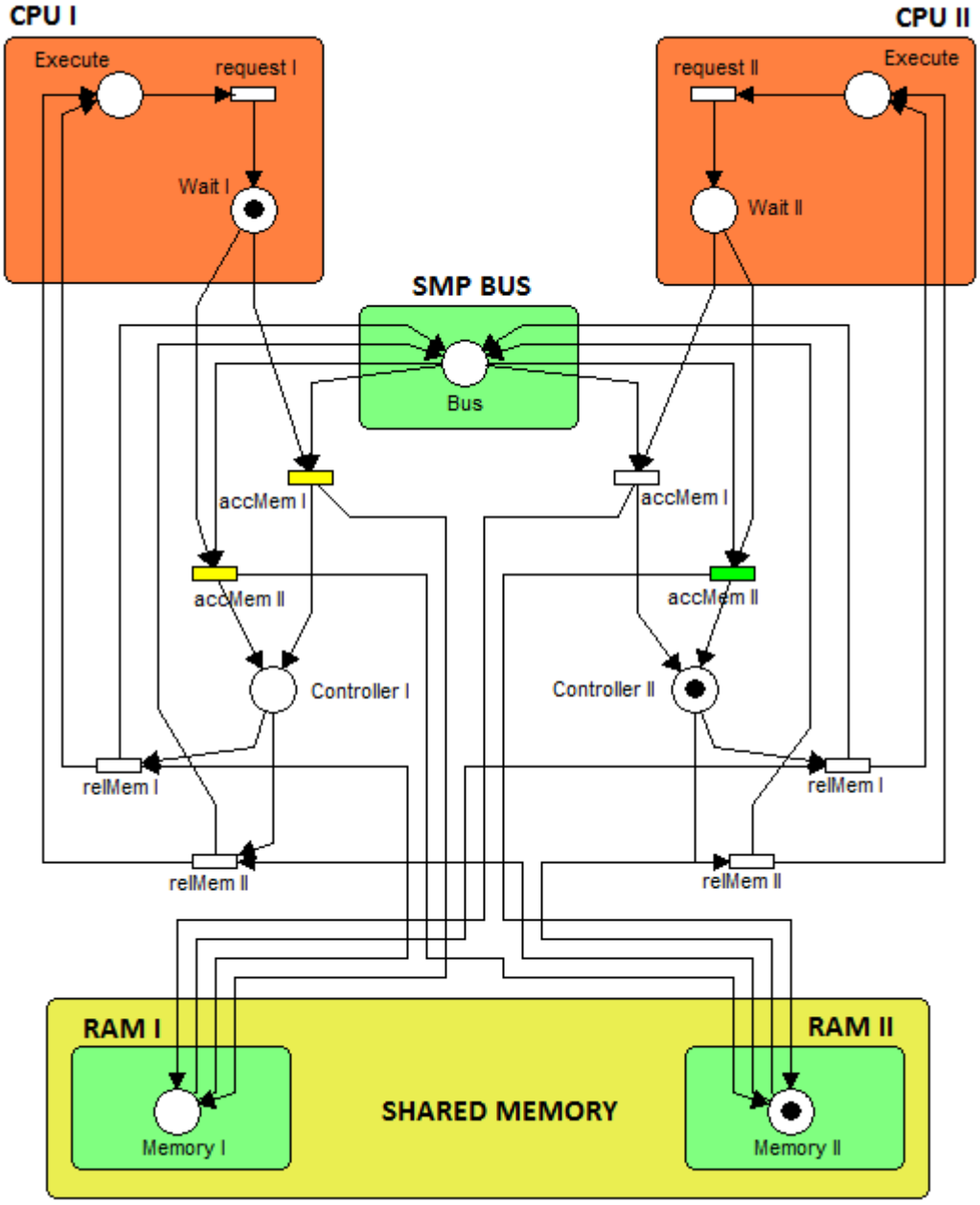
Obrázek 23: HPSim - SMP 2x2 (sběrnice). Zdroj vlastní.

Po ukončení běhu úlohy, procesor žádá přístup do paměti. Z místa *Execute* se uvolní token a je odpálen do místa *Wait* přes aktivní přechod *request*. V případě obr. 24 žádá přidělení paměti CPU II a token je přesunut do místa *Wait II*. Ve stejnou dobu CPU I stále vykonává úlohu.



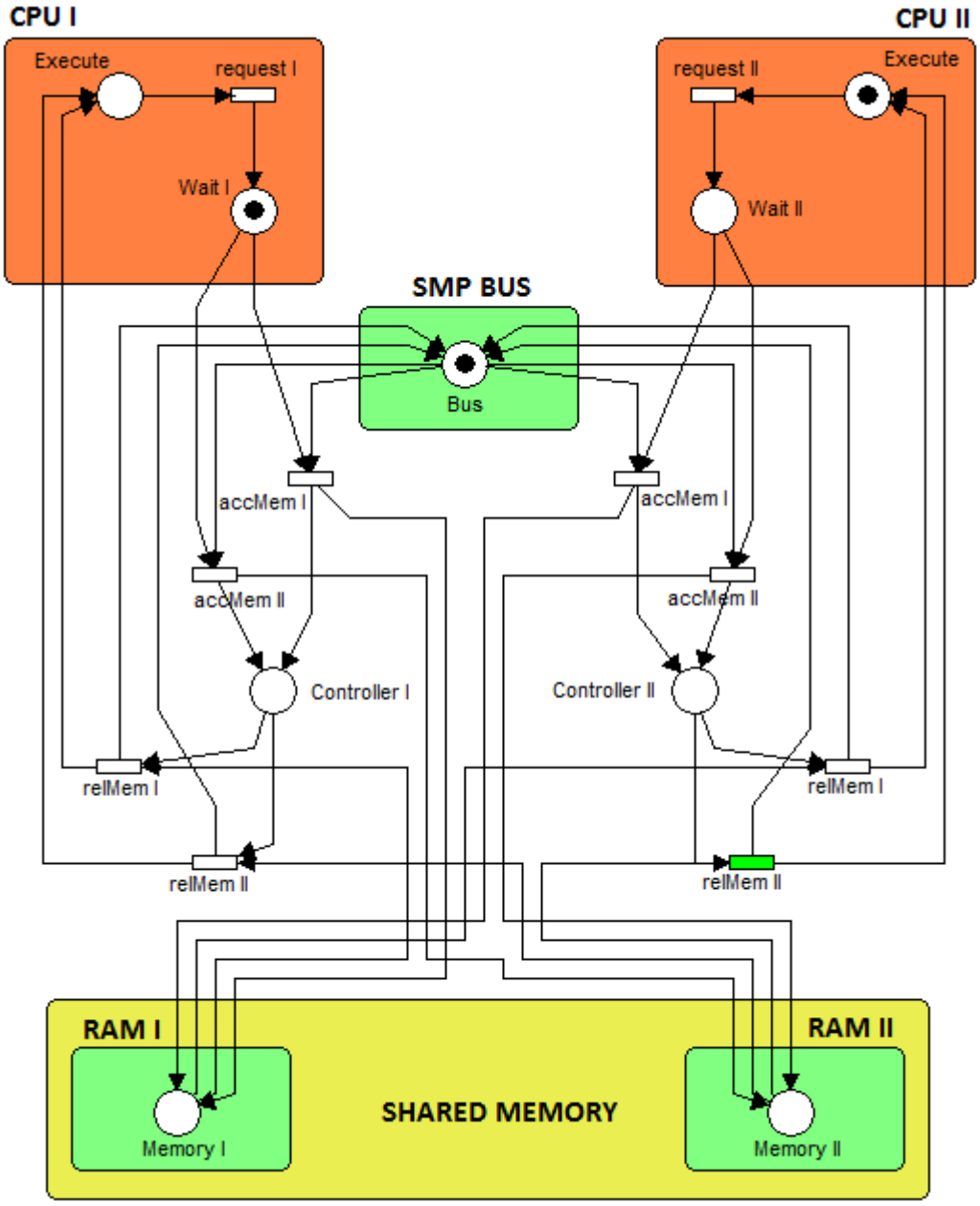
Obrázek 24: HPSim – SMP (sběrnice) žádost přidělení sběrnice. Zdroj vlastní

V počátečním stavu i v průběhu simulace dochází k situaci, kdy obě CPU chtějí komunikovat s pamětí a dochází ke konkurenci na přechodech *(accMem I, accMem II)*. Následně je jeden z přechodů, který má nenulové trvání umožněn a následně odpálen. V této fázi se z místa *Wait* uvolní token a vyšle jej do *Controlleru* a jedné z pamětí RAM *(místo Memory I nebo II)*. Místo *Controller* slouží pro určení komunikace CPU s pamětí a následné navrácení tokenu správně žádajícímu CPU. V tomto stavu CPU II komunikuje s pamětí, druhý musí čekat na uvolnění sběrnice viz. obr.25.



Obrázek 25: HPSim – SMP (sběrnice) přístup do sdílené paměti. Zdroj vlastní.

Z předešlého stavu, kdy jedno z CPU komunikuje a načítá potřebná data pro zpracování úlohy se aktivuje přechod *(relMem I, relMem II)*. Z místa *Memory* a *Controller* se uvolní token a přejde do místa *Execute*. Ve stejné době dochází k navrácení tokenu do místa *Bus*. Pokud je sběrnice opět volná pro komunikaci, v tuto chvíli může druhé CPU žádat o přidělení sběrnice a přistoupit do sdílené paměti Celý proces je zachycen na obr. 26.



Obrázek 26: HPSim - SMP (sběrnice) načtení dat z paměti. Zdroj vlastní.

Doba v místě *Execute* (vykonání úlohy) je závislá na nastavení přechodu *request*. Po umožnění a aktivaci přechodu *accMemory I, II* se z místa *Execute* uvolní token a přejde do místa *Wait I, II*. Tento stav odpovídá počátečnímu označení jednoho z CPU. Druhé CPU představuje vlastní jednotku, tudíž se může nacházet v jakémkoliv jiném stavu, než první CPU.

Předešlý popis modelu musí být doplněn o nastavení přechodů, o kterých dosud nebylo řečeno. Nastavené přechody modelu ukazuje tabulka 2.

