XII международная конференция
CEE-SECR / РАЗРАБОТКА ПО
Москва, 28-29 октября 2016

Виртуалтрединг: новая мета-архитектура компьютеров для прямого тонко гранулированного аппаратного мультипрограммирования

Ефимов А.И., к.т.н.
фрилансер, eai_andr_kb@mail.ru

Цель и результаты

Современные средства автоматизации проектирования аппаратуры ВС на основе HDL (например, си-подобный verilog вместо схемного описания аппаратуры) сняли языковый барьер, препятствовавший прямому участию системных программистов-разработчиков ОС в оптимизации аппаратуры. Недорогие макетные платы и доступные для свободного либо свободного к использования в академических целях проекты СнК, в частности проект MIPSfpga, облегчили участие системных программистов в такой оптимизациии и создали все необходимые условия для разработки развитых свободных СнК. Такие СнК со временем могут занять в аппаратной нише IT столь же значительное место, которое заняла свободная ОС Linux в программной нише.

Целью работы является представление сообществу разработчиков ОС и схемотехников виртуалтрединговой мета-архитектуры (ВтМа), которая в свободных СнК может обеспечить высокоэффективное прямое тонко гранулированное аппаратное мультипрограммирование.

Результатом работы является реализованный автором (системным программистом, осваивающим около 3х лет verilog-моделирование процессоров включая год макетировния на ПЛИС) проект основных архитектурных решений ВтМа в инфраструктуре MIPSfpga.

Далее на 8 слайдах приведен краткий анализ развития и недостатков существующих ВС, сопоставлена огранизация мультипрограммирования в них и в предлагаемой ВтМа, а также описана организация макета сопроцессора ВтМа и характеристики его быстродействия.

Современные вызовы в области IT

По оценке Бартона Смит, архитектора экстремально многонитевых *(далее ХМТ)* систем HEP, HORIZON,Tera-Mta, Cray-XMT: [1-4] “ … Современные микропроцессоры также плохи для параллелизма общего назначения *(далее ПВОН)*, как и динозавры, которые на них охотятся. ... требуются радикальные решения, способные вывести индустрию IT из спирали специализации, в которой высокопроизводительными считаются вычисления, которые хорошо выполняются существующими высокопроизводительными системами” [5,6] .

ПВОН сейчас - основная ниша по востребованности и объему вкладываемых средств. Их свойства - обработка интенсивного потока транзакций при ограничениях реального времени в следующих приложениях:

- on-line аналитика (OLAP) и обработка транзакций (OLTP) в среде больших СУБД;
- платежных и поисковые системы;
- управление сложными объектами с высокой надежностью при ограничениях жесткого (микросекунды) реального времени.

Предлагаемая ВтМа по существу является логическим завершением ХМТ и может обеспечить радикальное повышение эффективности ВС для ПВОН за счет снижения всех рассмотренных на следующем слайде видов латентности.

Коэффициент полезного действия ВС

- “Четыре всадника Апокалипсиса” (Томас Стерлинг [7]) вызывают состояния латентности, снижающей эффективность высокопроизводительных ВС:
- накладные расходы - работы по управление параллельностью;
- латентность - простои при взаимодействии распределенных модулей;
- конкуренция - бесполезная активность при соперничестве за ресурсы;
- голодание - простои из-за слабого параллелизма и дисбаланса загрузки - в частности, из-за стены параллелизма уровня инструкций (ILP) в суперскалярных и VLIW процессорах.

Поскольку ВС являются аппаратно-программными комплексами, естественно ввести понятие обобщенной латентности (ОЛ) компонентов и на его основе распространить общетехнический показатель качества — коэффициент полезного действии (КПД) - в область IT, определив его простой формулой:

КПД = 1 - Сол / С, где С и Сол – стоимости аппаратуры при выполнении работы, а Сол - части в состоянии ОЛ.

Определение Сол не так очевидно, как определение неиспользованной тепловой энергии в термодинамике, но формализация позволит количественно оценивать оптимальность ВС, предложить IT-аналог цикла Карно и разработать оптимальные архитектуры.

2D виртуализация ВС – основа сокращения латентности

1st D: Метаархитектура виртуализации активностей — от 10 виртуальных процессоров в периферийном сопроцессоре CDC6600 [8] (нейтрализована латентность ферритовой памяти) до 128-нитевых XMT-систем [1-4]

