Dariusz CABAN
POLITECHNIKA ŚLĄSKA, INSTYTUT INFORMATYKI
ul. Akademicka 16, 44-100 Gliwice

Określenie narzutów wprowadzanych przez wielozadaniowe jądro Xilkernel dla procesora PowerPC 405

Dr inż. Dariusz CABAN

Ukończył w 1991 roku studia na kierunku Informatyka na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej w Gliwicach. Stopień doktora nauk technicznych uzyskał w 1999 roku. Od października 2000 roku pracuje na stanowisku adiunkta w Instytucie Informatyki na tym samym wydziale. Jego zainteresowania naukowe związane są m. in. z konstrukcją i oprogramowaniem systemów wbudowanych.

e-mail: Dariusz.Caban@polsl.pl

Streszczenie

System wbudowany, w którym obsługa zdarzeń jest realizowana przez podprogramy wywoływane w pętli głównej programu może nie spełniać stawianych mu wymagań dotyczących czasów reakcji na zdarzenia. Rozwiązaniem problemu może być użycie wielozadaniowego jądra z wywłaszczaniem. Obsługa zdarzeń jest wtedy realizowana przez niezależne zadania o różnym stopniu ważności. Jądro zapewnia obsługę zdarzeń ważniejszych w pierwszej kolejności. Wzrastają jednak: obciążenie procesora i wymagania pamięciowe oprogramowania. W artykule przedstawiono wyniki prac, których celem było określenie tych narzutów wprowadzanych przez jądro Xilkernel.

Słowa kluczowe: system wbudowany, wielozadaniowe jądro z wywłaszczaniem, narzuty czasowe i pamięciowe.

Determining the size of overheads introduced by multitasking Xilkernel for PowerPC 405 processors

Abstract

An embedded system should react to events in its environment before the end of the determined time. The system is notified about the event occurrence usually by the interrupt request signal. The system software can be designed in such a way that only the event flag is set in the interrupt service routine, whereas the event service is carried out in the routine invoked in the main loop. If the time requirements are difficult or impossible to meet by the software of such structure, the multitasking preemptive kernel is used [1]. The system software is divided into independent tasks of various degrees of importance. The kernel ensures execution of more important tasks before less important ones. However, the kernel code occupies some fixed portion of the program and data memory and its execution takes the extra CPU time. This paper presents results of the research work whose purpose was to determine the size of such overheads introduced by Xilkernel, the multitasking kernel developed by Xilinx [2]. The Virtex-II Pro Development System was used to conduct experiments (Fig. 1). In the FPGA chip, contained in this board, a microprocessor system based on the PowerPC 405 core (one of two in the chip) was implemented (Fig. 2). There were measured the interrupt latency, the task latency and the execution time of kernel services. The results are given in Tables 1 and 2. The results should help embedded systems developer to assess whether a system designed can fulfill the timing requirements.

Keywords: embedded system, multitasking preemptive kernel, time and memory overheads.

1. Wprowadzenie

System wbudowany powinien reagować na zdarzenia zachodzące w jego otoczeniu przed upływem określonego czasu. O zajściu zdarzenia system jest z reguły powiadamiany za pomocą sygnału żądania przerwania. Przyjęcie żądania skutkuje wywołaniem odpowiedniego podprogramu. Zalecane jest, aby czas jego wykonania był możliwie jak najkrótszy. Oprogramowanie prostych systemów wbudowanych jest często tak zaprojektowane, że w podprogramach obsługi przerwań są tylko ustawiane znaczniki wystąpienia zdarzeń, natomiast ich obsługa jest realizowana przez podprogramy wywoływane w nieskończonej pętli głównej programu. Jeżeli znaczniki wystąpienia zdarzeń są testowane tylko raz w każdym obiegu pętli, rozpoczęcie obsługi danego zdarzenia w najgorszym przypadku będzie wstrzymane do czasu obsłużenia wszystkich pozostałych zdarzeń. Czas reakcji na zdarzenie można skrócić przez wielokrotne testowanie w pętli głównej znacznika wystąpienia zdarzenia.

