Зборник радова Факултета техничких наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/28BE38Radonjic

ИМПЛЕМЕНТАЦИЈА И ТЕСТИРАЊЕ ХИПЕРВИЗОРА ЗА ПАРАЛЕЛНО ИЗВРШАВАЊЕ ПАРТИЦИЈА НА *ZEDBOARD* ПЛАТФОРМИ

HYPERVISOR IMPLEMENTATION AND TESTING OF PARALLEL PARTITION EXECUTION ON ZEDBOARD PLATFORM

Ксенија Радоњић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *у овом раду описана је имплементација и рад хипервизора на развојној платформи ZedBoard. Ради се о слоју између хардвера и софтвера који оптимизује искоришћеност обје стране, омогућавајући извршавање вишеструких апликација на једном хардверу. За распоређивање паралелних партиција (задатака) на више процесорских језгара коришћен је EDF планер. Детаљно је објашњена структура и кораци за имплементацију система на ZedBoard плочи. Описана су спроведена тестирања комуникације, начина рада и распоређивања партиција на процесорским језгрима.*

Кључне речи: *хипервизор, распоређивање, партиција, паралелно извршавање, ZedBoard*

Abstract – *This article describes implementation and functionality of a hypervisor on the ZedBoard development board. Serving as a layer between software and hardware, the hypervisor optimizes resource utilization on both sides, facilitating the concurent execution of multiple applications on the physical hardware. The Earliest Deadline First (EDF) algorithm is employed for scheduling partitions (tasks) across multiple processor cores. The article details the system structure and provides steps for implementing the hypervisor on the target board. Partition communication, interactions and scheduling modes are tested and thoroughly explained.*

Keywords: *hypervisor, scheduling, partition, parallel execution, ZedBoard*

1. УВОД

Ембедед системи су рачунарски системи намијењени за извршавање специфичних задатака и функција, обично уграђени у уређаје који су веома ресурсно ограничени. Тема овог рада јесте развијање хипервизора [1] за овакво окружење, како би омогућио оптимизацију перформанси и смањење потрошње енергије уређаја. Хипервизор представља слој између софтверског и хардверског дијела система, чија је улога обезбјеђивање максималне искоришћености обје стране.

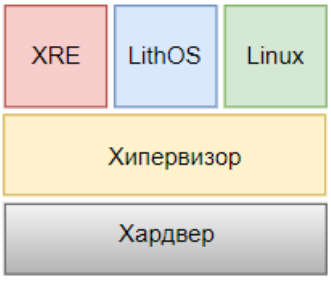
НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Предраг Теодоровић, ванр. проф.

Bare-metal хипервизор имплементира се директно на хардвер циљане платформе, без оперативног система. Предност ове методе је одвајање оперативног система од основног хардвера, тако да софтвер престаје бити директно зависан или ограничен на хардверске уређаје. Овакав приступ омогућава оперативним системима и повезаним апликацијама да функционишу на различитим типовима хардвера. Осим тога, хипервизор омогућава вишеструким оперативним система и виртуелним машинама истовремено дијељење хардверске платформе, на овај начин повећавајући искоришћеност ресурса и оптимизујући перформансе цјелокупног система.

Први корак при укључивању на пројекат представљао је упознавање и разумијевање развијеног концепта окружења. Хардвер коришћен за имплементацију хипервизора јесте развојна платформа *ZedBoard Zynq-7000 SoC* [2]. У ранијој фази пројекта развијен је алгоритам "*EDF*" (eng. *Earliest Deadline Fir*st) планера [3], чија је улога ефикасно распоређивање партиција (односно задатака) на више процесорских језгара. Након детаљне анализе рада алгоритма, развијен је фајл генератор за аутоматизацију процеса формирања структуре фајлова неопходне за конфигурацију хипервизора. Наредни кораци подразумијевали су имплементацију система на плочу, а потом и тестирање распореда, комуникације и начина рада партиција.

2. ХИПЕРВИЗОР

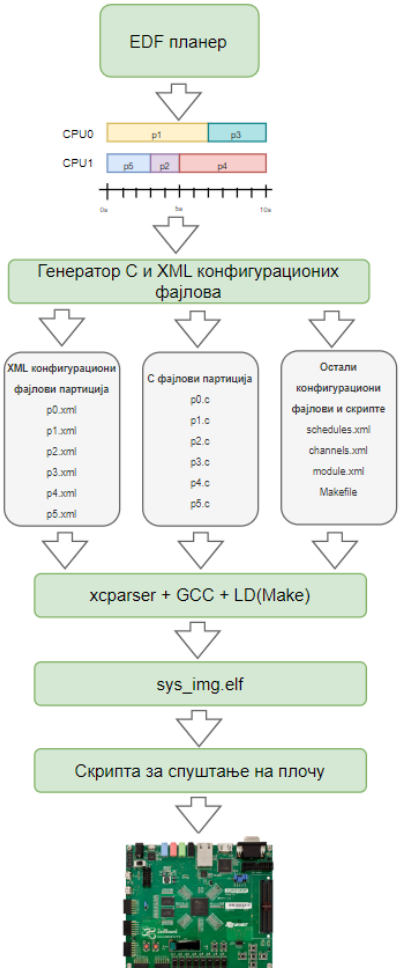
Поглавље описује теоријске основе хипервизора који омогућава покретање и управљање више оперативних система на једном хардверу. Слика 1 илуструје архитектуру таквог модела, са три истовремено покренута оперативна система.