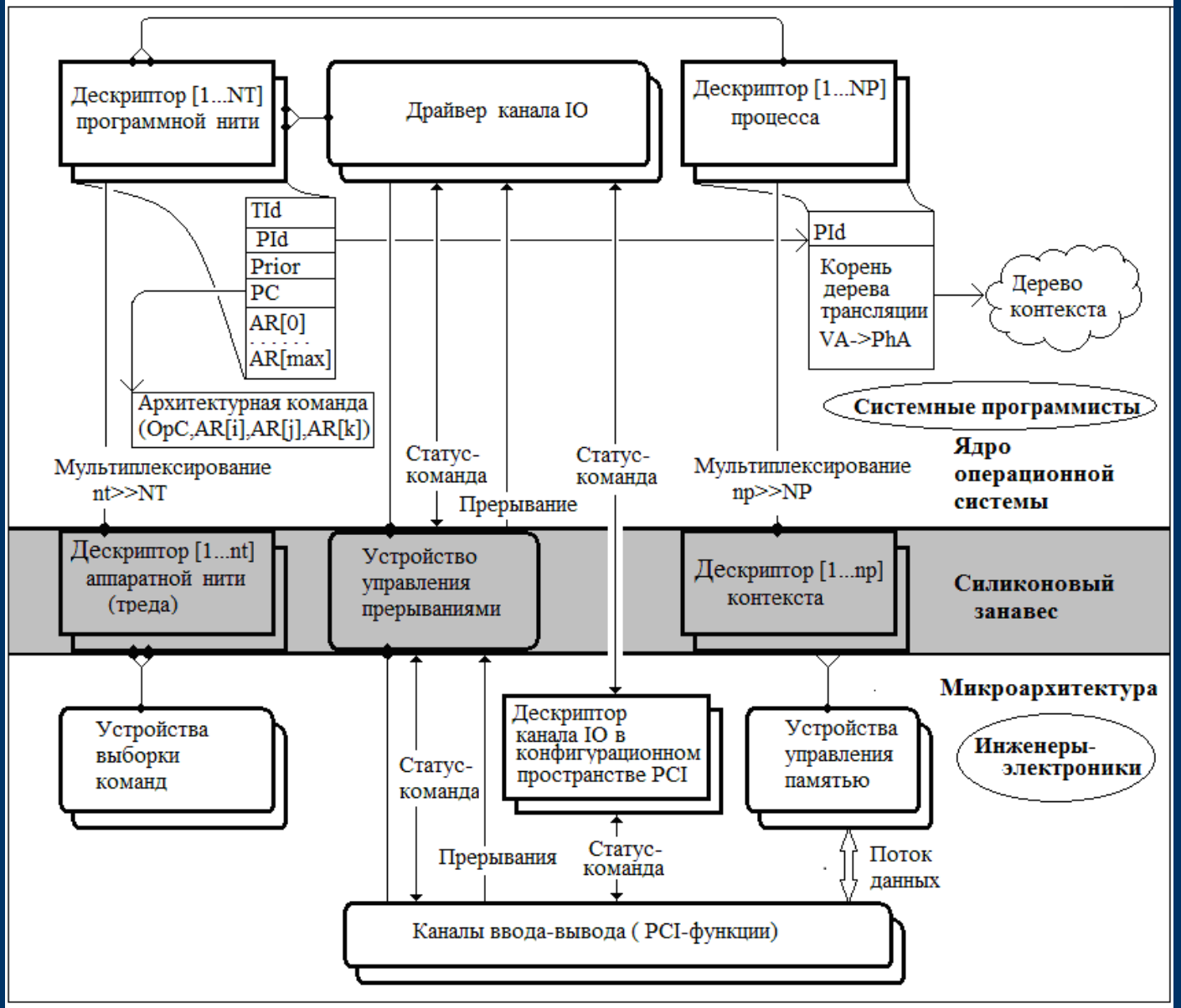
2nd D: Метаархитектура виртуализации памяти - от грубо гранулированного свопинга 512-байтных страниц [9] до тонко гранулированного, плоть до одного машинного слова, свопинга между кэшами уроней 1-4 и оперативной памятью во всех современных процессорах.

Резкая несбалансировать — в 1st D элементов свопинга не мене чем регистровый файла (РФ) (минимум в x86, средние для MIPS и максимальные для SPARC. Это приводит к высоким накладным расхода при частых переключениях программных нитей на аппаратных, число которых на порядки меньше числа программных.

Разная приоритетность программных нитей, в частности, вычислительных и нитей обслуживания ввода-вывода (IO), потребовала ввода прерываний (таймерных и IO), сильно усложнивших аппаратуру Следуя традиционному принципу борьбы со сложностью ”разделяй и властвуй”, системный программисты и схемотехники разделили свои области ответственности в по-существу общей задаче оптимизации ВС и ввели показанные на следующем слайде Силиконовый Занавес и чрезмерно затратные каноны аппаратно-программного мультипрограммирования.

Важность задачи нейтрализации прерываний было осознано как в старых, так и в новых ВС. В частности в упомянутой выше CDC6600 [8] центральный процессор работал без прерываний ввода-вывода, которые отрабатывал периферийный мультиредовый сопроцессор. В XMT-системах [1-4] их обрабатывали одна или более из 128 нитей, позволяя оставшимся непрерывно заниматься вычислениями.

Грубо гранулированное программно-аппаратное мультипрограммирование



Отношения: **- один к одному** (владелец-объект) **- один ко многим** (владелец- объекты)

Основные черты и эффективность ВтМа

Предлагаемая виртуалтрединговая архитектура (ВтМа) устраняет сложившийся дисбаланс виртуализации (грубая грануляция грануляция отображения виртуальный-физический для активностей) и предлагает более эффективный и, в то же время, простой путь повышения эффективности ВС, в котором Силиконовый Занавес перенесен в более подходящее место — перед выделенными Стерлингом [7] Всадниками Апокалипсиса.