Gdy wymagania czasowe są trudne lub niemożliwe do spełnienia przy takiej strukturze oprogramowania wówczas wykorzystuje się systemy operacyjne czasu rzeczywistego oparte o wielozadaniowe jądro z wywłaszczaniem lub – w prostych systemach wbudowanych – tylko jądro [1]. Oprogramowanie realizujące funkcje systemu wbudowanego dzieli się na zadania, którym nadaje się priorytety określające ich stopień ważności. Zadania znajdują się w pewnych stanach. Do wykonania mogą być kierowane tylko te, które osiągnęły stan gotowości, np. wskutek wystąpienia zdarzenia. Po zmianie stanu któregoś z zadań wywoływany jest podprogram szeregujący jądra, który spośród zadań gotowych uruchamia to o najwyższym priorytecie. Wykorzystanie systemu operacyjnego czasu rzeczywistego lub jądra wprowadza jednak pewne narzuty pamięciowe oraz czasowe.

2. Jądro Xilkernel

Xilkernel to wielozadaniowe jądro czasu rzeczywistego dla systemów wbudowanych opartych na procesorach: PowerPC 405 i 440 oraz Microblaze, implementowanych w układach FPGA firmy Xilinx. Jest to biblioteka funkcji dostarczana wraz ze środowiskiem XPS (*Xilinx Platform Studio*), większość ich kodu jest napisana w języku C. Przed użyciem jądra programista musi określić m. in. algorytm szeregowania zadań, które z usług jądra będą wykorzystywane w aplikacji, rozmiar kolejek zadań (gotowych, oczekujących na zwolnienie semafora, zawieszonych w oczekiwaniu na upływ zadanego odcinka czasu itd.). Do wyboru są dwa algorytmy szeregowania zadań: priorytetowy oraz karuzelowy (ang. *round-robin*). Jądro Xilkernel realizuje funkcje usługowe umożliwiające:

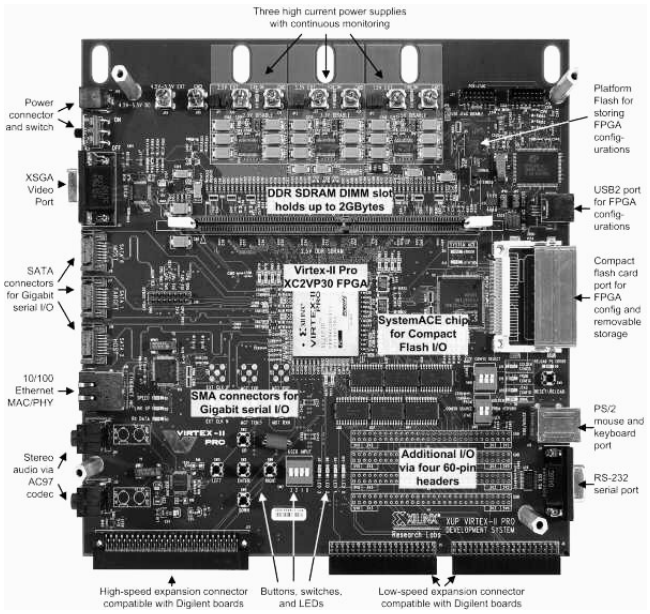
- samowywłaszczenie zadania,
- komunikację między zadaniami przy użyciu semaforów i muteksów oraz poprzez przesył komunikatów i pamięć dzieloną,
- zawieszenie zadania na określony odcinek czasu,
- rezerwację i zwalnianie bloków pamięci.

Interfejs programistyczny w większości przypadków jest zgodny ze standardem POSIX [2].