Слика 1. *Архитектура система са хипервизором*

У овом раду је представљен рад са “*XRE*” *(*eng. *XtratuM Runtime Environment)*, односно системом који обезбјеђује задатке са минималним “*C*” окружењем за извршавање.

Приказ корака за имплементацију система приказан је на слици 2. У наредним поглављима детаљно ће бити објашњен сваки дио дијаграма.



Слика 2. Кораци за имплементацију система на хардвер

2.1 “*EDF*” планер

Циљ имплементације “*EDF*” алгоритма јесте да на најефикаснији начин направи план, односно распоред задатака који обављају различите функционалности. Начин рада овог алгоритма јесте такав да се процеси постављају у приоритетни ред, тако да када језгро процесора буде спремно да изврши следећи задатак бира задатак са најранијим роком за извршење.
Осим тога, алгоритам је “*non-preemptive*”, што значи да задатак који започне да се извршава на неком од процесора мора и да се настави на истом процесору, све до краја његовог извршења. Информације које је потребно обезбиједити планеру као улаз су:

- Број језгара процесора
- Крајњи рок за чије вријеме требају бити извршени сви задаци на процесорима
- Идентификатор сваког задатка и вријеме потребно да буде извршен
- Зависности међу задацима, као и крајњи појединачни рокови сваког од њих

2.2 Партиција

Партиција представља основну јединицу сегрегације како у временском, тако и у просторном смислу. Могу се представити као пасивни контејнери који обезбјеђују виртуелно извршавање одређеног окружења, у овом случају “*XRE*” система. За почетак партицијама нису додијељени комплексни задаци, већ извршавање основних радњи које би указале да је исправно конфигурисан систем, односно распоред партиција на више језгара, комуникација посредством специјализованих портова, итд. Заправо се ријеч задатак из претходног поглавља може изједначити са ријечју партиција, и може се рећи да “*EDF*” планер распоређује партиције на процесорска језгра. У другом кораку на слици 2 приказан је распоред пет партиција на два процесорска језгра, са дужинама трајања партиција представљеним на временској оси.

2.3 Развијање фајл генератора

Излаз програма “*EDF*” планера је испис распореда партиција на процесорима у терминалу. Наредни корак у пројекту представљао је развијање фајл генератора за аутоматизацију формирања структуре датотека која конфигурише хипервизор, с обзиром на то да је мануелно уписивање добијених података у фајлове неефикасно решење.

Након темељног упознавања са радом “*EDF*” планера, код је модификован тако да екстрахује све неопходне податке и уписује их у текстуални фајл “*data_partitions.txt*”. Фајл генератор развијен у “*C++*” програмском језику, ишчитава и парсира податке из “*data_partitions.txt*” фајла. Користи библиотеку “*tinyxml2*” за упис форматираних података у *XML* конфигурационе фајлове. Осим тога, формира цјелокупну структуру података у директоријуму пројекта. У њему се налазе:

- **фолдер *xml***:
  - module.xml
  - channel.xml
  - hypervisor.xml
  - multipartition_health_monitor.xml
  - schedules.xml
  - <partition_name>.xml
- **<partition_name>.c**
- **Makefile**

Наведени фајлови детаљније су објашњени у наредним поглављима.

2.4 *XML* конфигурациони фајлови

Систем се статички конфигурише преко *XML* конфигурационих фајлова, који се називају “*XCF*” (eng. *XML Configuration Files*). Улога “*XCF*” јесте дефинисање системских ресурса и начина на који се додјељују свакој партицији и хипервизору током извршавања система. Неке од информација које се налазе у овим фајловима су подаци о плочи, броју

процесора, фреквенцији рада и начину кеширања. У њима се такође конфигурише конзола за интеракцију са корисником, комуникациони канали који повезују дефинисане партиције, стриктно дефинише распоред партиција на процесорима, итд.