Основу ВтМА составляет транзакционная микроархитектура, в которой:

- грубо гранулированное программно-аппаратное мультипрограммирование нитей заменено тонко гранулированным аппаратным мультипрограммированием транзакций, порождаемых в динамике исполнения архитектурных команд всех нитей;

- отсутствует понятие *аппаратная нить* (синонимы - тред, hardware thread, thread engine, strand) – есть единое для программистов и системотехников понятие *нить*.

- реализовано продвижение одновременно всех программных нитей (системных и пользовательских) в соответствии с их готовностью к исполнению своих работ и возможностями и их выполнить набором тонко гранулированных аппаратных модулей (сети исполнительных устройств, не объединенных общим конвейером).

По существу в ней устранен показанный на предидующем слайде Силиконовый Занавес, которым схемотехники и системные программисты разделили свои области разработок и таким образом увеличили связанную с переключением нитей латентность.

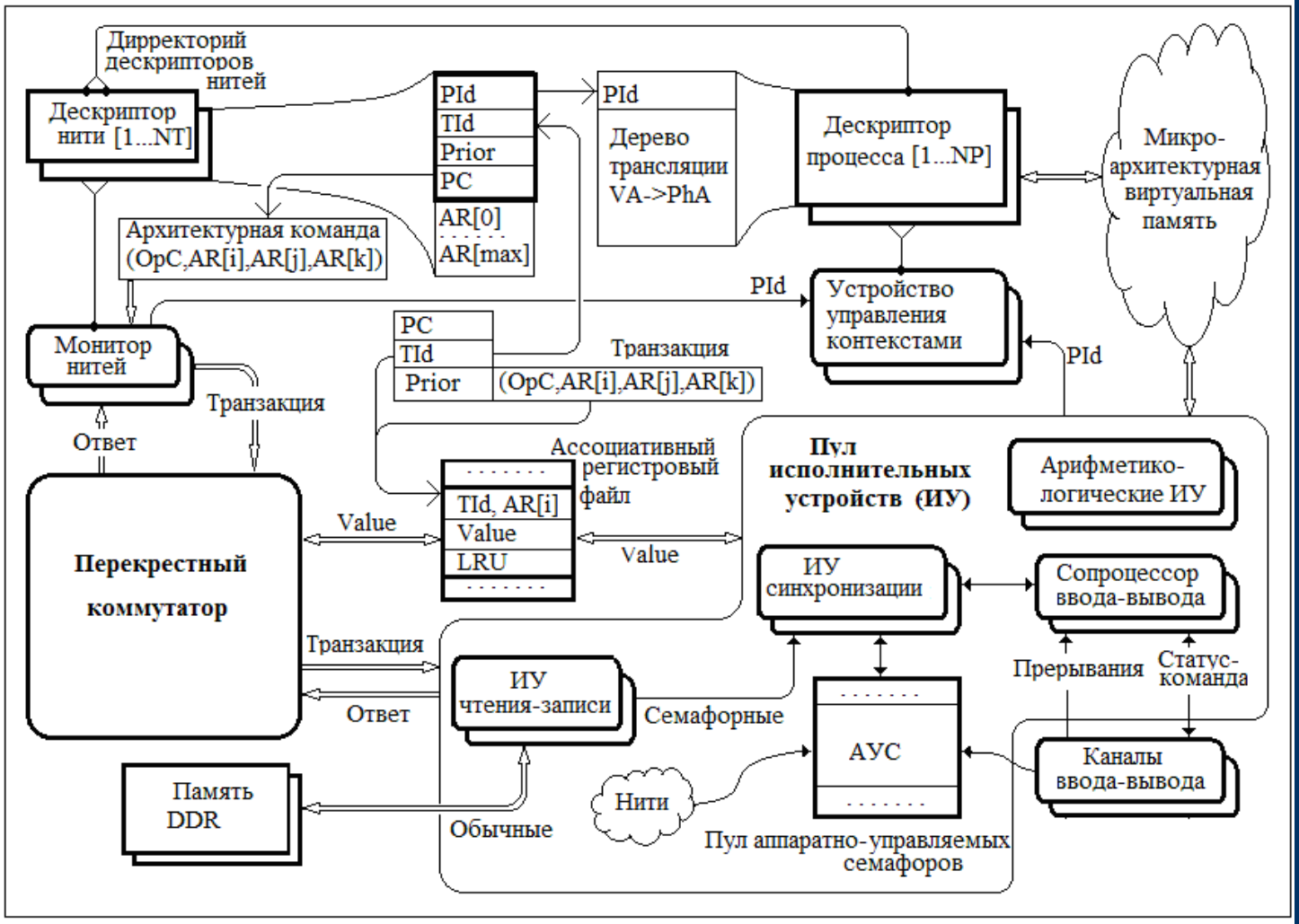
Планируемые этапы разработки ВтМа

На следущем слайде показана организация автономной СнК на основе ВтМа, сдедущей за ним паре - организация макета ВтМа-сопроцессора для повышения эффективности хост-процессора MIPSfpga при исполнении ПВОН и представление спруктуры распараллеливания информационных потоков в виде фрагмента verilog-кода.