3. Narzuty czasowe wprowadzane przez jądro

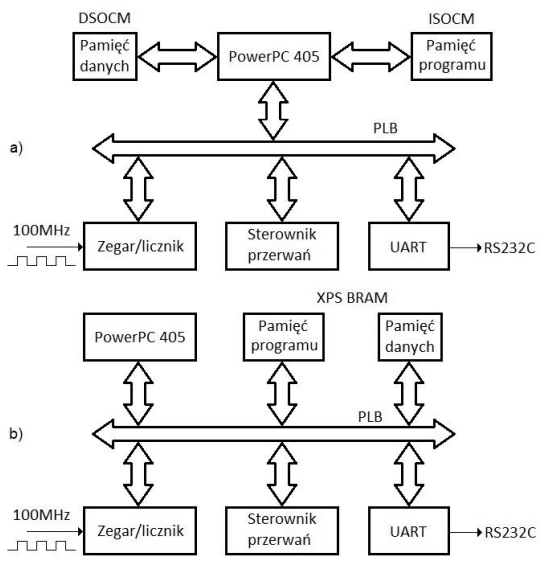
3.1. Stanowisko pomiarowe

Pomiary wielkości narzutów czasowych przyprowadzono przy wykorzystaniu zestawu uruchomieniowego z układem FPGA z rodziny Virtex-II Pro, w którym zaimplementowano system z procesorem PowerPC 405 (rysunek 1) [3]. Zewnętrzny dla procesora programowalny układ zegara-licznika był używany do pomiaru czasu wykonywania operacji przez jądro [4]. Pomiar polegał na zliczaniu podawanych na wejście tego układu impulsów o ustalonej częstotliwości. Wyniki zliczania po przekształceniu do postaci tekstowej były przesyłane przez łącze szeregowe RS232C do komputera PC i zapisywane na dysku, ich obróbka odbywała się w środowisku Matlab.



Rys. 1. Zestaw uruchomieniowy Virtex-II Pro
Fig. 1. Virtex-II Pro Development System

Warunki pomiarów były następujące:
- częstotliwość taktowania procesora - 300 MHz,
- częstotliwość impulsów zliczanych przez układ zegara-licznika - 100 MHz, co dawało rozdzielczość pomiaru równą 0.01 μs,
- program i dane w wewnętrznej pamięci FPGA, z możliwych wariantów ich rozmieszczenia wybrano dwa:
  - w blokach ISOCM (ang. *Instruction-Side On-Chip Memory*) oraz DSOCM (ang. *Data-Side On-Chip Memory*), podłączonych przez dedykowane magistrale (rysunek 2a),
  - w blokach XPS BRAM (ang. *XPS Block RAM*), podłączonych do magistrali PLB (ang. *Processor Local Bus,* rysunek 2b),
- wersja jądra – 4.00.a,
- algorytm szeregowania zadań – priorytetowy,
- wykonywano serie 1200 pomiarów.



Rys. 2. Konfiguracje systemu wykorzystywane w badaniach
Fig. 2. System configurations used for research

3.2. Obsługa przerwań

Po przyjęciu żądania przerwania zachowywany jest kontekst przerwanego zadania, następuje przełączenie na stos jądra i wywołanie podprogramu obsługi przerwania (ISR). W wyniku jego wykonania pewne zadanie może osiągnąć stan gotowości. W takim przypadku wywoływany jest podprogram szeregujący, który do dalszej realizacji wybierze zadanie gotowe o najwyższym priorytecie. Na koniec następuje odtworzenie kontekstu wybranego zadania i jego uruchomienie. Całość odbywa się przy zablokowanych przerwaniach.

Zmierzono czasy opóźnień od zgłoszenia żądania przerwania do uruchomienia podprogramu obsługi przerwania (ang. *interrupt latency*) oraz zadania (ang. *task latency*) [5]. Dla dokonania pomiaru układ zegara licznika został zaprogramowany do zliczania impulsów w górę i cyklicznego zgłaszania żądań przerwania. Zliczanie w każdym cyklu przebiegało od pewnej wartości początkowej. Do sygnalizowania wystąpienia żądania posługiwano się semaforem. Po uruchomieniu ISR oraz zadania była odczytywana zawartość rejestru danych układu zegara-licznika. Wyniki pomiarów stanowiły różnice między odczytanymi wartościami a wartością początkową. Wartości czasów opóźnienia zamieszczono w tabeli 1.