Tabulka 2: Přechody SMP sběrnice. Zdroj vlastní.

název přechodu	typ přechodu
request I	*deterministic*
request II	*deterministic*
accMem I	*deterministic*
accMem II	*deterministic*
relMem I	*deterministic*
relMem II	*deterministic*

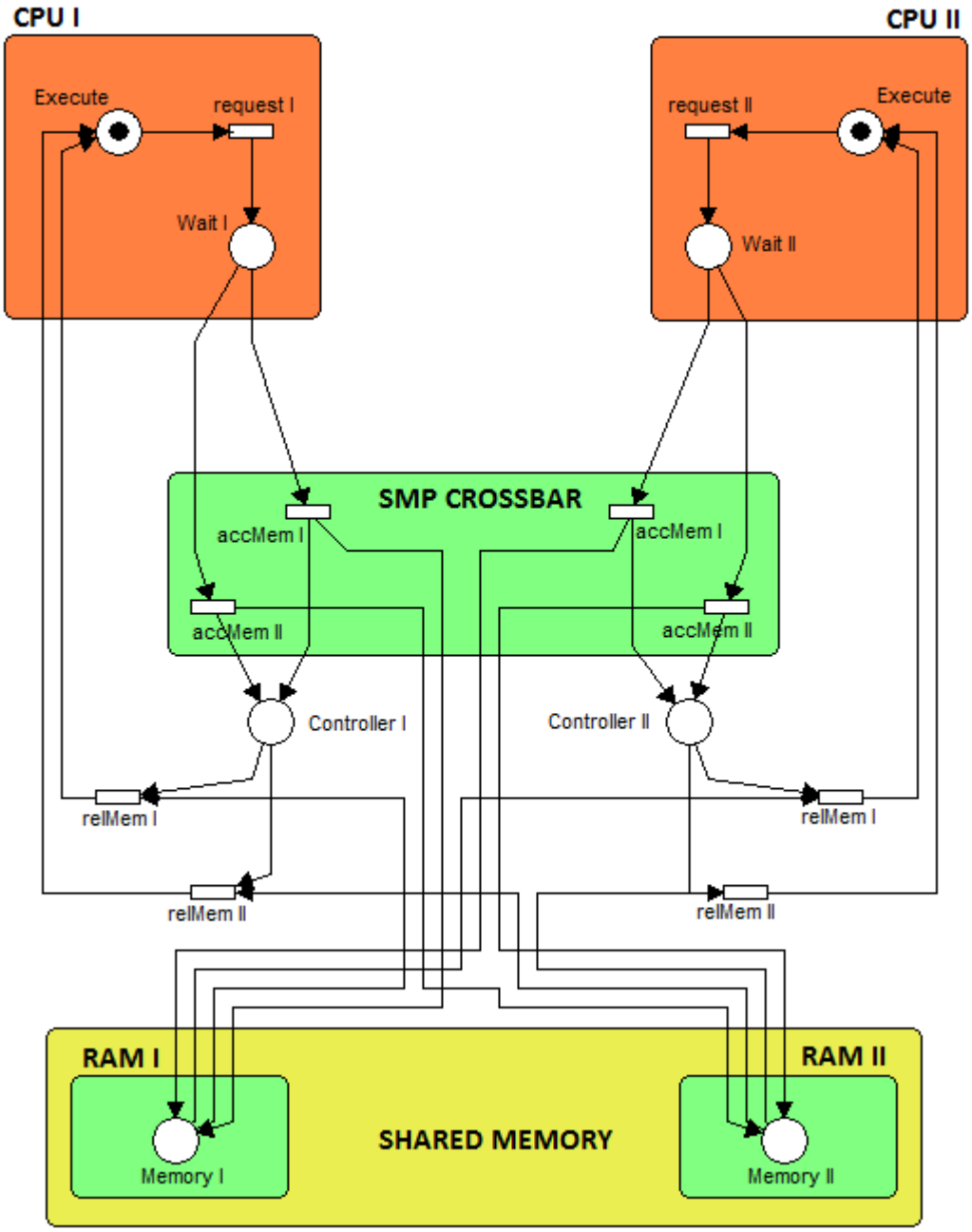
Pro další analýzu sítě byly modelovány SMP architektury pro 4x4 a 8x8, které jsou přiloženy v příloze. Jedná se o stejný způsob modelování jako na obr. 24, s tím rozdílem, že sdílená paměť je modelována jako jedno místo s kapacitou odpovídající počtu paměťových modulů.

3.2 SMP 2x2 – křížový přepínač

Křížový přepínač byl původně využíván v telefonních ústřednách. Možnost využití křížového přepínače pro propojení procesorů s pamětí je jednou z alternativ, ovšem toto řešení se v komerční oblasti široce nevyužívá. [4][6]

Jak už bylo řečeno, v první kapitole propojovací síť tvořena křížovým přepínačem představuje síť přepínacích prvků, která spojuje *n* procesorů s *m* paměti. Se škálovatelností systému se tak zvyšuje počet prvků v síti *N x M*. Z navrhnutého modelu tedy roste síť N^2, což odpovídá stejnému počtu procesorů a pamětí. Z důvodu složitosti modelované části je tato část (zeleně označena) pouze abstraktní. Představuje tak počet potřebných přepínacích prvků. [6][10]

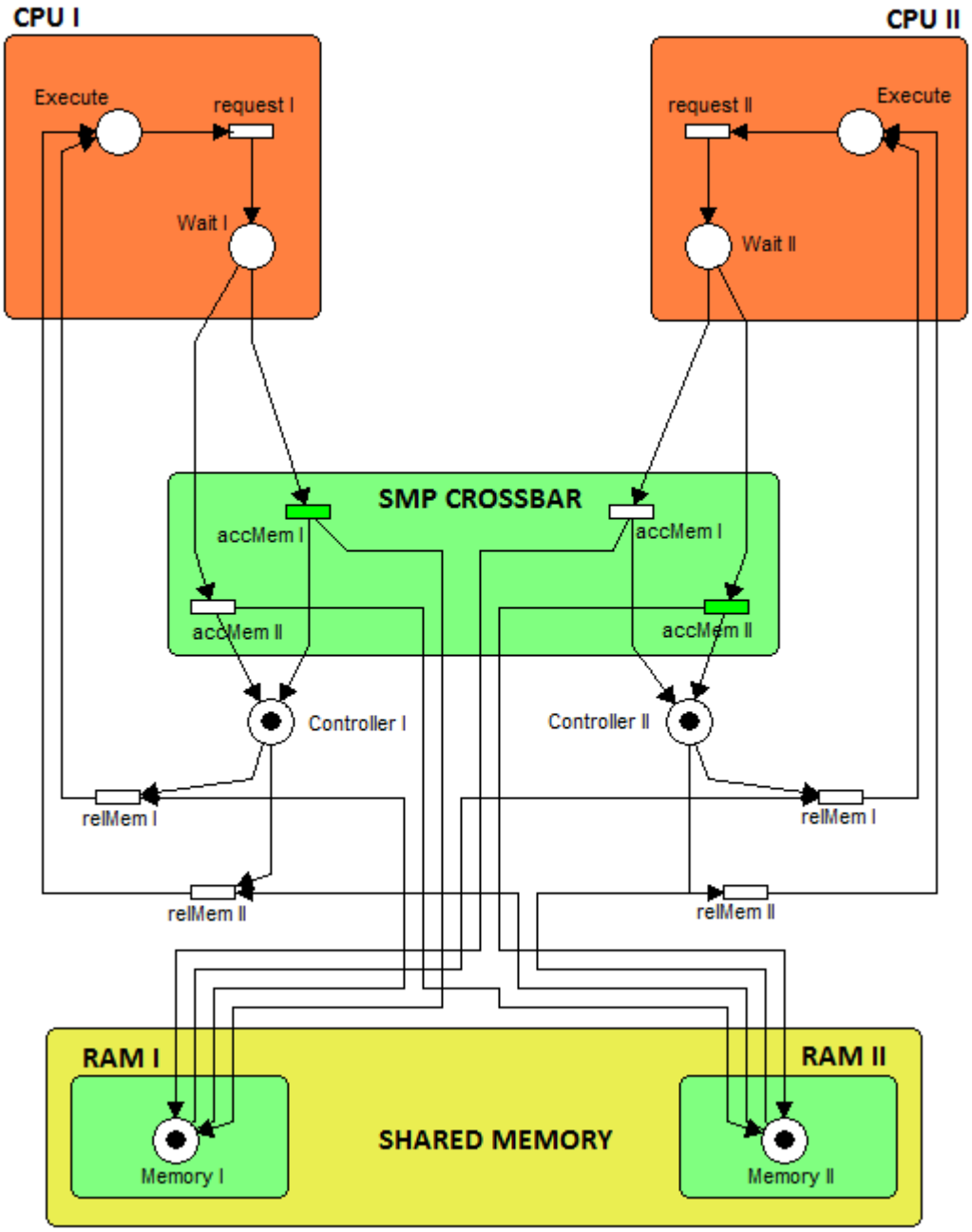
Propojení SMP 2x2 systémů se sdílenou pamětí pomocí křížového přepínače je znázorněno na obr. 27. Počáteční stav sítě odpovídá stejnému rozložení tokenů jako v případě sběrnice. Oba procesory jsou v místech, kdy procesor zpracovává úlohu. Propojovací síť ovšem představuje křížový přepínač (CROSSBAR), který obsahuje čtyři přepínací prvky a propojuje paměťové moduly. Přepínací prvky v modelu dovolují přistoupit k paměťovým modulům zároveň, jak je vidět v obrázku nedochází k žádné konkurenci mezi přechody.



Obrázek 27: HPSim - SMP 2x2 (křížový přepínač). Zdroj vlastní.

Pokud oba procesory ukončí práci nad úlohou, přejdou z místa *Execute* do místa *Wait,* kde oba procesory žádají přístup do paměti. V tomto momentě se procesory nachází v místech *Wait I a Wait II.* V tomto okamžiku můžou nastat dva stavy. Jeden z procesorů chce číst data z paměti a druhý procesor tak musí čekat, dokud svoji činnost nedokončí a poté může vstoupit do paměti. Druhá možnost, která může nastat, že každý

procesor zapisuje v jiné paměti a na rozdíl od sběrnicového typu, můžou oba přistoupit do jiné paměti, tak jak je znázorněno na obr.28.



Obrázek 28: HPSim – SMP (kříž. přepínač) přístup do sdílené paměti. Zdroj vlastní.