Создание автономной СнК на основе ВтМа - большой нукоемкий стартап. Его патентную чистоту может обеспечить патент Республики Беларусь (Ефимов, [10]) (приоритет от 2004.09.16). Ближайшим аналогом является более поздняя работа [11] (Oehmke, November, 2015), описывающая частное решение задачи виртуализации регитрового файла.

В то же время, разраоботка ВтМа-сопроцессора может быть уже сейчас начата в академической среде. На следующей (за verilog-кодом) паре показаны простейшая схема подготовки boot-кода в виде verilog-текста и трасса выборки команд в системе Icarus verilog. Далее показана времення диаграмма реализации в макете ВтМа-сопроцессора средств мультитрединговой обработки (8 нитей), на основе которых будет реализован мультитрединг. Эти результаты воспроизведены при исполнении проекта на плате DE0-CVA, но частоту следования запросов (rq*) пришлось уменьшить вдвое, выдавая их только по каждому втрому первичному синхроимпульсу частоты 50 MHz . Три последуюих слайда иллюстрируют аппаратные средства синхронизации нитей ВтМа, достаточные для эффективной реализации функций ядра и драйверов ОС без прерываний ввода-вывода и таймера.

Виртуалтрединг: тонко гранулированное прямое аппаратное мультипрограммирование



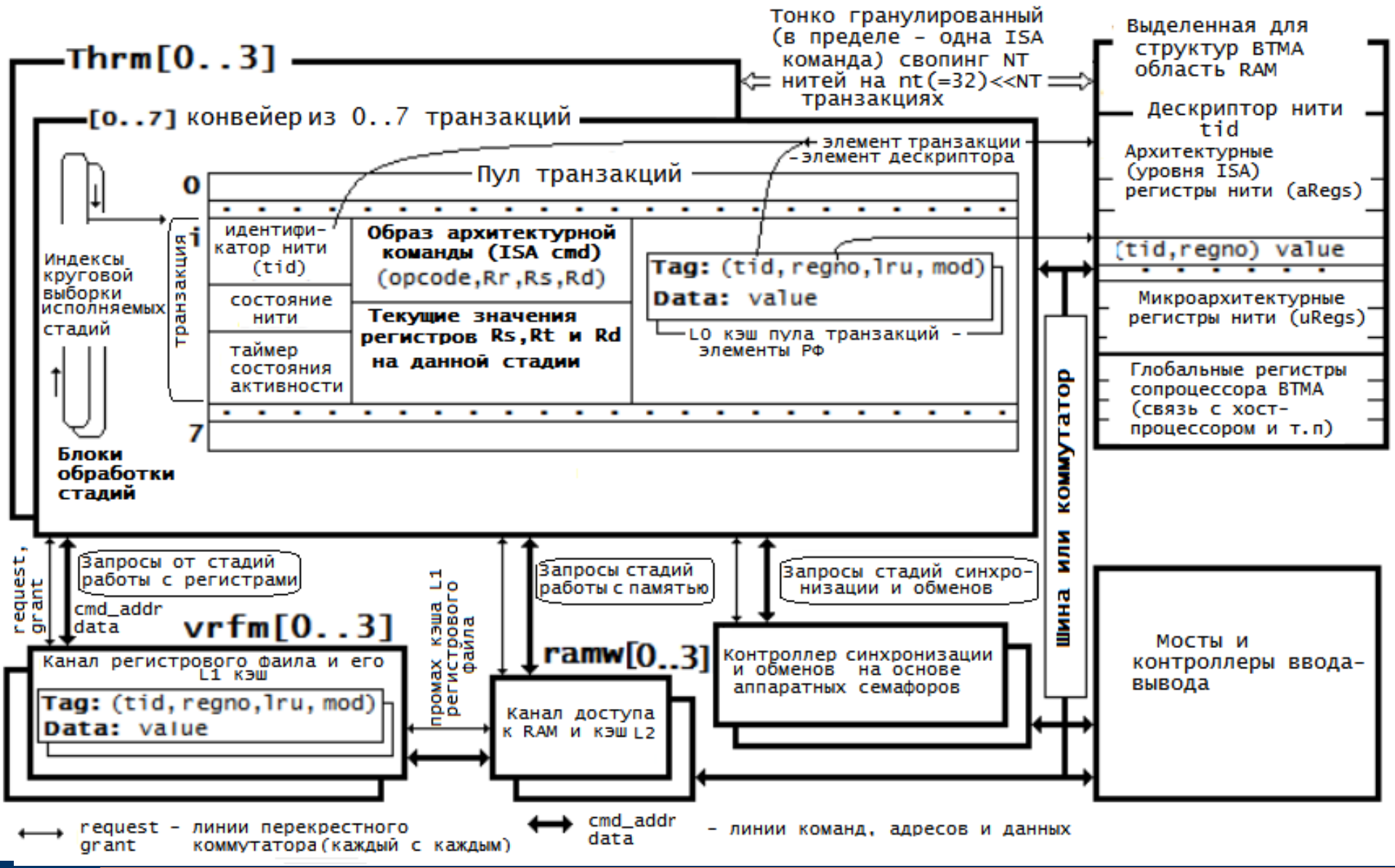
Отношения: - **один к одному** (владелец-объект)

- **один ко многим** (владелец- объекты)