2.5 Изворни фајлови партиција

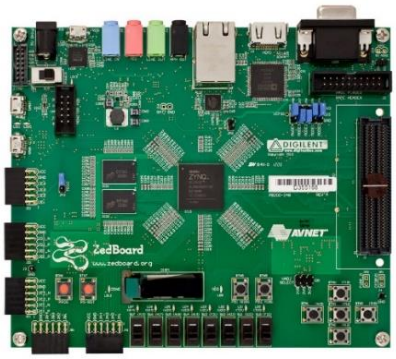
У *"<partition_name>.c*" фајловима налазе се изворни кодови партиција, гдје је имплементирана њихова функционалност. На почетку фајла укључују се библиотеке и заглавља за руковање партицијама, развијени у претходним фазама пројекта. Позивају се предефинисане функције за провјеру исправног учитавања партиције и комуникационих канала, а потом имплементирају жељене операције. У фази пројекта коју описује овај рад, циљ имплементације партиције јесте слање једноставних порука свим партицијама које зависе од ње.

2.6 Генерисање извршне датотеке

"*Makefile*" је текстуални фајл који се користи за аутоматизацију процеса компајлирања и изградње софтвера. Након формирања структуре фајлова, развијен је "*Makefile*" чија је улога парсирање "*XCF*", као и компајлирање и линковање изворних фајлова. Коришћен је рачунар са "*Ubuntu*" оперативним системом, а за компајлирање "*GCC*" (eng. *GNU Compiler Collection*) компајлер. Позивом команде *make* покреће се "*Makefile*" и компајлирају сви фајлови, тако да се као излаз добија извршна датотека "*sys_img.elf*". Ova бинарна датотека садржи машински код и потребне информације о партицијама и хипервизору за учитавање података у меморију циљане хардверске платформе.

2.7 Имплементација

У овом раду коришћена је *ZedBoard* развојна плоча, која припада серији *Zynq7000 SoC* (eng. *System on Chip*) платформи и приказана је на слици 3.



Слика 3. ZedBoard развојна плоча

ZedBoard је развојна плоча која комбинује *ARM* процесор са *FPGA (*eng. *Field-Programmable Gate Array)* технологијом. У овом пројекту користи се само "*PS*" (eng. *Processing System*) дио интегрисаног кола, што подразумијева два 32-bit *ARM Cortex A-9* процесорска језгра, без коришћења *FPGA*.

Рачунар, на ком се налази извршна датотека са свим подацима, повезује се са плочом коришћењем два *USB* (eng. *Universal Serial Bus*) кабла. Један служи за серијско повезивање са конзолом и приказ порука партиција. Други USB кабл је "*JTAG*" програматор који спушта хипервизор на циљану плочу. "*JTAG" (*eng. *Joint Test Action Group)* је интерфејс који се користи за комуникацију између рачунара и развојних плоча како би се омогућило програмирање и дебаговање интегрисаних кола на истим.

Крајња скрипта *"jtag_deployment.sh*" увезује све неопходне датотеке и имплементира систем на *ZedBoard* плочу. Уколико се на исправан начин покрене, сви дјелови пројекта ће заузети меморијске ресурсе на начин који је дефинисан. Партиције ће се распоредити на процесорима и започети своје извршавање. У терминалу је кориснику омогућено праћење повратних информација, односно порука које партиције исписују. На овај начин вршено је дебаговање и истраживање различитих функционалности система. Овај дио експерименталне фазе описан је у наредном поглављу.

3. ЕКСПЕРИМЕНТАЛНА ФАЗА

3.1 Испис порука на конзолу

Прва фаза тестирања обухватила је испитивање правилног распоређивања партиција на хардверској плочи. За дебаговање је у изворним фајловима партиција коришћена функција за испис порука на конзолу. Ово не представља ефикасно решење јер захтјева додатно процесорско вријеме за обраду исписа, али у том тренутку ефикаснији алати нису били развијени. Проблем је постао изразитији због паралелне обраде на два процесорска језгра, што је изазвало преклапање и преплитање порука различитих партиција.

У циљу решавања проблема преплитања порука, примијењени су мутекси. Мутекс (eng. "*mutual exclusion*", односно „међусобно искључивање") је механизам синхронизације који се користи у паралелном програмирању како би се осигурало да само једна нит (партиција) може да приступи одређеном дијелу ресурса у датом тренутку. Међутим, сама операција исписа порука на конзолу је ресурсно захтјевна из следећих разлога:

- поруке које се шаљу на конзолу чувају се у баферу прије прослијеђивања на излазни ток
- процесор мора прекинути свој рад да би исписао поруку на конзолу
- током процеса слања података на конзолу, процесор мора обрадити те поруке и извршити операције за ову радњу

Наведене ставке захтјевају значајно вријеме и ресурсе процесора.У контексту времена извршавања, ако се задатак понављано заустављa ради исписа порука на конзолу, то може довести до прекорачења његовог рока извршавања, посебно у окружењима гдје је вријеме извршавања критично.