Propojovací síť pro 2x2 tvoří čtyři přepínací prvků. Při konstrukci SMP 4x4 představuje propojovací síť 16 přepínačů a pro řešení 8x8 až 64 prvků. Modelování samotných přepínačů je abstrakcí světle zelené části, která v sobě představuje síť

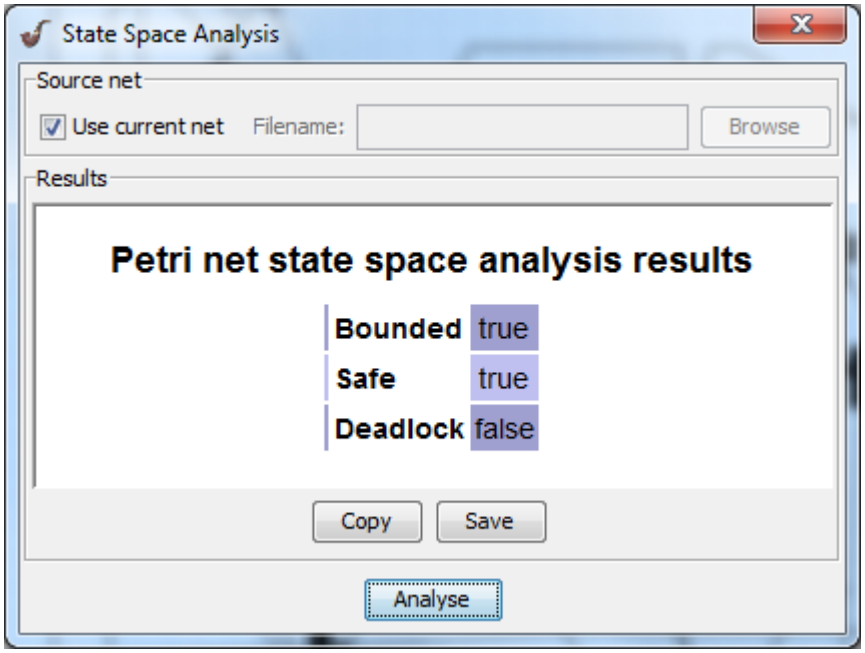
přepínacích prvků. Následující tabulka 3 ukazuje ohodnocení přechodů pro SMP křížový přepínač.

Tabulka 3: Přechody SMP kříž. přepínače. Zdroj vlastní.

název přechodu	typ přechodu
request I	*deterministic*
request II	*deterministic*
accMem I	*deterministic*
accMem II	*deterministic*
relMem I	*deterministic*
relMem II	*deterministic*

3.3 Verifikace modelu

Ověření navrženého modelu SMP bylo provedeno na základě stanovených vlastností v kapitole 2.4 . Z důvodu nedostačující verifikace programu HPSim, který ověřuje pouze uváznutí, byla tato verifikace provedena v programu PIPE3.0, který je přiložen na CD. Ověření navržených Petriho modelu – SMP je znázorněno na obr. 29.



Obrázek 29: PIPE3.0 -Verifikace Petriho modelu SMP (sběrnice). Zdroj vlastní.

Jak je vidět z obrázku, síť splňuje vlastnost omezenost, bezpečnost a to, že nenastane v žádném stavu síťě uváznutí. Pokud nedochází k uváznutí sítě, to značí, že PS je živá.

V případě ověření vlastností SMP (křížového přepínače) odpovídá omezenost (true), bezpečnost (false) a uváznutí (false). V navrhnutém modelu je porušena bezpečnost tím, že ve sdílené pamětí (Memory) je kapacita místa nastavena na počet odpovídajícímu počtu paměťových modulů. Bezpečnost je však obecnější vlastnost, která celkové nemá příliš velký význam.

4 Analýza komunikační režie

V rámci navržených modelů SMP – sběrnice (2x2, 4x4, 8x8) a SMP – křížový přepínač (2x2, 4x4, 8x8) byla provedena analýza komunikační režie s pamětí z kapitoly 3. Testování probíhalo v programu HPSim, který dovoluje následný export vyhodnocených dat do Microsoft Excel.

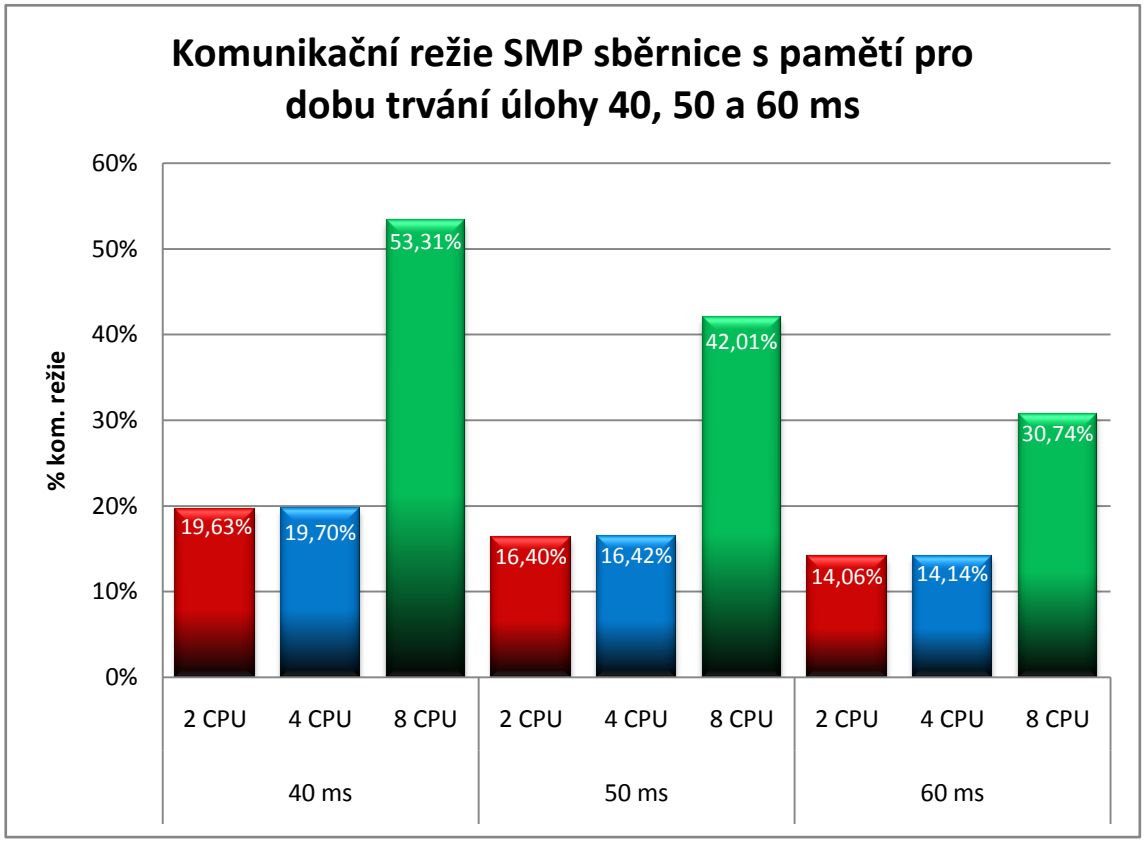
Pro analýzu byly stanoveny tři typy úloh, které jsou různě časově náročné. Z hlediska zkoumaných úloh byla nastavena **doba náročnosti zpracování úlohy** na 40, 50 a 60 ms. Analýza komunikační režie probíhala v simulačním čase **10 000 ms**. Časová režie odpovídá stavu, kdy procesory čeká v místě *Wait*. U SMP sběrnice je režie vyjádřena délkou času, kdy bude sběrnice přidělena jednomu z žádajících procesorů. U SMP křížového přepínače je komunikační režie stanovena počtem přepínacích elementů v sítí, která se škálovatelností roste. Hodnoty na přechodech křížového přepínače byly empiricky stanoveny s růstem přepínacích prvků. Následující tabulka ukazuje nastavení přechodů pro tři typy úloh.

Tabulka 4: Nastavení přechodů pro stanové úlohy. Zdroj vlastní.

úloha	název přechodu	sběrnice	křížový přepínač
1	*request I*	40	40
	request II	40	40
	accMem I	10	11
	accMem II	10	11
	relMem I	1	1
	relMem II	1	1
2	*request I*	50	50
	request II	50	50
	accMem I	10	12
	accMem II	10	12
	relMem I	1	1
	relMem II	1	1
3	*request I*	60	60
	request II	60	60
	accMem I	10	17
	accMem II	10	17
	relMem I	1	1
	relMem II	1	1

4.1 Komunikační režie SMP - sběrnice

Komunikační režii SMP sběrnice s pamětí pro dobu trvání úlohy 40, 50 a 60 ms znázorňuje graf 1. Z grafu je vidět, že z hlediska doby trvání úlohy 40 ms, 2 CPU a 4 CPU architektury dochází k minimálnímu nárůstu komunikační režie. Režie v tomto případě odpovídá 19,63% pro 2 CPU a 19,70% pro 4 CPU stráveného času čekáním na přidělení sběrnice. Z toho vyplývá, že sběrnice změnou doby náročnosti úlohy stíhá obsluhovat všechny žádající procesory, které chtějí přistupovat do sdílené paměti. Problém však nastává u 8 CPU architektury. Jak je vidět z grafu, nárůst prvků zvýší komunikační režii na 53,31% z celkového času simulace. Zde nastává problém s přidělení sběrnice a procesory, které žádají o paměť, stráví více jak 50% času. Tím, že procesory tráví více času naslouchání na sběrnici, se promítne do výkonnosti celého systému.



Graf 1: Komunikační režie SMP - sběrnice s pamětí pro dobu trvání úlohy 40, 50 a 60 ms. Zdroj vlastní.

Pokud zpracovanou úlohu prodloužíme o 10 ms simulačního času na 50 ms, je vidět následující komunikační režie. Prodloužením zpracování úlohy procesorem se 2

CPU a 4 CPU architektura dostane na 16,4% času čekání. Dojde tak k poklesu o 2,2% režie než v případě úlohy s menší náročností. Velký rozdíl zaznamenala architektura s osmi procesory. Jak je vidět, komunikační režie klesla na 42%, což znamená téměř 11 % pokles režie. Z toho vyplývá, že pro v 8 CPU architektuře, která je tvořena sběrnicí, dochází k nižší časové režii komunikace s pamětí a tím mírně klesne počet žádajících procesorů o paměť.