Ассоциативные ссылки: - **по простому ключу**

- **по составному ключу**

Эталонная модель виртуалтредингового сопроцессора



Verilog-представление перекрестных коммутатор и шин макета ВТМА конфигурации 4-4-4
(* – индексы 0:3 эллементов групы сигналов)

```
generate genvar git; for (git=0; git<`NTM; git = git+1) begin : thrm
thrm  #(.ie(git)) thrm [git:git] (.clk(clk), .Edgeno(Edgeno), .ReadyBoot(ReadyBoot)
,.t2v_rqs(t2v_rqs[git]),   .tfv_gnts(tfv_gnts[git])   // trmw MASTER # vrfw crossbar
,.t2r_rqs(tw2r_rqs[git]),  .tfr_gnts(twfr_gnts[git])  // trmw MASTER # ramw crossbar

,.vrfway0*_in_ca_thrm(rfways_in_ca[*]),   .vrfway0*_in_d_thrm (rfways_in_d [*]),
,.vrfway0*_out_thrm (rfways_out  [*]),    .ramway*_in_ca_thrm(ramways_in_ca[*]),
,.ramway*_in_d_thrm (ramways_in_d [*]),  .ramway*_out_thrm  (ramways_out [*]),
 );   end endgenerate // thrm

generate genvar giw; for (giw=0; giw<=`NW-1; giw = giw +1) begin : vrfwm
vrfwm  #(.ie(giw)) vrfwm [giw:giw] (.clk(clk), .Edgeno(Edgeno), .ReadyBoot(ReadyBoot)
,.rqs(wft_rqs[giw]),                          .w2t_gnts(w2t_gnts[giw])
,.w2r_rqs(tw2r_rqs[`NTM+giw]),                .wfr_gnts(twfr_gnts[`NTM+giw])
.ramway*_in_ca_vrfwm(ramways_in_ca[*]),       ,.ramway*_in_d_vrfwm (ramways_in_d [*]),
.ramway*_out_vrfwm  (ramways_out_vrf [*]),
.vrfway_in_ca(rfways_in_ca[giw])              s,.vrfway_in_d(rfways_in_d[giw])
); end   endgenerate // vrfwm

generate genvar gim; for (gim=0; gim<=`NRW-1; gim = gim +1) begin : ramw
ramwm  #(.ie(gim)) ramw [gim:gim] (.clk(clk), .Edgeno(Edgeno), .ReadyBoot(ReadyBoot)
,.tw2r_rqs(rftw_rqs[gim])            ,.r2tw_gnts(r2tw_gnts[gim])
,.ramway_in_ca(ramways_in_ca[gim]), .ramway_in_d (ramways_in_d [gim])
,.ramway_out  (ramways_out  [gim]), .ramway_out_vrf(ramways_out_vrf[gim])
,.rdcd_i(rdcd_i),  .rdcd_cmd(rdcd_cmd), .rdcd_done(rdcd_done)     // BOOT loading
); end   endgenerate // ramwm

generate genvar it,iw ; for (iw=0;iw<`NW;iw=iw+1) begin : ItLoop
for (it=0;it<`NTM;it=it+1) begin : IwLoop
assign wft_rqs[iw][it] =t2v_rqs[it][iw]; assign tfv_gnts[it][iw]=w2t_gnts[iw][it];
end end endgenerate

generate genvar itw,ir ; for (ir=0;ir<`NRW;ir=ir+1) begin : IrLoop
for (itw=0;itw<`ALLRW_RQS;itw=itw+1)   begin : ItwLoop
assign rftw_rqs[ir][itw]=tw2r_rqs[itw][ir]; assign twfr_gnts[itw][ir]=r2tw_gnts[ir][itw];
end end endgenerate
```

Подготовка и исполненеие boot-кода в макете ВТМА

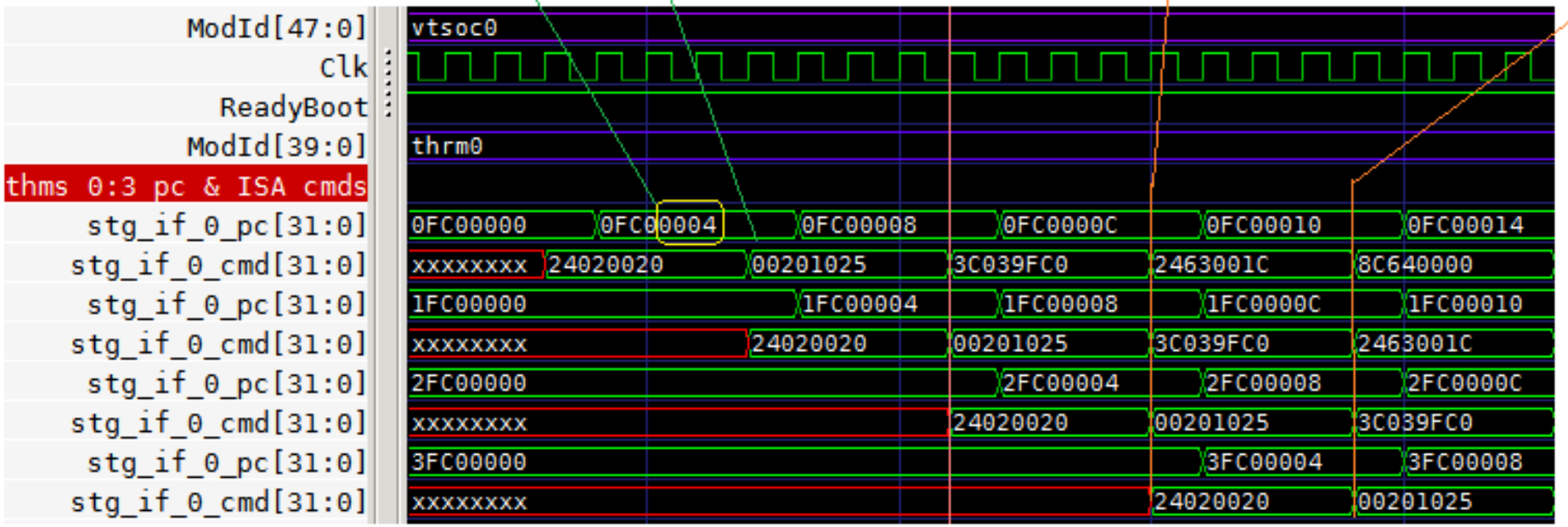
Подготовка boot-кода для макета:

