SYSTEMY OPERACYJNE PROCESORÓW SYGNAŁOWYCH

Opracowanie: Grzegorz Szwoch

Politechnika Gdańska, Katedra Systemów Multimedialnych

Programowanie bez OS

- Proste programy na DSP nie potrzebują systemu operacyjnego.
- Program taki wykonuje po kolei zaplanowane operacje.
- Taki sposób programowania nazywa się *bare metal* - bezpośrednio na procesorze.
- Wada: jeżeli program oczekuje na dane, nie może wykonywać w tym czasie innych operacji.
- Dla bardziej złożonych programów jest to niewydajne: dostępny czas przetwarzania zamienia się na czas bezczynności.

Operacje wejścia/wyjścia

- Operacje wejścia/wyjścia (I/O): pobieranie danych wejściowych, wysyłanie wyników na wyjście.
- Powodują zawieszenie wykonywania programu – czekanie na nadejście danych.
- Problematyczne gdy program pobiera dane z kilku wejść.
- Chcemy wykonywać obliczenia w czasie gdy czekamy na nowe dane.
- Rozwiązanie: podzielenie programu na wątki.
- Do zarządzania wątkami potrzebny jest system operacyjny.
- Na DSP, system operacyjny jest dostarczany przez producenta, jego kod jest łączony z programem.

Program i wątki

- Program – samodzielna jednostka zawierająca operacje wykonywane na DSP.
  Jeden DSP (lub jeden rdzeń DSP) = jeden program.
- Wątek (*thread*) – wyodrębniona część programu.
- W ramach programu może działać wiele wątków.
- Nazywamy to programowaniem wielowątkowym (*multithtreading*) lub współbieżnym (*concurrency*).
- Wątki rywalizują o dostęp do zasobów (*resources*), np. cykli procesora, buforów w pamięci.
- System operacyjny na DSP: nadrzędny wątek zarządzający działaniem innych wątków i ich dostępem do zasobów.

Przykład wątków

Wątek 1 – odczyt danych:

- odczytuje dane wejściowe, np. z czujnika
- przekazuje je wątkowi 2,
- zasypia, czekając na nowe dane

Wątek 2 – przetwarzanie:

- odbiera dane od wątku 1,
- wykonuje obliczenia i przekazuje wynik na wyjście,
- czeka na kolejne dane

Współbieżność a zrównoleglenie

Często mylone pojęcia:

Współbieżność (*concurrency*):

- wiele wątków jest uruchomionych,
- liczba wątków wykonujących operacje w danej chwili ≤ liczba dostępnych DSP / rdzeni DSP,
- jeden rdzeń DSP – jeden wątek wykonuje operacje, reszta wątków jest uśpiona.

Zrównoleglenie (*parallelism*):

- kilka wątków jednocześnie wykonuje operacje,
- wymaga wielordzeniowego DSP lub układu wielu DSP.

Typy wielozadaniowości

Wielozadaniowość współpracująca (*cooperative multitasking*):

- wątek musi zawiesić swoje wykonywanie aby inny wątek mógł zacząć działać,
- stosowany w programach, w których programista ma pełną kontrolę nad działaniem wątków,
- czasami stosowane w systemach wbudowanych.

Wielozadaniowość z wywłaszczaniem (*preemptive multitasking*):

- system operacyjny zarządza pracą wątków,
- każdy wątek może w dowolnym momencie zostać zatrzymany (wywłaszczony) na rzecz innego wątku,
- bardziej wydajny pod kątem wykorzystania zasobów,
- stosowany na PC i w większości programów na DSP.

Przełączanie kontekstu

- Kontekst (*context*) – uruchomiony wątek i przydzielone mu zasoby.
- Przełączenie kontekstu (*context switch*) – zawieszenie działania wątku i przekazanie zasobów innemu wątkowi.
- Priorytet (*priority*) – liczba określająca hierarchię wątków.
- Przełączenie kontekstu następuje najczęściej w dwóch przypadkach:
  - wątek zasypia (*sleep*), np. czekając na dane (dobrowolne oddanie zasobów),
  - wątek o wyższym priorytecie niż aktualnie działający żąda dostępu do zasobów (wywłaszczenie wątku).