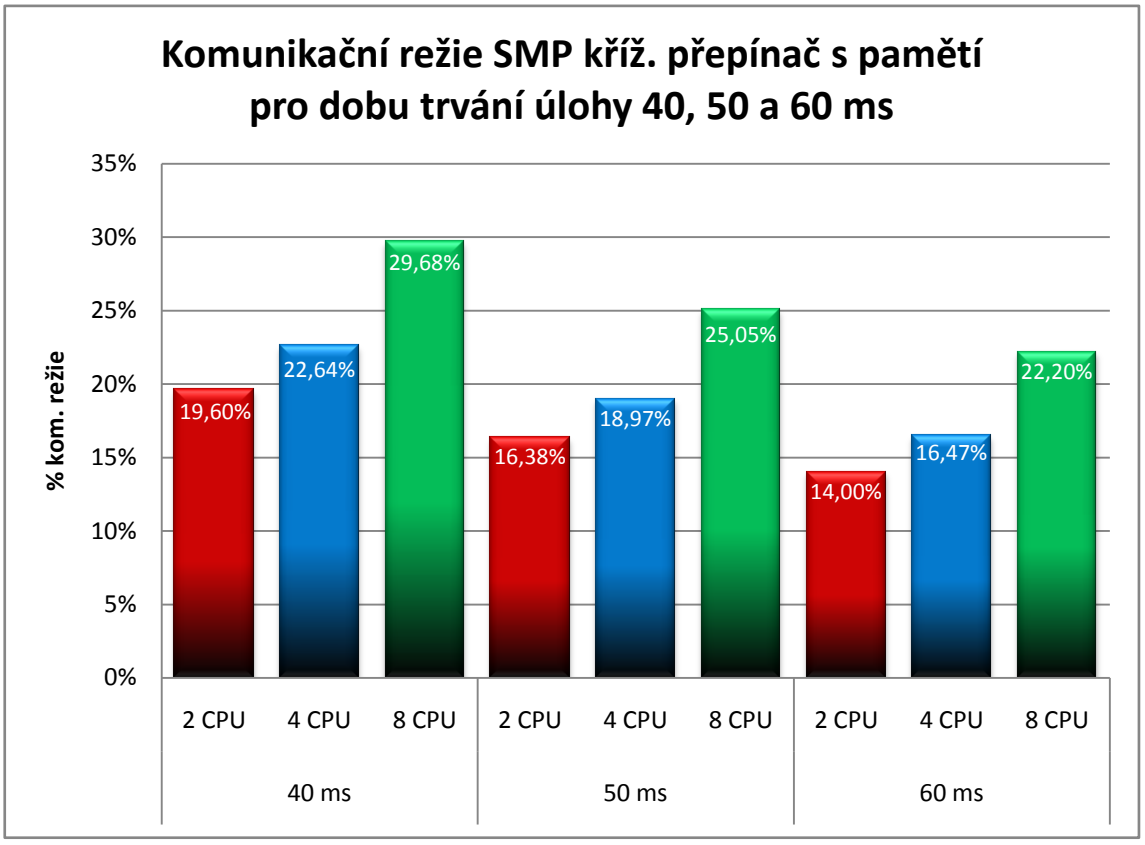
Nakonec byla provedena analýza komunikační režie pro úlohu s náročností 60 ms. Je vidět, že v případech 2 CPU a 4 CPU architektury, sběrnice stíhá obsluhovat všechny žádající procesory o paměť. Časová režie je u obou architektur pohybující kolem 14%. Dále je vidět, že komunikační režie 8 CPU je 30,74%. Z předcházejících grafů je možné vidět, že **s rostoucí náročností úloh klesá doba čekání na přidělení sběrnice** u 8 CPU a velmi malé poklesy zaznamenávají architektury s 2 CPU a 4 CPU.

Jak už bylo řečeno s růstem náročnosti úlohy klesá komunikační režie. Sběrnice tak v rámci možností obsluhuje menší počet procesorů, jež žádají přístup do sdílené paměti.

4.2 Komunikační režie SMP - křížový přepínač

Komunikační režie spojena s SMP – křížovým přepínačem je znázorněna v grafu 2. Z grafu je vidět, že režie vynaložená přepínáním mezi elementy sítě, není mezi 2 CPU a 4 CPU veliká. Rozdíl činí kolem 2% režie, což ve výsledném zpracování úloh promítne zvýšením výkonnosti, jelikož režie je mezi těmito architekturami minimální. Architektura s 8 CPU nevykazuje vysoký nárůst komunikační režie. Narůst odpovídá 6%. Z toho vyplývá, že růst počtu procesorů a pamětí v síti, tak nečinní velké problémy, co se týče komunikace s pamětí a tím působí i na zrychlení paralelního systému.

Pro úlohu, která má dobu trvání 50 ms, vidíme, že komunikační režie klesla na 16% u 2 CPU a 18% u 4 CPU. Lze tedy konstatovat, že komunikační režie výrazně neklesla s růstem procesorů. Na druhé straně architektura s 8 CPU dosahuje hodnoty 25%, což značí minimální nárůst doby komunikace s pamětí, což povede k zvýšení výkonnosti mezi 4 CPU a 8 CPU architekturou.



Graf 2: Komunikační režie SMP kříž. přepínače s pamětí pro dobu trvání úlohy 40, 50 a 60 ms. Zdroj vlastní.

Úloha, která má dobu zpracování 60 ms vykazuje architektura s dvěma procesory 14% režii komunikace s pamětí, což je minimální spotřeba času na komunikaci. U architektury se 4 CPU je režie kolem 16%. Srovnání dvou a čtyřprocesorové architektury je tato režie komunikace minimální. K snížení také dochází u 8 CPU vzhledem k 4 CPU architektury. Z toho vyplývá, že s větší škálovatelností architektury, není zaznamenán vysoký nárůst komunikační režie.

4.3 Výkonnost architektury/komunikační režie

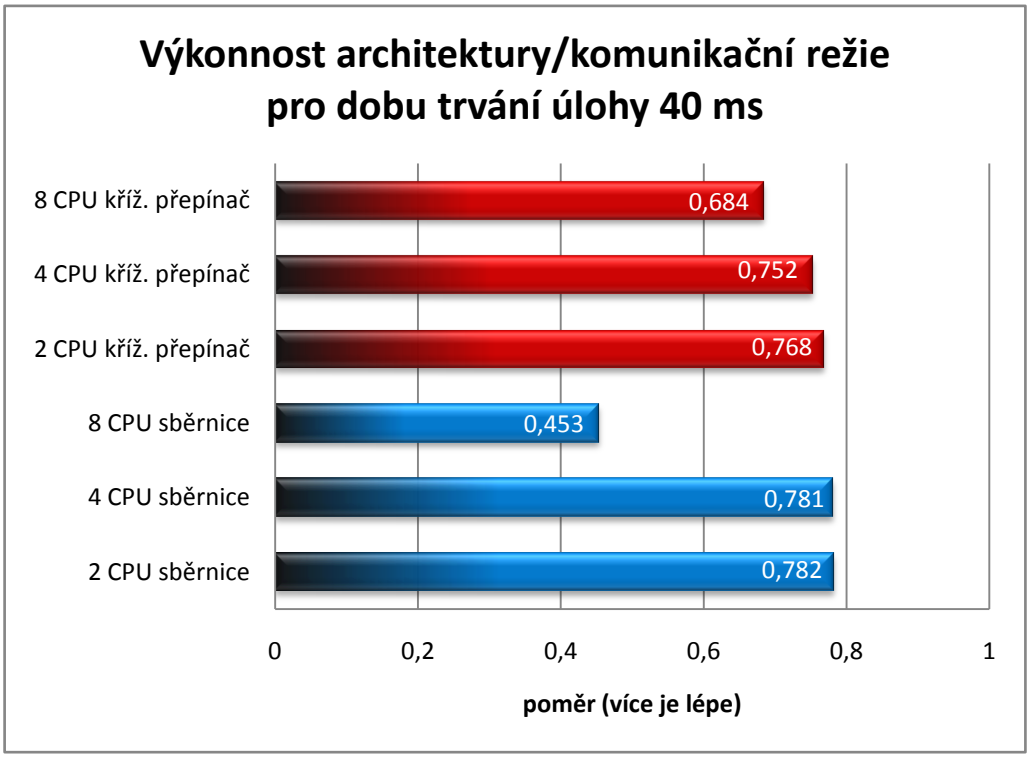
V předchozích kapitolách 4.1 a 4.2 byla analyzována komunikační režie, kterou stráví procesor čekáním na přidělení paměti. V této části bude uvedeno, jak se tato komunikační režie promítne do doby běhu zpracování úloh procesorem (výkon procesoru). **Poměr vyjadřuje, kolik je architektura schopna zpracovat úloh (v závislosti na komunikační režii) vůči výkonu architektury bez komunikační režie.** To znamená, jak architektura ztrácí svoji výkonnost v závislosti na komunikační

režii vůči architektuře, která nemá žádnou režii spojenou s přístupem do paměti (čistý čas stráveným výpočtem úloh). Tento poměr bude mít větší vypovídací schopnost kolik výkonu dostaneme za komunikační režii potřebnou komunikací s pamětí. Větší číslo pak znamená lepší výsledek, s tím, že **hodnota 1 odpovídá maximálně možné výkonnosti bez komunikační režie**.

Následující graf 3 ukazuje poměr mezi výkonem architektur pro dobu trvání úlohy 40 ms. Je patrné, že mezi 2 CPU architekturami dochází k minimálnímu rozdílu závislosti výkonu na komunikační režii. V případě sběrnice je hodnota 0,782 a křížového přepínače 0,768. Nezáleží tak jakým způsobem se procesory budou propojovat s paměťovými moduly. V tomto případě je spíše řešením sběrnice, která je mnohonásobně levnější.

Dále je možné vidět, že architekturu se 4 CPU tvořena sběrnicí, není poznamenána výrazným poklesem výkonu vůči komunikaci s pamětí. Z toho plyne, že dochází k nárůstu výkonu oproti architektuře s dvěma procesory. Hodnota odpovídá 0,781. K poklesu však dochází u propojení s křížovým přepínačem, kde režie spojená s komunikací trvá déle než v případě sběrnice, což se promítne do výsledného počtu zpracovaných úloh. Podstatně větší výkonnost poskytne architektura s osmi procesory s křížovým přepínačem oproti sběrnici. V tomto momentě, čekaní procesoru na sběrnici, je příliš dlouhá, aby byla udržena výkonnost. Dlouhá komunikační režie způsobena čekáním na přidělení sběrnice má za následek ztrátu výkonu. V případě křížového přepínače s osmi procesory, až tak k razantnímu poklesu poměru výkonnosti nedochází. Výkonnost 8 CPU s křížovým přepínači je tak daleko vyšší než z hlediska sběrnicového provedení.