MIPS tool chain: xxx.{ c | s } => xxx.elf => xxx.dis == own utility ==> boot.vh

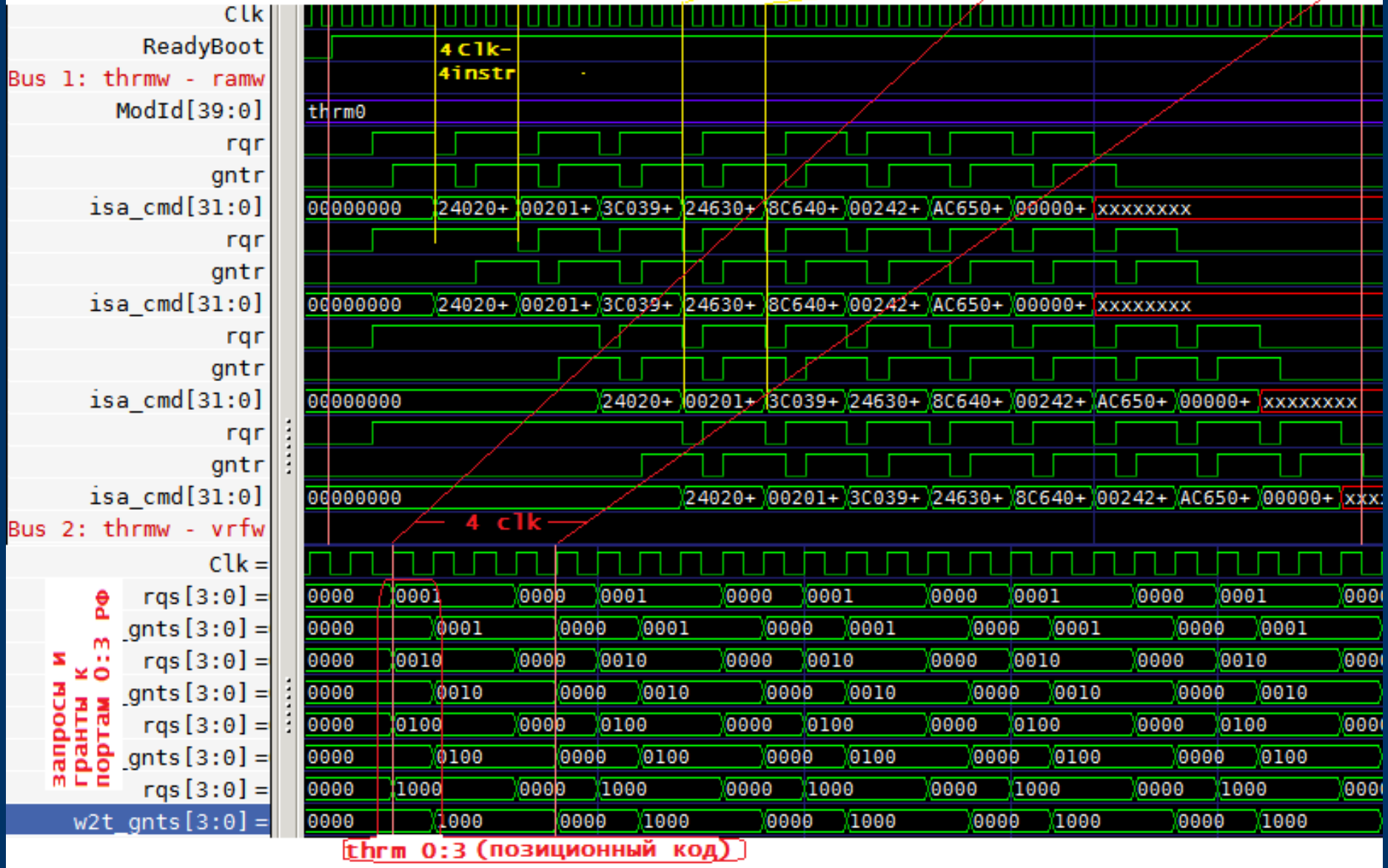
```
wire [15:0] wda = rdcd_i[15:0];  wire [31:0]  rdcd_cmd =
// start():
(wda[15:0]=='h0000) ? 'h24020020 : // w  0000 -> 9fc00000: 24020020 li v0,32
(wda[15:0]=='h0004) ? 'h00201025 : // w  0001 -> 9fc00004: 00201025 move  v0,at
(wda[15:0]=='h0008) ? 'h3c039fc0 : // w  0002 -> 9fc00008: 3c039fc0 lui   v1,0x9fc0
(wda[15:0]=='h000c) ? 'h2463001c : // w  0003 -> 9fc0000c: 2463001c addiu v1,v1,28
(wda[15:0]=='h0010) ? 'h8c640000 : // w  0004 -> 9fc00010: 8c640000 lw a0,0(v1)
(wda[15:0]=='h0014) ? 'h00242820 : // w  0005 -> 9fc00014: 00242820 add   a1,at,a0
(wda[15:0]=='h0018) ? 'hac650000 : // w  0006 -> 9fc00018: ac650000 sw a1,0(v1)
// array1():
(wda[15:0]=='h001c) ? 'h00000001 : // w  0007 -> 9fc0001c: 00000001 movf  zero,zero,$fcc
// Csum: nop
(wda[15:0]=='h0020) ? 'hbd75d842 : // w  0008 -> 00000020 !!! <== Csum !!!
```