Tab. 1. Czasy opóźnień uruchomienia ISR oraz zadania obsługi zdarzenia
Tab. 1. Time delay to start the interrupt service routine and the task

Rozmieszczenie programu i danych	Czas opóźnienia uruchomienia	
	ISR	zadania
bloki ISOCM i DSOCM	1.89÷5.47 μs	19.99÷24.35 μs
bloki XPS BRAM	2.72÷2.79 μs[1] 5.32÷5.42 μs[2]	43.06÷49.53 μs[1] 54.11÷68.48 μs[2]

1. przy odblokowanej pamięci podręcznej
2. przy zablokowanej pamięci podręcznej

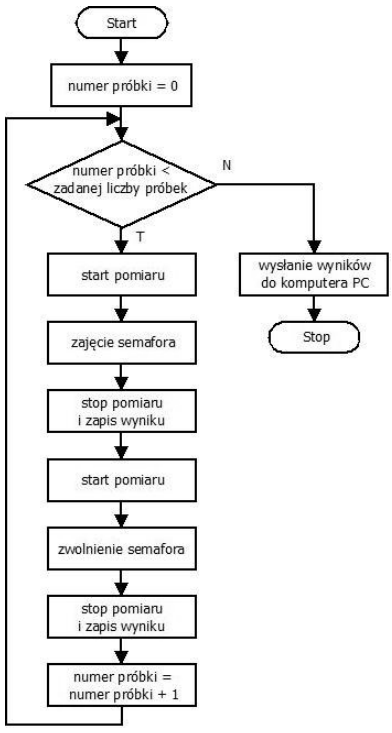
3.3. Usługi systemowe

W tabeli 2 zamieszczono wyniki pomiaru czasu realizacji usług systemowych [5]. Aplikacja do badania usług 3÷10 składała się z zadania testowego oraz z systemowego zadania tracenia czasu, o niższym priorytecie. Na rysunku 3 przedstawiono schemat blokowy zadania dla przetestowania usług semaforowych, dla pozostałych przypadków był on podobny. Pominięto usługi związane z pamięcią dzieloną, ponieważ odpowiednie funkcje usługowe służą tylko do jej konfigurowania.

Tab. 2. Czasy realizacji funkcji usługowych jądra Xilkernel
Tab. 2. Execution times of Xilkernel services

Lp.	Usługa systemowa	Czas wykonania		
		A	b	C
1.	samowywłaszczenie zadania	13.28÷17.66 μs	27.03÷33.82 μs	36.29÷50.42 μs
2.	zawieszenie zadania na określony odcinek czasu	13.44 μs	26.47 μs	65.08 μs
3.	zajęcie semafora	2.04 μs	4.08÷4.19 μs	5.88 μs
4.	zwolnienie semafora	2.10 μs	4.22÷10.72 μs	5.84÷20.01 μs
5.	zajęcie muteksu	2.32 μs	4.87÷4.92 μs	6.64÷20.77 μs
6.	zwolnienie muteksu	1.94 μs	4.21÷10.81 μs	5.42 μs
7.	wysłanie komunikatu o długości 16 bajtów	7÷11.24 μs	15.04÷42.64 μs	19.29÷60.69 μs
8.	odbiór komunikatu o długości 16 bajtów	6.08 μs	12.83 μs	16.84÷58.06 μs
9.	rezerwacja bloku pamięci o rozmiarze 16 bajtów	2.14 μs	4.44÷4.49 μs	5.85÷19.93 μs
10.	zwolnienie bloku pamięci	2 μs	4.11 μs	5 μs

a) program i dane w blokach ISOCM i DSOCM
b) program i dane w blokach XPS BRAM, odblokowana pamięć podręczna
c) jak w punkcie b, zablokowana pamięć podręczna