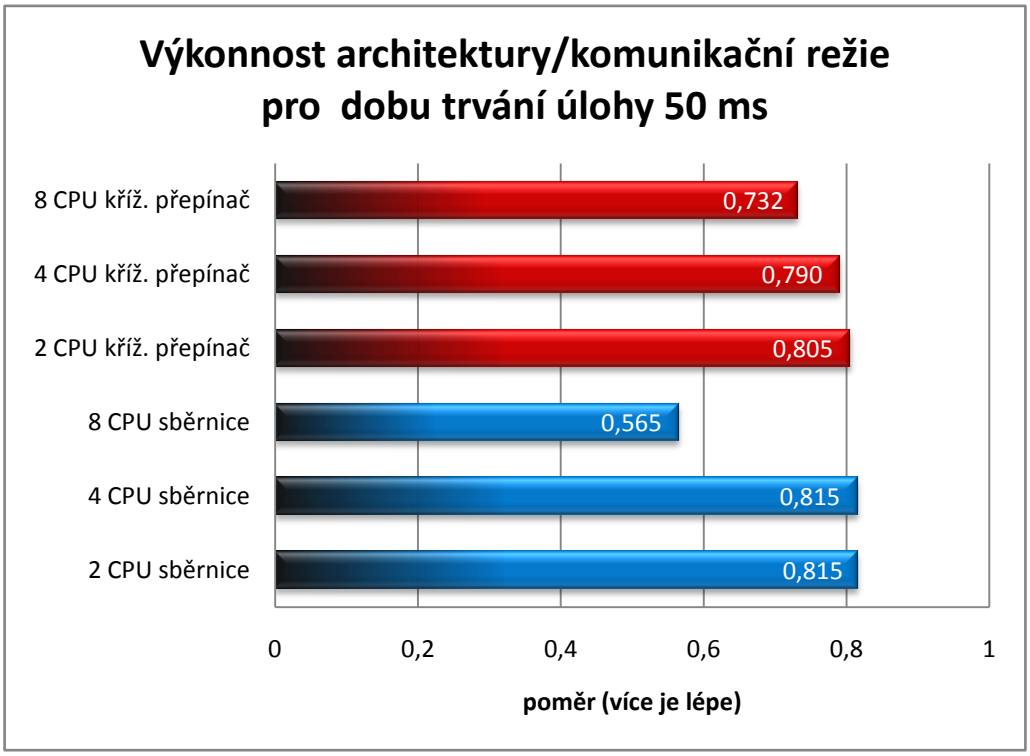
Z hlediska sběrnicové architektury se jeví pro 2 - 4 CPU jako lepší řešení, avšak dalším zdvojnásobením procesorů a pamětí dojde k velkému snížení výkonu. Právě s růstem škálovatelnosti se jeví architektura s křížovým přepínačem jako lepší.



Graf 3: Výkonnost architektury/komunikační režie pro dobu trvání úlohy 40 ms. Zdroj vlastní.

Graf 4 ukazuje výkonnost architektur v závislosti na komunikační režii pro dobu trvání úlohy 50 ms. Poměr výkonu s komunikační režií je u architektury se sběrnicí 2 CPU a 4 CPU téměř stejná. Z toho vyplývá, že nedochází ke komunikačním ztrátám. Rozdíl naznačuje 4 CPU architektura mezi sběrnicí a křížovým přepínačem. Čtyřprocesorový systém tvořený sběrnicí se stává výkonnější vůči křížovému přepínači. To je sice způsobeno minimální komunikační režií, ovšem ta se projeví na výkonnosti. Propustnost propojovací sítě se především projevuje v 8 CPU architektuře, kde se doba na přidělení sběrnice sice zmenšuje z hlediska sběrnice, ale nedostatečně vůči křížovému přepínače.

V případě architektury s osmi procesory je poměr výkonu s komunikační režií nejvíce viditelný. Je patrné, že režie, která ovlivňuje dobu zpracování procesoru je z hlediska menší škálovatelnosti přijatelnější pro sběrnici. S růstem prvků v systému však roste režie, která způsobí, že sběrnice přestane být tak rychlá, aby byla schopna udržet žádosti procesorů na její přidělení.



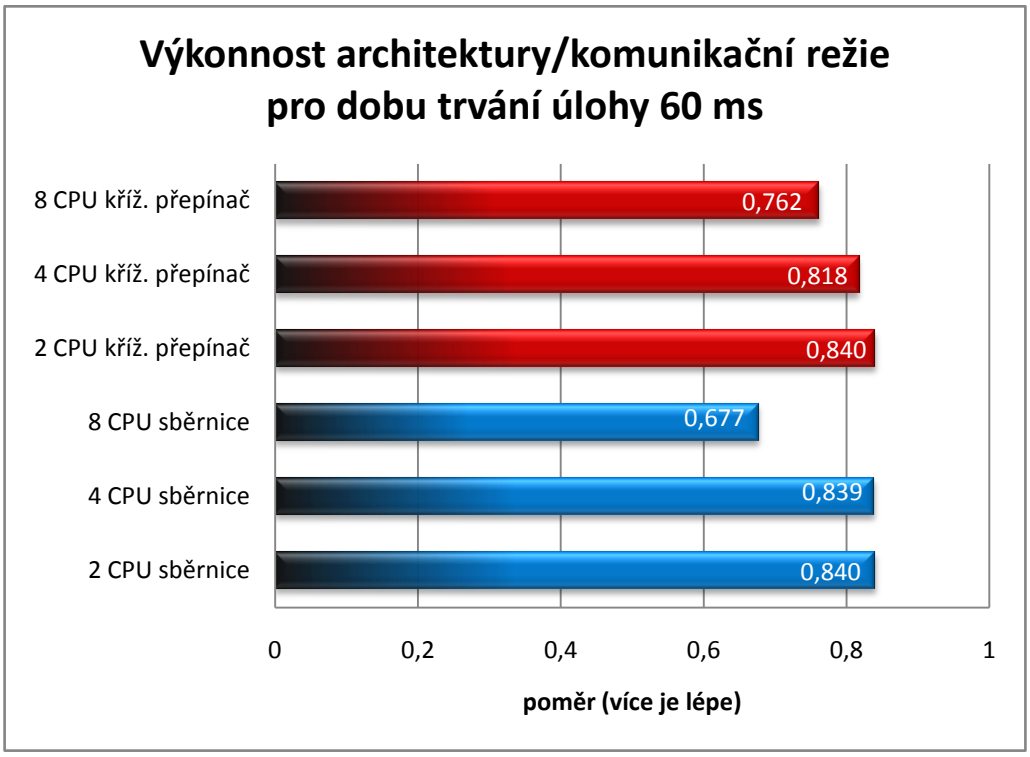
Graf 4: Výkonnost architektury/komunikační režie pro dobu trvání úlohy 50 ms. Zdroj vlastní.

Výkonnost architektury v závislosti na komunikační režii s dobou trvání 60 ms znázorňuje graf 5. Z grafu je vidět, že z hlediska 2 CPU se sběrnicí a křížovým přepínačem je poměr výkonu a komunikační režie téměř stejný. V tomto případě nemá komunikační režie vliv na výkonnost. Pokud dojde k zdvojnásobení počtu procesorů a pamětí je vidět, že tento nárůst způsobí pokles výkonnosti architektury se 4 CPU tvořenou křížovým přepínačem. Tento pokles způsobený režií není tak výrazný, ale z hlediska výkonnosti se jeví jako lepší řešení.

Architektura s 8 CPU, která je tvořena křížovým přepínačem, nezaznamenává velký pokles výkonnosti vzhledem k architektuře se 4 CPU. S porovnáním se sběrnicí dochází k poklesu výkonnosti, ale tento rozdíl není tak veliký. Je možné konstatovat, že problém škálovatelnosti systému se v případě křížového přepínače jeví jako lepší řešení. V rámci 8 CPU je vidět, že náročnost úlohy zmenšuje rozdíly dle zadaných parametrů modelu. Stále však možné tvrdit, že režie komunikace přes sběrnici s pamětí je příliš vysoká oproti řešení křížového přepínače.

Obecně pokud v modelu budeme zvyšovat náročnost zpracování úlohy, sběrnice se stane propustnější a bude se tak přibližovat výkonnosti křížového přepínače.

V případě opačném bude docházet k tomu, že sběrnice nebude na tolik rychlá, aby byla přidělena všem žádajícím procesorům.



Graf 5: Výkonnost architektury/komunikační režie pro dobu trvání úlohy 60 ms. Zdroj vlastní.

5 Hraniční výkonnost architektury

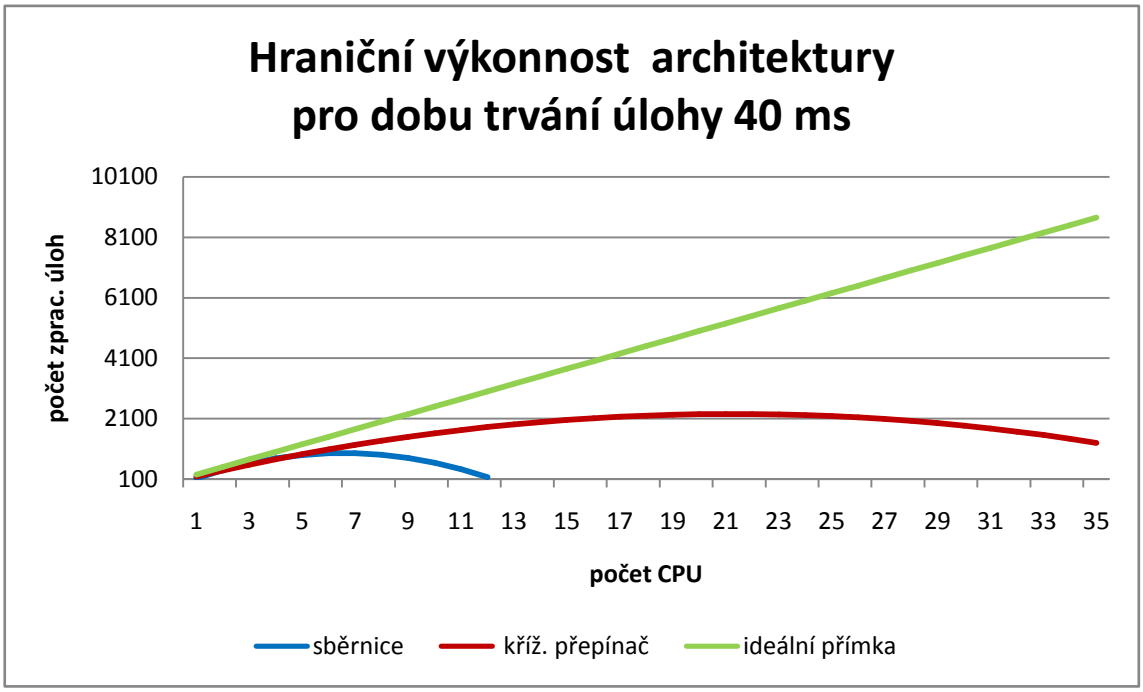
V této části bude navázáno na předchozí kapitolu. Bude zde zkoumána hraniční výkonnost architektury pro každý typ úlohy. To znamená, že je hledána hranice (vrchol), do kolika procesorů je architektura stále výkonná. Právě hraniční možnosti architektury ukáže do jaké míry je škálovatelnost systému hranicí pro sběrnici a křížový přepínač vhodná.