Конвейер выборки команд в макете

: 4 экземпляра модуля "монитор нитей" (thrm*)
: обеспечивают выборку 4х команд в 4х тактах



8 транзакций (по 2 в 4х модулях мониторов нитей thrm) реализуют параллелную выборку в 4х тактах 4х команд и 4х обращений к РФ



Синхронизация на основе аппаратно-управляемых семафоров

- Средства синхронизации основаны на использовании аппаратно-управляемых семафоров (АУС), объединяющих функции мутекса и условной переменной современных ОС. Они дополнительно реализуют функцию локального сторожевого таймера для контроля времен работы нитей в критической секции и ожидания входа в нее.

  АУС организован как активный компонент микроархитектуры, содержащие локальные регистры:

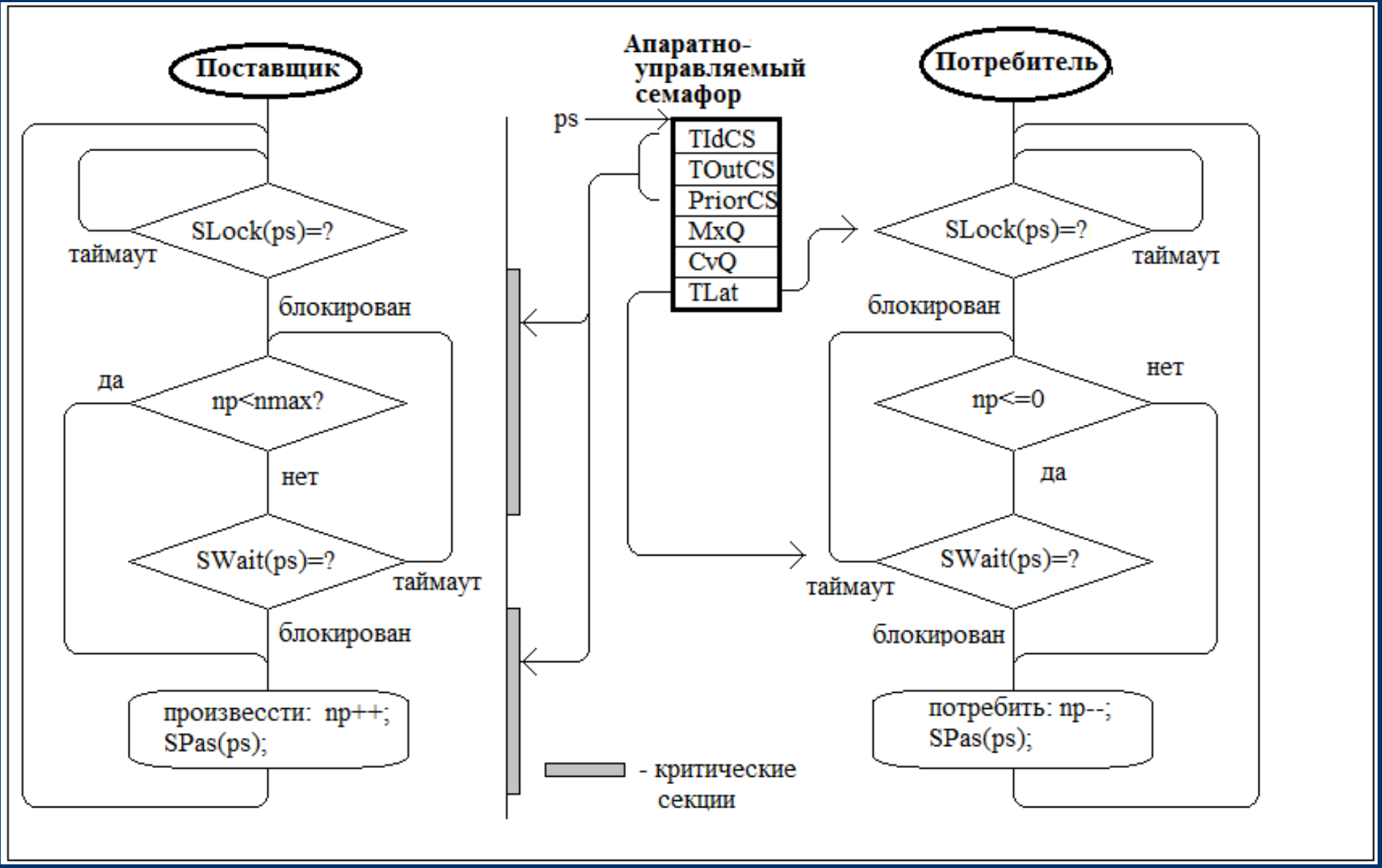
- TIdCS - идентификатор нити, находящейся в критической секции (КС);
- TOutCS - таймаут времени пребывания нити в КС;
- PriorCS - временный (повышенный) приоритет при работе нити в КС, обеспечивающий простое и эффективне решение проблемы инверсии приоритетов;
- MxQ- очередь запросов нитей к мутексу данного АУС;
- CvQ - очередь запросов нитей к условной переменной данного АУС;
- TLat - таймаут ожидания входа в КС по описываемым ниже командам SLOck и SWait.

