Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/02IH07Boberic

JEDNO OD REŠENJA ZA PARTICIONISANJE CENTRALNE PROCESORSKE JEDINICE SA VIŠE JEZGARA

ONE SOLUTION FOR PARTITIONING MULTICORE CENTRAL PROCESSING UNIT

Milan Boberić, *Fakultet tehničkih nauka, Novi Sad*

Oblast –MEHATRONIKA

Kratak sadržaj – *U ovom radu predstavljena je implementacija i optimizacija sistema za particionisanje centralne procesorske jedinice sa više jezgara. Sistem se sastoji iz UltraZed-EG ploče na kojoj je implementiran Xen Hypervisor kao sistem za particionisanje. Xen Hypervisor je virtuelni sloj koji koristi postojeći operativni sistem (host) koji mu omogućava pristup memoriji, prekidima i drugim resursima i omogućava kreiranje virtuelnih jezgara koja se po instrukcijama korisnika mogu raspoređivati na fizička jezgra. Dat je akcenat na real-time particionisanje i optimizaciju Xen Hypervisora u cilju poboljšanja performansi.*

Ključne reči: *Particionisanje centralne procesorske jedinice, Xen Hypervisor, virtuelizacija, UltraZed-EG*

Abstract – *This master thesis presents the implementation and optimization of a system for partitioning a multicore central processing unit. The system contains an UltraZed-EG board on which Xen Hypervisor is implemented as the partitioning system. Xen Hypervisor is a virtual layer which uses the existing operative system (host) to access memory, interrupts, and other resources and allows creation of virtual cores which can be scheduled across physical cores. Accent is given on real-time partitioning and Xen Hypervisor's optimization in order to improve it's performance.*

Keywords: *Central processing unit partitioning, Xen Hypervisor, virtualization, UltraZed-EG*

1. UVOD

Particionisanje centralne procesorske jedinice sa više jezgara znači podela grupe jezgara ili pojedinačnih jezgra za specijalnu upotrebu jedne ili više aplikacija. Svaka sekcija se ponaša kao zaseban sistem sa promenljivim stepenom fleksibilnosti raspodele. *Real-time* particionisanje sistema opisuje izloženost hardvera i softvera real-time ograničenjima, na primer od nekog događaja do reakcije sistema na taj događaj, *real-time* programi moraju garantovati reakciju sistema u određenom vremenskom intervalu.

Jedan od sistema za particionisanje jeste Xen Hypervisor. U ovom radu je opisana implementacija Xen Hypervisor-a sistema za particionisanje kao i poređenje performansi sistema sa i bez Xen Hypervisor-a na ploči UltraZed-EG kao i na ploči ZedBoard.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Vladimir Rajs, docent.

Pored drugih hypervisor-a kao što su: VMware, ESXi, Hyper-V, KVM, itd. Xen Hypervisor ima cenu, performance i sigurnost da parira najboljim na svetu. Ima deset godina iskustva u radu sa najvećim *cloud*-ovima na svetu. Siguran je, stabilan i proveren izbor za virtuelizaciju koji koriste giganti u industriji kao što su: *Amazon, Rackspace, Verizon,* itd.

2. HARDVER

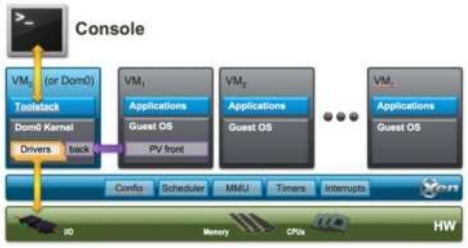
U okviru projekta koristi se UltraZed-EG ploča sa nosačom kartice. UltraZed-EG SOM (*system on module*) je visoko fleksibilan, robustan system-on-module baziran na Xilix Zynq UltraScale+MPSoC. UltraZed-EG omogućava lak pristup ka 180 I/O pinova, 26 *PS MIO* pinova i četiri *PS GTR* transivera velike brzine sa četiri GTR referentna klok ulaza koja imaju tri I/O konektora na poleđini modula. Takođe sadži Quad-core Cortex A53, Dual-core Cortex-R5 *real-time* procesor i ARM MALI 400MP GPU gde su 4 jezgra Cortex A53 procesora na raspolaganju za virtuelizaciju. Na slici 1 je prikazan izgled ploče.