Dosažitelnost architektury byla odhadovaná na základě dosažených hodnot simulací modelů v programu HPSim. Měřítkem pro odhad byl počet tokenů v místě *Execute*, kde byl sledován počet zpracovaných úloh. **Ideální přímka** grafu odpovídá výkonnosti, kterou by dosáhly procesory, pokud by neexistovala žádná režie komunikace s pamětí. Jednalo by se pouze o čistý čas zpracování úloh procesorem.

5.1 Hraniční výkonnost SMP pro dobu trvání úlohy 40 ms

Odhadovaná výkonnost architektury pro dobu trvání úlohy 40 ms je znázorněna na grafu 6. V situaci, kdy úloha zpracování trvá 40 ms, je vrchol hraniční výkonnosti architektury dosažen u 7 procesorů se sedmi paměťovými moduly. Jak jsme mohli vidět v kapitole 4 , tento fakt je způsoben velkou komunikační režií. Dlouhá komunikační režie tak způsobuje, že velký počet procesorů, který čeká na přidělení sběrnice, způsobí pokles výkonnosti architektury. Odhadovaná křivka hraniční výkonnosti sběrnice odpovídá rovnici $y = -27{,}33x^2 + 359x - 217{,}6$.

Na rozdíl od sběrnice, křížový přepínač tuto dosažitelnost silně převyšuje. Tento předpoklad už byl naznačen v kapitole 4 , kdy komunikační režie, která je způsobena přepínacími prvky v síti, dovoluje systému větší propustnost a zároveň tak dochází k zvýšení výkonu. Odhadovaná křivka s křížovým přepínačem odpovídá rovnici $y = -5{,}041x^2 + 214{,}2x - 24{,}33$. Dosažitelnost dle zadaných parametrů tak dosahuje vrcholu kolem 21 procesorů, což 3x více převyšuje architekturu tvořenou sběrnicí.

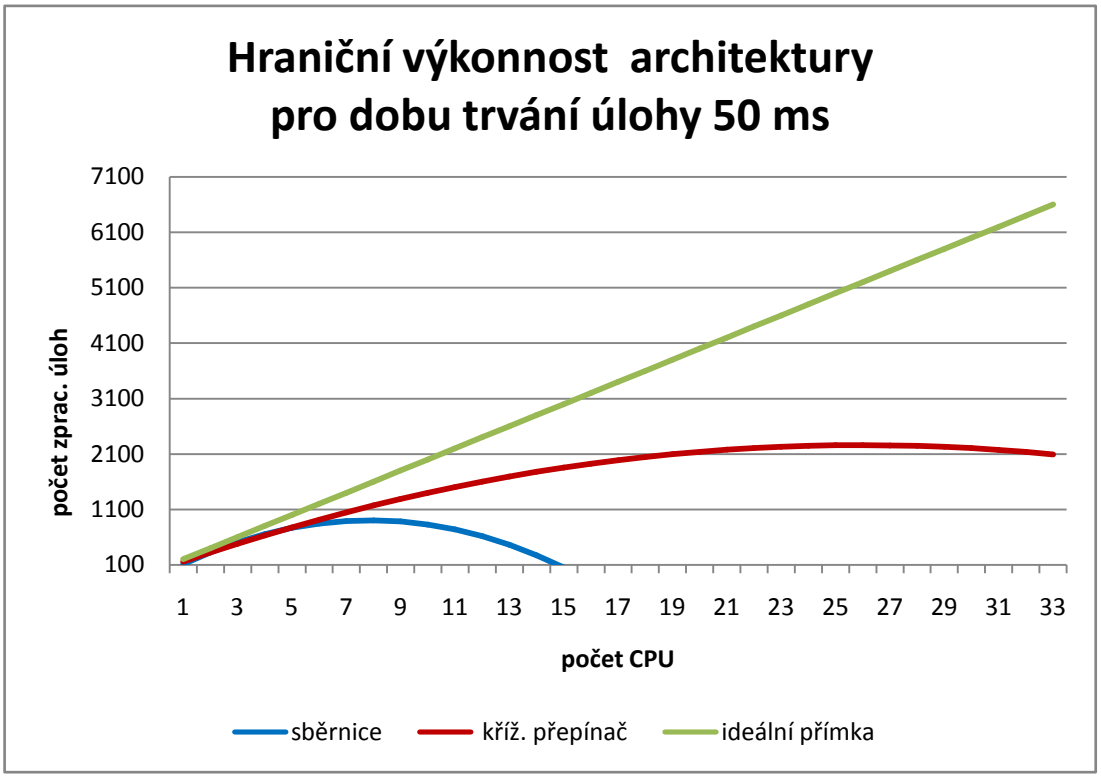


Graf 6: Hraniční výkonnost architektury pro dobu trvání úlohy 40 ms. Zdroj vlastní.

5.2 Hraniční výkonnost SMP pro dobu trvání úlohy 50 ms

Hraniční výkonnost SMP pro dobu trvání úlohy 50 ms znázorňuje graf 7. Z hlediska sběrnice je odhadovaný vrchol výkonnosti s 8 procesory. Zvýšením doby zpracování úlohy tak došlo k mírnému zvýšení hraniční výkonnosti. Dá se však konstatovat, že komunikační režie je stále vysoká, a proto s růstem škálovatelnosti systému nedochází k výraznému zvýšení výkonnosti. Odhadovaná křivka hraniční výkonnosti pak odpovídá rovnici $y = -16{,}79x^2 + 264{,}2x - 135{,}3$. Počet zpracovaných úloh se pohybuje kolem 900 zpracovaných úloh.

Zvýšením náročnosti úlohy tak došlo k zvýšení počtu procesorů za předpokladu téměř neměnného počtu zpracovaných úloh. Křížový přepínač dosáhne svého maximálního vrcholu ve 26 procesorech. Tím, že se zvýší náročnost úlohy, dochází k menší latenci. Vrchol počtu zpracovaných úloh tak dosáhne na 2250 zpracovaných úloh.

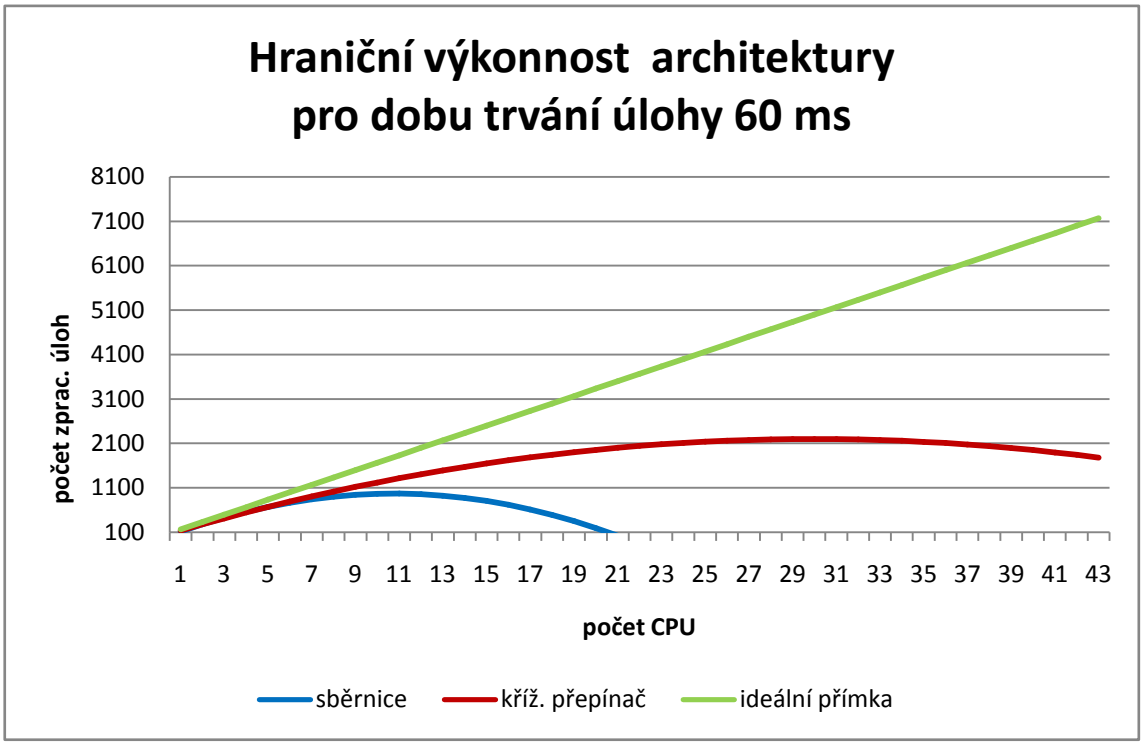


Graf 7: Hraniční výkonnost architektury pro dobu trvání úlohy 50 ms. Zdroj vlastní.

5.3 Hraniční výkonnost SMP pro dobu trvání úlohy 60 ms

Situaci, kdy dosažitelnost architektury dosahuje svého maxima pro náročnost úlohy 60 ms, znázorňuje graf 8. Architektura, která je řešena sběrnicí dosáhne vrcholu 11 procesorů. S rostoucí škálovatelností systému dochází k snížení počtu zpracovaných úloh. Dá se konstatovat, že v této fázi se komunikační režie začíná zvyšovat a procesory stráví více času naslouchání na sběrnice a následným čekáním na přidělení sběrnice. Počet zpracovaných úloh přesáhne mírně 900 zpracovaných úloh a odhadovaná rovnice křivky odpovídá $y = -9{,}041x^2 + 194{,}2x - 72{,}33$.

V případě odhadované křivky křížového přepínače dosáhne svého vrcholu v bodě 30 procesorů. Maximální počet zpracovaných úloh 30 procesory je 2199. Další zvýšení počtu procesorů a pamětí má za následek pokles výkonu, poté začne docházet k větším nárokům na komunikační režii a procesory začnou zpracovávat méně úloh.



Graf 8: Hraniční výkonnost architektury pro dobu trvání úlohy 60 ms. Zdroj vlastní.