Из наведених разлога је додатна пажња посвећена минимизацији броја, као и дужине порука које се исписују. Након експериментисања са позицијом исписа у коду, дужине поруке, броја података, мутексима, итд., партиције су успјешно имплементиране и

правилно распоређене на, за њих предвиђеном, процесору.

3.2 Коришћење комуникационих канала

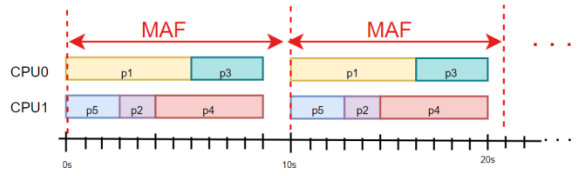
За комуникацију међу партицијама користе се комуникациони портови. Режим порта је "*queuing*", који функционише на начин да се поруке из партиција чувају у *"FIFO"* (eng. *"First In First Out"*) баферу. Све нове поруке чекају у реду, а затим се обрађују редоследом којим су и уписане. Сви коришћени канали морају бити дефинисани у изворним фајловима партиција и *XML* конфигурационим фајловима.

На почетку је имплементирана најједноставнија функционалност гдје свака партиција дефинише портове и шаље поруку оним партицијама које зависе од ње. У овом кораку је комуникација, односно коришћење механизама канала и портова међу партицијама, успјешно имплементирана. Омогућен је пренос информација међу задацима, што је од велике важности за касније фазе пројекта чији је циљ обезбјеђивање паралелног рада много комплекснијих апликација.

Због тога је следећи корак био проширивање величине, као и броја порука које се шаљу. Успјешно је тестиран примјер гдје партиција шаље више порука партицији зависној од ње. Након што их прими, зависна партиција исписује на конзолу укупан број примљених карактера чија вриједност потврђује да су све прочитане, а потом и садржај последње примљене порука чиме се осигурава исправност пријема. Оваква имплементација од великог је значаја због информација које партиције у многим случајевима морају међусобно размијенити.

3.3 Тестирања режима рада система

У складу са циљем пројекта који подразумева имплементацију реалних оперативних система, извршене су адаптације кода ради обезбјеђивања непрекидног рада система. Важан атрибут планера јесте његов главни оквир, "*MAF*" (eng. *Major Frame*), дефинисан као временски интервал у ком се изврше све партиције. У овом кораку примјењен је концепт периодичног понављања распореда, приказан на слици 4.



Слика 4. Примјер рада са понављањем "MAF"-а

Након ове фазе, спровођени су различити експерименти за провјеру могућих режима рада партиција. Успјешно је имплементирана подјела једне партиције у два одвојена временска слота извршена у различитим временским интервалима, на истом процесору. Потом је успјешно спроведен и тест извршавања партиције на два различита процесора у истом временском интервалу. Ово је тестирано као провјера дате могућности и није даље испитивано, већ је остављено за касније фазе развоја пројекта.

У наредном тесту је имплементирано успјешно повезивање система са дигиталним пиновима *ZedBoard* плоче. Изворни код партиције је прилагођен тако да иницијализује пин ровезан са "*PB1*" дугметом уграђеним на хардверску платформу. Након успостављања комуникације ишчитава његово стање, а потом и исписује поруку са информацијом да ли је дугме притиснуто или не.

4. ЗАКЉУЧАК

У овом раду представљена је иницијална фаза пројекта у ком се развија хипервизор за оптимизацију перформанси ембедед система који су ресурсно ограничени. Објашњене су основе начина рада и структура система, кораци за његову имплементацију, а потом и различита тестирања којима је потврђен исправан рад система.

Будући да је описан рад почетна фаза цјелокупног пројекта многа унапређења су могућа, а наводе се само нека од њих:

- Коришћење *Zynq UltraScale MPSoC* [4] плоче са шест процесорских језгара.
- Развој специјализованих алата за дебаговање функционалности партиција.
- Имплементација оперативних система веће комплексности.

5. ЛИТЕРАТУРА

[1] https://www.vmware.com/topics/glossary/content/hypervisor.html, април 2024.

[2] https://digilent.com/reference/programmable-logic/zedboard/reference-manual, април 2024.

[3] https://www.baeldung.com/cs/scheduling-earliest-deadline-first , април 2024.

[4] https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html , април 2024.

Кратка биографија:

Ксенија Радоњић рођена је у Никшићу 1999. године. Дипломски рад на Факултету техничких наука из области Електротехника и рачунарство – Ембедед системи и алгоритми одбранила је 2022. године.

контакт:
radonjicksenija20@gmail.com