Slika 1 - *Izgled UltraZed-EG ploče [1]*

3. XEN HYPERVISOR U LINUX OPERATIVNOM SISTEMU

Xen je virtuelni sloj kao što je prikazano na slici 2. Na osnovu virtuelnih mašina (*VM*) i podešavanja Xen Hypervisor-a, *VM-e* se raspoređuju po fizičkim jezgrima (*pCPU*) i imaju pristup određenim delovima fizičkog hardvera u zavisnosti od passthrough-a.

U ovom slučaju *host* tj. dom0 je PetaLinux, Xen guests su totalno izolovani od hardvera i nemaju pristup ni hardveru niti I/O funkcijama i nazivaju se domU. Xen Hypervisor neće funkcionisati bez dom0 koji je prva virtuelna mašina koju sistem pokrene.



Slika 2 - *Primer arhitekture Xen Hypervisor-a [2]*

Tipovi particionisanja Xen Hypervisor-a su:

- soft
- hard
- particionisanje sa null *scheduler-om.*

U ovom radu je korišteno particionisanje sa null *scheduler-om,* ovaj tip particionisanja daje najbolje performanse i najmanji *jitter*. U ovom načinu particionisanja svako virtuelno jezgro (*vCPU*) je striktno mapirano jedan na jedan sa odgovarajućim *pCPU*. Ostali tipovi particionisanja se zasnivaju na raspoređivanju vCPU-a na pCPU-e na osnovu prioriteta što zahteva korišćenje određenog sistema za raspoređivanje (*scheduler-a*).

4. *BUILD* XEN HYPERVISOR-A SA PETALINUX-OM

PetaLinux alat poseduje sve neophodne stvari za prilagođavanje, build-ovanje i razvijanje embedded linux rešenja na Xilix sistemima. Prilagođen je da ubrza produktivnost dizajna, ovo rešenje i Xilinx alat za dizajniranje hardvera u mnogome olakšavaju razvoj Linux sistema za Zynq UltraScale+MPSoC, Zynq-7000 SoC i MicroBlaze. PetaLinux 2018.2 je korišten u ovom radu.

Zahtevi računara za instalaciju PetaLinux-a: 8GB RAM (preporučeni minimum za Xilinx alate), 2GHz CPU clock ili ekvivalentni (minimum 8 jezgara), 100GB slobodnog prostora na hard disku, operativni sistem: (Red Hat Enterprise Workstation/Server 7.2, 7.3, 7.4 (64-bit), CentOS 7.2, 7.3, 7.4 (64-bit) ili Ubuntu Linux 16.04.3 (64-bit) koji je izabran u ovom radu), root pristup nekim operacijama. PetaLinux alati trebaju biti instalirani kao non-root user, PetaLinux zahteva razne standardne razvojne alate i biblioteke koje moraju biti instalirane na Linux host-u. U tabeli 1 prikazan je tok razvoja Xen Hypervisor-a korišćenjem PetaLinux alata [6].

Tabela 1 - *Tok dizajna [6]*

Koraci prilikom dizajniranja	**Alat / Proces rada**
Pravljenje hardware platforme	Vivado
Pravljenje PetaLinux projekta	petalinux-create -t project
Inicijalizovanje PetaLinux projekta	petalinux-config --get-hw-description
Konfigurisanje system-level opcija	petalinux-config
Pravljenje korisničkih komponenti	petalinux-create -t COMPONENT
Konfigurisanje Linux kernel	petalinux-config -c kernel
Konfigurisanje Root FileSystem-a	petalinux-config -c rootfs
Build-ovanje sistema	petalinux-build
Sistem deployment	petalinux-package
Testiranje sistema	petalinux-boot

5. PRIPREMA ZA BOOT-OVANJE PETALINUX IMAGE-A NA ULTRAZED PLOČI SA SD KARTICOM

Preduslovi:

- instalirano PetaLinux radno okruženje
- instaliran board support package (*BSP*)
- serijska komunikacija je podešena sa *baudrate-om* 115200
- UltraZed je podešen za boot-ovanje sa SD kartice, podesiti prekidače *(boot mode switches SW2*) na OFF-ON-OFF-ON