Závěr

Diplomová práce je zaměřena na problematiku paralelních systémů. Tato problematika je modelována pomocí Petriho sítí. Hlavním cílem práce bylo namodelovat činnost SMP se sdílenou pamětí pomocí Petriho sítě. Pro modelování Petriho sítí byl zvolen program HPSim. I když tento software není dál vyvíjen, poskytuje velmi příjemné prostředí, dobrou animaci a především export stavů simulace do prostředí Microsoft Excel.

V rámci práce byly vytvořeny modely SMP pomocí propojovací sítě sběrnice a křížového přepínače. Co se týká křížového přepínače, ten byl uveden, jako jedna z alternativ propojení procesorů s pamětí, která není v komerční oblasti multiprocesorové architektury široce používána.

Dále bylo nutné navržené modely ověřit, zda splňují požadované vlastnosti. Především se jednalo o vlastnost živosti sítě. Živost a další doplňující vlastnosti byly ověřeny pomocí programu PIPE3.0.

Pro analýzu chování modelů byly stanoveny tři typy úloh, které se lišily délkou doby zpracování jedné úlohy procesorem. Namodelované architektury byly pomocí těchto úloh analyzovány v simulačním čase 10 000 ms. Z výstupních dat modelu, která se zpracovávala v programu Microsoft Excel, byla analyzována komunikační režie, která odpovídala době, kterou procesor strávil čekáním přístupu do sdílené paměti. Dále byl určen poměr výkonu architektury vůči komunikační režie, který odpovídal počtu zpracovaných úloh (s komunikační režii) vůči zpracovaným úlohám (bez komunikační režie). Nakonec byly určeny hraniční možnosti architektury, které znamenalo nalezení maxima počtu CPU, kdy bude architektura stále výkonná.

Z výsledků bylo dosaženo premise, že s rostoucí škálovatelností systému se projeví nedostatečná propustnost sběrnice. Tento fakt byl způsoben tím, že komunikace přes sběrnici nedovoluje komunikovat více procesorům zároveň. To se také promítlo do hraniční výkonnosti architektury, kdy sběrnice dosáhla vrcholu u 11 CPU oproti křížovému přepínači, který měl maximum v 30 CPU. Výsledky byly stanoveny dle zadaných simulačních parametrů modelu. V případě nastavení jiných hodnot na přechodech se dosahované hodnoty mohou lišit. Obecně lze říci, že prodloužením doby

náročnosti zpracovaných úloh, se sběrnice stávala propustnější a tím docházelo k snížení rozdílů mezi sběrnicí a křížovým přepínačem.

Modely, které byly navrženy pomocí Petriho sítí lze dále rozšířit nebo zkoumat s jiným typem architektury. Nabízí se možnost srovnávat výkonnost cluster systémů, které jsou především propojeny statickými sítěmi. Petriho sítě nabízí široké možnosti modelování různých dynamických systémů a jejich následné zkoumání vlastností.

Cíl práce, tedy namodelovat spolupráci SMP se sdílenou pamětí (propojené sběrnicí a kříž. přepínačem) pomocí Petriho sítí a provedení následné analýzy architektur, byl splněn.

Použitá literatura

[1] ABD-EL-BARR, Mostafa; EL-REWINI, Hesham. *Advanced computer architecture and parallel processing*. New Jersey : WILEY INTERSCIENCE, 2005. 287 s. ISBN 0-471-4670-5.

[2] ADDISON, Wesley. *Introduction to Parallel Computing*. Second Edition. Harlow : The Benjamin/Cummings Publishing Company, 2003. 856 s. ISBN 0-201-64865-2.

[3] BLAISE, Barney. *Introduction to Parallel Computing* [online]. 2010 [cit. 2011-03-15]. Introduction to Parallel Computing. Dostupné z WWW: <https://computing.llnl.gov/tutorials/parallel_comp/>.

[4] ČEŠKA, Milan, et al. Petriho sítě : Studijní opora [online]. Brno : VUT v Brně, 2009 [cit.2011-04-19]. Dostupné z WWW: <http://www.fit.vutbr.cz/study/courses/PES/public/Pomucky/PES_opora.>.

[5] HENNESSY, John L.; PATTERSON, David A. *Computer architecture : a quantitative approach*. San Francisco : Morgan Kaufmann, 2007. 912 s. ISBN 978-0-12-370490-0.

[6] HLAVIČKA, Jan. *Architektura počítačů*. Praha : ČVUT, 1994. 220 s. ISBN 80-01-01847-4.

[7] *Introduction to Petri Net Theory* [online]. 2010 [cit. 2011-01-04]. Introduction to Petri Net. Dostupné z WWW: <http://cc.ee.ntu.edu.tw/~yen/papers/yen.pdf>.

[8] MARKL, Jaroslav. Petriho sítě [online]. Ostrava : VŠB - Technická univerzita Ostrava , 1998 [cit. 2010-06-24].

[9] *Petri nets* [online]. 2007 [cit. 2011-04-21]. Petri Nets I. Dostupné z WWW: <www.cs.nyu.edu/courses/fall07/G22.2631-001/petrinet.ppt>.

[10] RAUBER, Thomas ; RÜNGER, Gudula. *Parallel Programming : for Multicore and Cluster Systems*. Berlin : Springer, 2010. 463 s. ISBN 978-3-642-04817-3.

[11] *SMP* [online]. 2003 [cit. 2011-04-21]. Multiprocesorové a paralelní počítačové systémy. Dostupné z WWW: <http://noel.feld.cvut.cz/vyu/scs/prezentace2003/MultiProc/>.

[12] TANENBAUM, Andrew Stuart. *Modern operating systoms*. New Jersey : Prentice Hall PTR, 1992. 728s. ISBN 0-13-031358-0.

[13] *Teorie grafu* [online]. 2010 [cit. 2011-04-15]. Základní pojmy teorie grafu. Dostupné z WWW: <http://www2.ef.jcu.cz/~jfrieb/tspp/data/teorie/grafy.pdf>.

[14] GRUBER, Ralf; KELLER, Vincent . *HPC@GreenIT : Green High Performance Computing Methods*. Berlin : Springer, 2009. 221 s.

[15] *Wikipedia, the free encyclopedia* [online]. 2007 [cit. 2011-04-19]. SISD. Dostupné z WWW: <http://en.wikipedia.org/wiki/File:SISD.svg>.

[16] *Wikipedia, the free encyclopedia* [online]. 2007 [cit. 2011-04-19]. SIMD. Dostupné z WWW: <http://en.wikipedia.org/wiki/SIMD>.

[17] *Wikipedia, the free encyclopedia* [online]. 2007 [cit. 2011-04-19]. MISD. Dostupné z WWW: <http://en.wikipedia.org/wiki/File:MISD.svg>.

[18] *Wikipedia, the free encyclopedia* [online]. 2007 [cit. 2011-04-19]. MIMD. Dostupné z WWW: <http://en.wikipedia.org/wiki/File:MIMD.svg>.

Seznam zkratek

ALU	Aritmetic Logic Unit (Aritmeticko Logická Jednotka)
CPU	Central Processing Unit
NUMA	Non-Uniform Memory Access
MIMD	Multiple Instruction, Multiple Data
MINs	Multistage Interconnection Networks (Víceúrovňové propojovací sítě)
MISD	Multiple Instruction, Multiple Data
PS	Petriho sítě
PU	Processor Unit
SIMD	Single Instruction, Single Data
SISD	Single Instruction, Single Data
SMP	Symmetric multiprocessing
UMA	Uniform Memory Access