Rys. 3. Schemat blokowy zadania dla testu operacji semaforowych
Fig. 3. Flowchart of the task for semaphore operations testing

4. Wymagania pamięciowe jądra

Rozmiar kodu wynikowego jądra zależy od tego, które usług systemowych są wykorzystywane w aplikacji. Kod ten może zająć od 16 do 32 kB pamięci [2]. Na rozmiar potrzebnej pamięci danych wpływ ma wiele czynników m. in. liczba poziomów priorytetu, długość kolejek zadań, długość kolejek komunikatów, liczba czasomierzy.

5. Wnioski

Zastosowanie wielozadaniowego jądra z wywłaszczaniem upraszcza projektowanie i późniejszą rozbudowę oprogramowania systemu wbudowanego. Wykonanie zadania obsługi zdarzenia o wysokim stopniu ważności nie jest odwlekane do chwili zakończenia obsługi zdarzeń mniej ważnych. Ceną za to jest większe zapotrzebowanie na pamięć programu i danych oraz czas procesora. W artykule przedstawiono wyniki badań nad wielkością tych narzutów wnoszonych przez jądro Xilkernel. Znajomość narzutów czasowych powinna pomóc w oszacowaniu, czy system będzie w stanie spełnić stawiane mu wymagania co do czasu reakcji na zdarzenia.

6. Literatura

[1] Szymczyk P.: Systemy operacyjne czasu rzeczywistego. Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH, Kraków 2003.
[2] Xilinx: OS and Libraries Document Collection. September 2008. www.xilinx.com/support/documentation/sw_manuals/edk10_oslib_rm.pdf
[3] Xilinx: XUP Virtex-II Pro Development System. Hardware Reference Manual. July 2009. http://www.xilinx.com/univ/XUPV2P/Documentation/ug069.pdf
[4] Stewart D. B.: Measuring execution time and real-time performance. Embedded Systems Conference, Boston, September 2006.
[5] Carbone J., Lamie W.: Measure your RTOS's real-time performance. EE Times, May 2007. http://www.eetimes.com/design/automotive-design/4007081/Measure-your-RTOS-s-real-time-performance

otrzymano / received: 02.12.2010
przyjęto do druku / accepted: 02.02.2011 *artykuł recenzowany*

INFORMACJE

Nowa inicjatywa PAK

Na stronie internetowej Wydawnictwa PAK został utworzony dział: **Niepewność wyników pomiarów** w którym są zamieszczane aktualne informacje dotyczące problemów teoretycznych i praktycznych związanych z szacowaniem niepewności wyników pomiarów. W dziale znajdują się:

- aktualne informacje o publikacjach dotyczących niepewności wyników,
- informacje o przedsięwzięciach naukowo–technicznych i edukacyjnych, o tematyce związanej z niepewnością,
- dokumenty dotyczące niepewności,
- pytania do ekspertów (FAQs).

Zapraszamy:

- autorów opublikowanych prac dotyczących niepewności o nadsyłanie tekstów do zamieszczenia w tym dziale,
- organizatorów przedsięwzięć naukowo – technicznych lub edukacyjnych do nadsyłania informacji o imprezach planowanych lub odbytych,
- zainteresowanych zagadnieniami szczegółowymi do nadsyłania pytań do ekspertów.

Materiały mogą mieć formę plików lub linków do źródeł. Warunkiem zamieszczenia w tym dziale strony internetowej PAK materiałów lub linków jest przysłanie do redakcji PAK pocztą zwykłą zgody właściciela praw autorskich na takie rozpowszechnienie. Zamieszczanie i pobieranie materiałów i informacji w tym dziale strony internetowej jest bezpłatne. Redakcja PAK będzie nadzorować zawartość działu, ale za szczegółowe treści merytoryczne odpowiadają autorzy nadsyłanych materiałów.

Tadeusz SKUBIS
Redaktor naczelny Wydawnictwa PAK