Команды синхронизации нитей в ВтМаА

Виртуалтрединговые средства синхронизации представлены следующими вынесенными на уровень ISA аппаратными командами ВтМа, реализующими осноные функции синхронизации стандарта POSIX:

- -SGet с операндами tw –(тайм-аут ожидания входа нити в критическую секцию – (КС)) и prs (приоритет при работе в критической секции) объединяет функций pthread_mutex_init и pthread_cond_init, выделяет программе аппаратный семафор (из пула свободных) и возвращает его указатель ps;
- - SLock(ps, ts) заменяет функцию pthread_mutex_lock и обеспечивает либо вход нити в КС с тайм-аутом пребывания в ней ts, приоритетом prs (если эта секция свободна), либо ожидание входа в нее в течении ts до выдачи команды SPas(ps);
- - SWait(ps, ts) (аналог pthread_cond_timedwait) реализует неактивное ожидание входа в КС до выдачи команды SPas(ps), позволяя устранить ресурсоемкий опрос разделяемых переменных в критической секции;
- - SPas(ps) объединяя действия функций pthread_mutex_unlock и pthread_cond_broadcast, выводит нить из критической секций, охраняемой семафором ps и освобождает ее для других нитей.

Алгоритм поставщик-потребитель на основе АУС



Спасибо за внимание

Ефимов А.И., к.т.н., eai_andr_kb@mail.ru

фрилансер, г.Гомель, Беларусь

Автор выражает глубокую благодарность Imagination Technologies и лично Юрию Панчулу за предоставленную макетную плату, поддержку и консультации.

Список цитируемых источников

1. B.J. Smith, Architecture and applications of the HEP multiprocessor computer system, Proc. International Society for Optical Engineering, pp. 241-248, 1982. http://www.cs.auckland.ac.nz/compsci703s1c/resources/BSmith.pdf

2. HORIZON A PROCESSOR ARCHITECTURE FOR HORIZON, Mark R. Thistle, Burton J. Smith, CH2617 9/88/0000/0035 01.00 1988 IEEE http://www.cs.berkeley.edu/~culler/cs252-s02/papers/a-processor-design-for-horizon.pdf

3. The Tera Computer System. Robert Alverson, David Callahan, Daniel Cummings, Brian Koblenz, Allan Porterfield, and Burton Smith ACM International Conference on Supercomputing, pp. 1 6, June 1990 http://www.cray.com/downloads/mta2 papers/arch.pdf

4. Cray MTA-2 System - HPC Technology Initiatives http://www.cray.com/products/programs/mta 2/index.html CrayReseach. www.cray.com

5. Burton Smith, Reinventing Computing. Microsoft Research Faculty Summit 2007 http://www.cct.lsu.edu/~estrabd/LACSI2006/Smith.pdf

6. Burton Smith, The Quest for General Purpose Parallel Computing. 1994 Developing a computer science agenda for high-performance computing ACM New York, NY, USA ©1994

Список цитируемых источников (продолжение)

7. T. Sterling. Critical Factors and Directions for Petaflops-scale Supercomputers. California Institute of Technology, NASA Jet Propulsion Laboratory, Presentation to IFIP WG10.3 e-Seminar Series. 4 January 2005.
8. http://en.wikipedia.org/wiki/CDC 6600
9. T. Kilburn, D. Edwards, M. Lanigan, and F. Sumner.
One-level storage system. IRE Trans. Elect. Computers, 37:223–235, 1962.
10. Ефимов А.И. Способ организации многопроцессорной ЭВМ. Патент на изобретение Республики Беларусь N 9989 (приоритет от 2004.09.16)
11. How to Fake 1000 Registers, Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture
http://aggregate.ee.engr.uky.edu/LAR/micro05.pdf