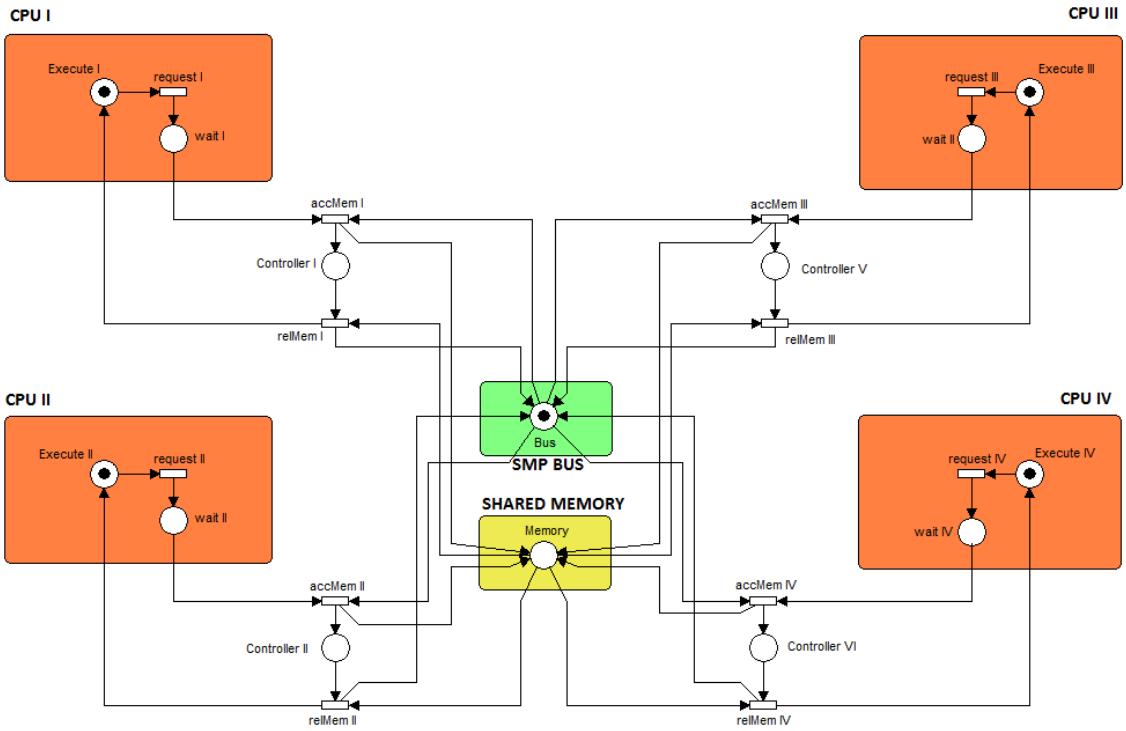
Seznam obrázků

Seznam grafů

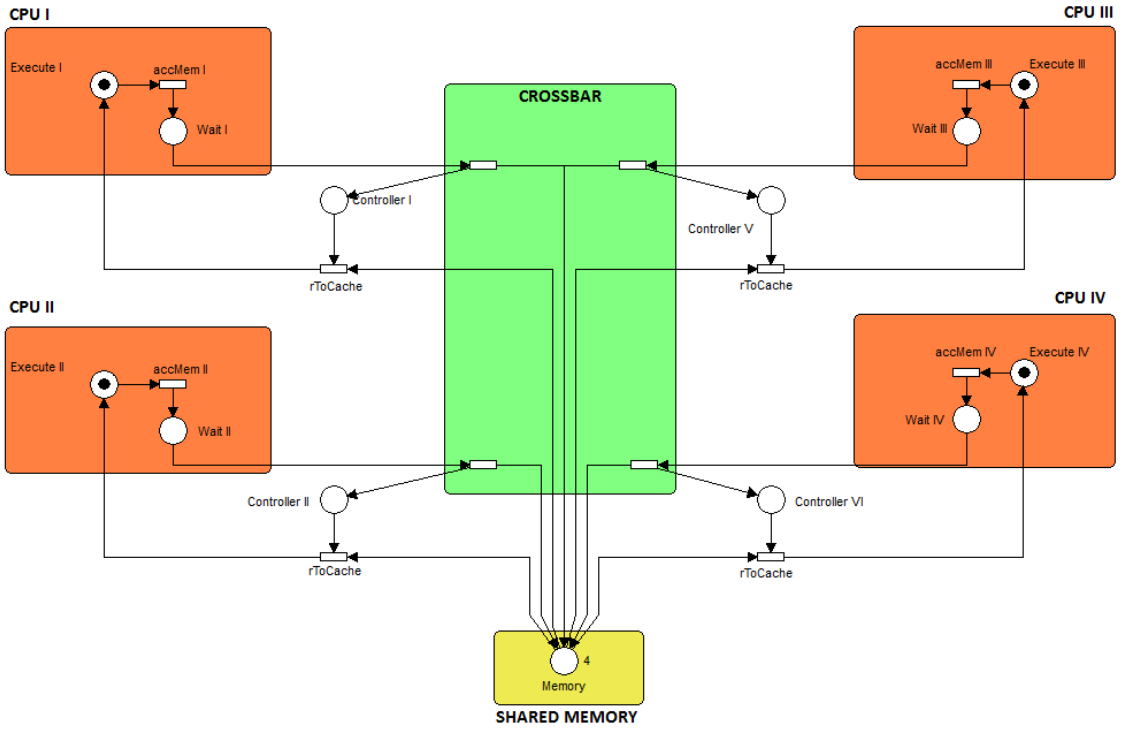
Seznam tabulek

Seznam příloh

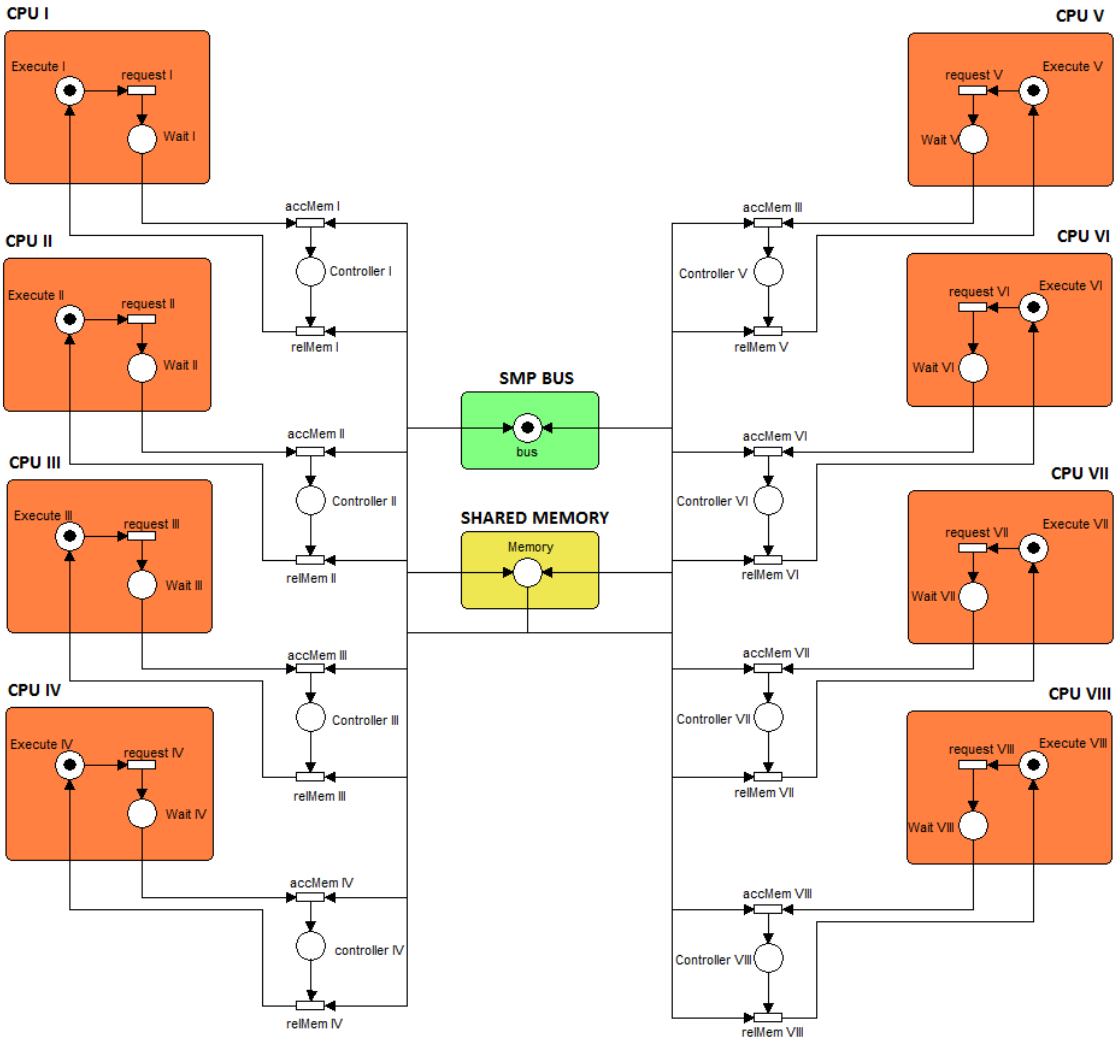
PŘÍLOHA Č. 1: HPSim - SMP 4x4 (sběrnice). Zdroj vlastní.



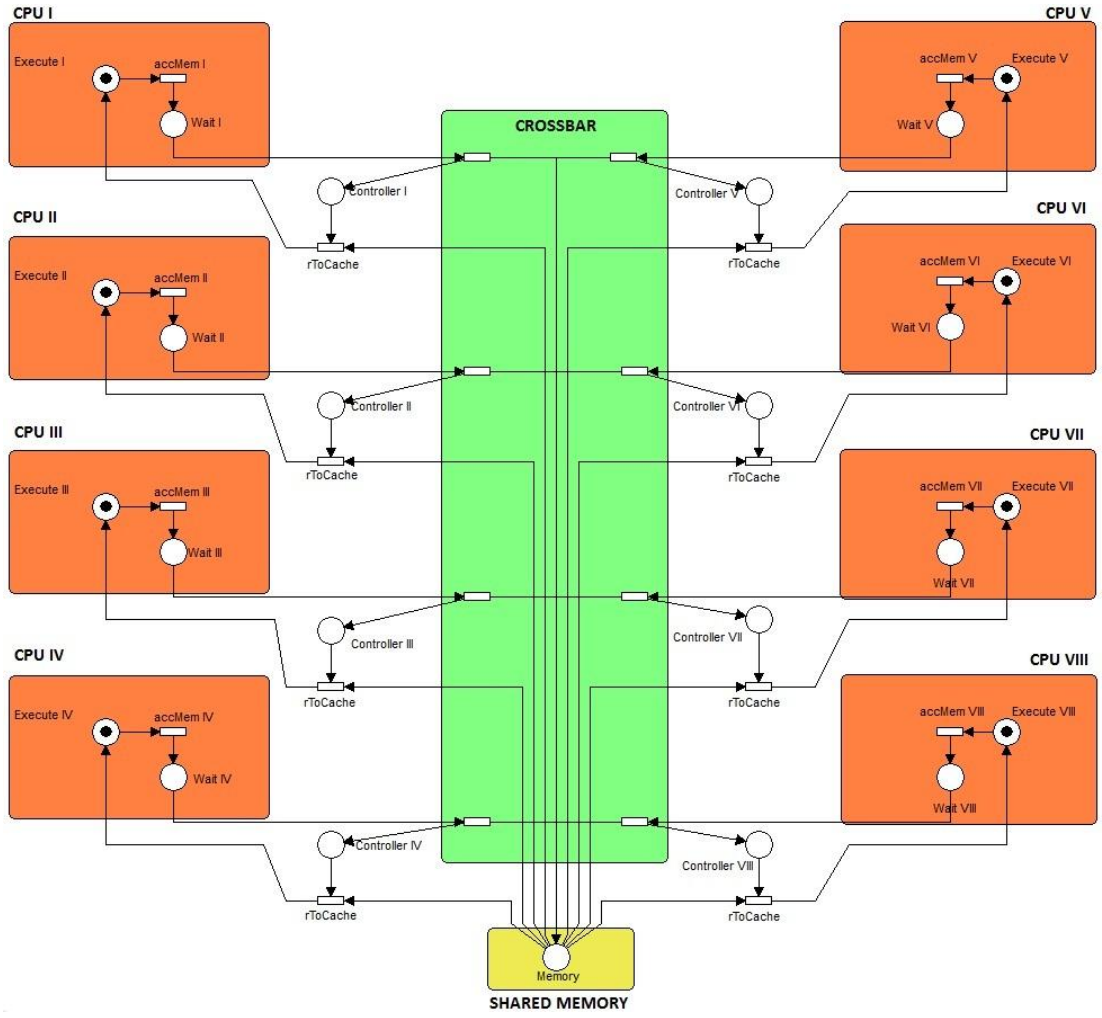
PŘÍLOHA Č. 2: HPSim - SMP 4x4 (kříž. přepínač). Zdroj vlastní.



PŘÍLOHA Č. 3: HPSim - SMP 8x8 (sběrnice). Zdroj vlastní.



PŘÍLOHA Č. 4: HPSim - SMP 8x8 (kříž. přepínač). Zdroj vlastní.



PŘÍLOHA Č. 5: Obsah CD

CD obsahuje:

- HPSim – program na simulaci Petriho sítí
- PIPE3.0 – program na simulaci Petriho sítí
- Složka Modely - obsahuje modely, které byly vytvořeny v programu HPSim
- Složka Excel – obsahuje zpracovaná data z programu HPSim