6. KORACI ZA BOOT-OVANJE PETALINUX-A NA HARDVER SA SD KARTICOM

1. Ubaciti SD karticu u računar
2. Na karticu kopirati sledeće fajlove, ovi fajlovi se nalaze u direktorijumu <path_to_project>/images/linux:
    - BOOT.bin
    - Image
    - system.dtb
    - xen.ub
    - rootfs.cpio.gz.u-boot
3. Povezati serjiski port ploče sa računarom
4. Otvoriti terminal na računaru (Putty, Kermit, minicom, gtkterm, itd.) i podesiti baud rate na 115200
5. Isključiti ploču
6. Postaviti boot mode preko prekidača na ploči kao što je opisano u prethodnoj sekciji
7. Ubaciti SD karticu u ploču
8. Uključiti ploču
9. Obratiti pažnju na terminal, pojaviće se boot poruke

Kada dom0 završi boot-ovanje može se pristupiti kreiranju jednostavnog domU *guest-a.* Potrebno je kopirati *guest* Image u dom0 *filesystem*, može se koristiti *prebuilt* PetaLinux Image kao domU *guest*. Svaki domU ukoliko se koristi *bare-metal* aplikacija, koja se kreira u Xilinx SDK random okruženju, mora imati .bin i .cfg fajlove. Fajl sa ekstenzijom .bin sadrži samu *bare-metal* aplikaciju i generiše se u Xilinx SDK random okruženju a .cfg fajl sadrži potreba podešava za Xen Hypervisor [4].

Primer .cfg fajla:

```
#Guest name
  name = "bm1"
# Kernel image to boot
  kernel = "ultraled.bin"
# Kernel command line options - Allocate 8MB
  memory = 8
# Number of VCPUS
  vcpus = 1
# Pin to CPU 2
  cpus = [2]
  irqs = [ 48 ]
  iomem = [ "0xff0a0,1" ]
```

Da bi domU imao pristup na primer *UART-u* i *GPIO* ubačen je *passthrough* dodatkom u *device-tree* pre build-a [3]:

```
&uart1{
        xen,passthrough=<0x1>;
};

&gpio{
        xen,passthrough=<0x1>;
};
```

Nakon pozicioniranja u direktorijum sa .bin i .cfg fajlovima za kreiranje domU, koji će biti pokrenut samo na trećem jezgru, korištena je sledeća komanda:

$ xl create -c example-simple.cfg
gde je *example-simple.cfg* konfiguracioni fajl. Konzola guest-a se može napustiti sa **Ctrl+].** U dom0 konzoli se mogu izlistati trenutni domU komandom:
$ xl list

Vraćanje u konzolu guest-a može se uraditi komandom:
$ xl console <ime-guest-a>

A domU može biti isključen komandom:

$ xl destroy <ime domU-a >

7. "OVERHEAD" XEN HYPERVISOR-A

Ovo poglavlje se bavi optimizacijom performansi Xen Hypervisor-a kao i upoređivanjem jitter-a bare-metal aplikacije "pod" Xen Hypervisor-om gde je CPU na kom je host tj. dom0 PetaLinux stresiran na 100% opretećenja i bare-metal aplikacije koja se sama spušta na ploču bez Xen Hypervisor-a.

Xen Hypervisor podržava više različitih *scheduler-a* tj. sistema za raporeĎivanje *vCPU-ova*. Posao hypervisor scheduler-a jeste da donosi odluke koji *vCPU* od brojnih *vCPU-ova* različitih *VM* će biti pokrenut na *pCPU-ima* i u koje vreme. Takođe je podržana opcija postojanja više različitih aktivnih *scheduler-a* u odvojenim grupama *pCPU-a* koji se nazivaju *cpu-pool-ovi.*

CPU-pool-ovi omogućavaju odvajanje fizičkih *CPU-a* u odvojene grupe pod nazivom *cpu-pool*. Svaki pool sadrži svoj potpuno odvojeni *scheduler*. *Domain-i* su dodeljeni *pool-ovima* pri kreiranju i mogu biti premeštani iz jednog *pool-a* u drugi.

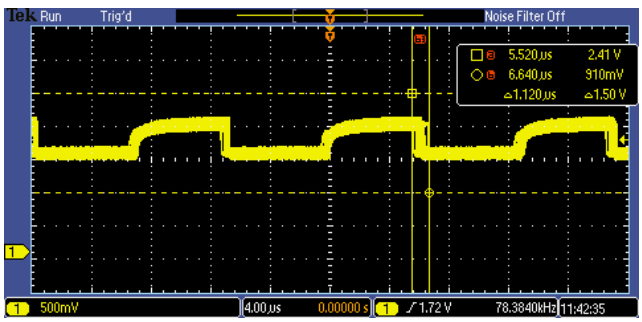
Tipovi Xen Hypervisor scheduler-a [4]:

- *Credit Scheduler* – koji je opšte namene i podešen je po default-u
- *Credit2 Scheduler* – je naslednik *Credit Scheduler-a,* skalabilniji je i bolji je sa opterećenjima osetljivim na "*latency*" ali je pored toga baziran na kao *scheduler* opšte namene
- *RTDS Scheduler* – je *real-time scheduler* koji služi za podšku *real-time* opterećenja u *cloud-u* kao i za embedded i mobilnu virtuelizaciju
- *ARINC653 Scheduler* – je embedded (automobilska i avio industrija) *real-time scheduler*
- *Null Scheduler* – koji je korišten u ovom radu jer poseduje najbolje performanse jer nema *scheduler-a* tako da nema "donosenja odluka" i pomeranja *vCPU-ova* po *pCPU-ima* što dodatno smanjuje vreme, svaki *vCPU* je dodeljen određenom *pCPU-u*.

U ovom radu *bare-metal* aplikacija je programirana da uključuje i isključuje tj. da trepće PS LED na UltraZed carrier card-u na osnovu čega se meri *jitter*.
Da bi se postigle najbolje performanse, najmanji overhead i najmanji interrupt latency ubačene su sledeće komande u xen-overlay fajl kao xen-bootargs [5]:

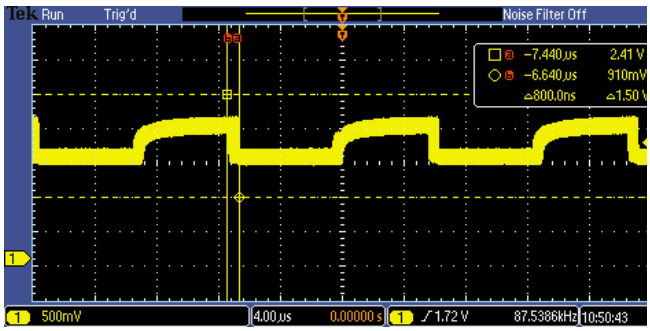
sched=null
vwfi=native

sched=null isključuje default Credit scheduler čime se dobija najniži overhead.
vwfi=native, gde *vwfi* znači *virtual wait for interrupt* tj. čekanje na virtuelni prekid, kada je postavljeno na „*native*" smanjuje *interrupt latency* približno 60% jer Hypervisor tada ne zadržava wfi i wfe komande koje su instrukcije ARM procesora za „*sleep*", podrazumevano tj. default podešavanje za vwfi je „*trap*". DEBUG opcija je takođe isključena, kada je uključena daje mnogo korisnih poruka i provera po ceni povećanog *latency-a*.
Kao što je prikazano na slici 4, jitter meren osciloskopom je 1,120 µs na UltraZed-EG ploči sa Xen Hypervisor-om.



Slika 3 - *Jitter sa Xen Hypervisor-om*

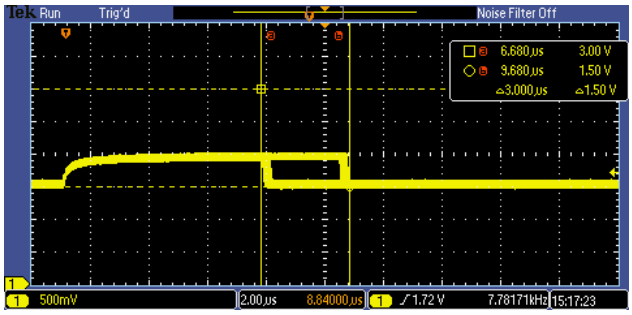
Na slici 4 je prikazan *jitter* meren osciloskopom na ploči UltraZed-EG gde je *bare-metal* aplikacija za treptanje PS LED-a spuštena na ploču preko JTAG-a, naravno bez Xen Hypervisora, iz Xilnx SDK radnog okruženja. *Jitter* iznosi 800 ns.



Slika 4 - *Jitter bez Xen Hypervisor-a*

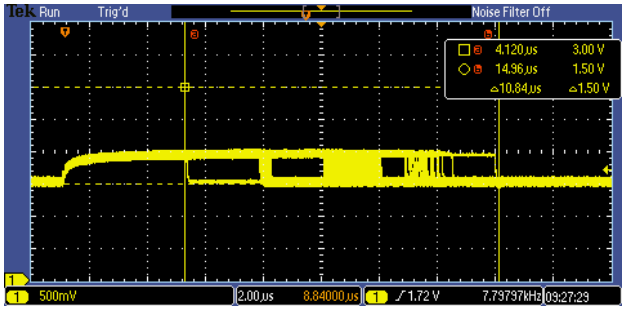
Jitter ostaje isti prilikom stresiranja CPU-a na kom se nalazi dom0.

Izvršeno je testiranje sa aplikacijom koja pored uključivanja i isključivanja LED-a postavlja neke vrednosti i generiše tajmer interrupt svakih 10 µs. Izmeren je jitter od 3 µs što ukazuje da se Xen Hypervisor ima prevelik overhead da bi se koristio u sistemu za *real-time* simulacije, *real-time signal processing*, itd. Smanjenjem frekvencije *interrupt-a* na 15 µs jitter se smanjuje na vrednost od 1,120 µs što ukazuje na problem pri generisanju *interrupt-a* visoke učestanosti. Na slici 5 prikazan je *jitter* pri frekvenciji *interrupt-a* od 10 µs.



Slika 5 - *Jitter sa aplikacijom koja genneriše interrupt-e*

Takođe je izvršeno merenje korišćenjem iste aplikacije i *default credit scheduler-a* gde je dobijen jitter od 10 µs pri frekvenciji *interrupt-a* od 10 µs što je i prikazano na slici 6.



Slika 6 - *Jitter sa credit scheduler-om i aplikacijom koja generiše interrupt-e*

8. ZAKLJUČAK

Dat je kratak pregled real-time particionisanja kao i njegova podela. Primećen je porast u potražnji *real-time* sistema sa sistemom za particionisanje. Objašnjen je *overhead* i *jitter* čije će se vrednosti smanjivati primenom određenih dodataka.

Dati su zahtevi računara za build-ovanje. Predstavljen je PetaLinux kao *host* tj. dom0 Xen Hypervisor-a. Objašnjen je proces build-ovanja korišćenjem PetaLinux-a kao i potrebna podešavanja za pokretanje sa SD kartice. Takođe opisuje pokretanje jednostavnog *guest-a* tj. domU-a i CPU pinning koji je veoma bitan za *real-time* particionisanje.

Izvršeno je upoređivanje jitter-a bare-metal aplikacije sa i bez Xen Hypervisor-a. Predstavljene su vrste sistema za raspoređivanje tj. vrste scheduler-a gde je ustanovljeno da null scheduler daje najbolje rezultate.

Izmeren je jitter na aplikaciji koja uključuje i isključuje LED velikom brzinom gde je otkriveno da je jitter Xen Hypervisor-a izmeren osciloskopom 1,120 µs. Predstavljeni su rezultati merenja izvršavanja aplikacije koja pored uključivanja i isključivanja LED-a generiše tajmer interrupt svakih 10 µs. Zaključeno je da su najbolji rezultati dobijeni korišćenjem null scheduler-a i vwfi=native komande.

Nakon iznetih rezultata može se primetiti značajan overhead Xen Hypervisor-a pri konstantnom generisanju interrupt-a te se kao poboljšanje predlaže dodatna optimizacija performansi u slučaju interrupt-a koji se generiše na svakih 10 ili manje od 10 µs.

4. LITERATURA

[1] Donny Saveski. (2018, Sep.) zedboard.org. [Online]. http://zedboard.org/sites/default/files/documentations/5264-UG-AES-ZU3EGES-1-SOM-G-v1-1-V1.pdf

[2] Xilinx. (2018, Sep.) wiki.xilinx.com. [Online]. https://wiki.xen.org/wiki/Xen_Project_Software_Overview

[3] Xilinx. (2018, Sep.) wiki.xilinx.com. [Online]. http://www.wiki.xilinx.com/Building%20Xen%20Hypervisor%20with%20Petalinux%202018.1

[4] xenproject.org. (2018, Sep.) wiki.xenproject.org. [Online]. https://wiki.xenproject.org/wiki/Xen_Project_Schedulers

[5] Stefano Stabellini. (2018, Sep.) xenproject.org. [Online]. blog.xenproject.org/author/stefano-stabellini/

[6] Xilinx. (2018, Sep.) xilinx.com. [Online]. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_2/ug1144-petalinux-tools-reference-guide.pdf#nameddest=PetaLinuxToolsInstallationSteps

Kratka biografija:

Milan Boberić rođen je u Zrenjaninu 1994. god. Master rad na Fakultetu tehničkih nauka iz oblasti Mehatronike odbranio je 2018.god. kontakt: milanboberic94@gmail.com