Konflikty dostępu do pamięci

Rywalizacja wątków o dostęp do pamięci może powodować konflikty. Przykład:

Wątek 1 (wyższy priorytet):

- (czeka)
- wznawia działanie
- zapisuje nowe dane do tablicy
- zawiesza działanie
- (czeka)

Wątek 2 (niższy priorytet):

- odczytuje połowę tablicy w pamięci
- wywłaszczenie przez W1
- (czeka)
- wznawia działanie
- odczytuje drugą połowę tablicy – dane są błędne!

Muteks

- Muteks (*mutex*) – obiekt pozwalający na wyłączny dostęp wątku do zasobu.
- Wątek, który chce uzyskać dostęp do zasobu, musi pobrać i zablokować muteks.
- Jeśli muteks jest zablokowany przez inny wątek, pierwszy wątek czeka lub wykonuje w tym czasie inne operacje.
- Muteks należy zwolnić po wykonaniu operacji.
- Sekcja krytyczna (*critical section*) – fragment kodu zabezpieczony muteksem, który powinien być wykonany w całości przez jeden wątek.
- Stosowanie muteksów spowalnia program. Należy je stosować tam, gdzie są konieczne.

Muteks - przykład

Wątek 1:

- blokuje Muteks
- odczytuje tablicę z pamięci
- zwalnia Muteks

Wątek 2:

- …
- czeka na Muteks
- blokuje Muteks
- zapisuje nowe dane do tablicy
- zwalnia Muteks

Semafor

- Semafor (*semaphore*) jest obiektem, który umożliwia dostęp wielu wątków do ograniczonej liczby zasobów.
- Semafor posiada licznik (*counter*).
- Pobranie semafora zmniejsza licznik o 1,
  zwolnienie – zwiększa o 1.
- Nie można pobrać zasobu, gdy licznik jest równy 0.
- Przykład: jest 5 buforów w pamięci
  - aktualny licznik: 1 (cztery bufory już pobrane),
  - wątek 1 pobiera bufor, licznik: 0,
  - wątek 2 nie może już pobrać bufora, musi poczekać na zwolnienie go przez inny wątek.

Kolejki

- Wątki potrzebują przekazywać dane między sobą.
- Kolejka (*queue*) jest strukturą do przekazywania danych między wątkami.
- Typowy schemat przetwarzania – „producent-konsument”:
  - jeden wątek („producent”) generuje dane (np. pobiera z interfejsu) i wstawia je do kolejki,
  - drugi wątek („konsument”) wyjmuje dane z kolejki i przetwarza je.
- Kolejka jest zwykle chroniona wewnętrznym muteksem.
- Nie można wstawić danych gdy kolejka jest pełna.

Przykład kolejki

Wątek 1 (producent):

- odczytuje dane z wejścia
- wstawia dane do kolejki
- czeka na nowe dane

Wątek 2 (konsument):

- próbuje pobrać dane z k.
- gdy kolejka pusta – czeka
- przetwarza pobrane dane

Optymalizacja kolejek

- Programista powinien zadbać o zrównoważenie czasu wykonywania wątków.
- Kolejka nie powinna być nigdy ani pełna, ani pusta.
- Jeżeli kolejka jest zapisywana częściej niż odczytywana: przepełnienie kolejki *(overflow),* dane mogą zostać utracone.
- Jeżeli jest odczytywana częściej niż zapisywana: efekt niedopełnienia (*underrun*) lub „zagłodzenia” wątku (*starvation*), marnowanie cykli procesora.

Zdarzenia

- Zdarzenie (*event*) jest sygnałem przesyłanym do innych wątków.
- Najczęściej komunikuje o dostępności nowych danych.
- Może mieć dwa stany: włączone lub wyłączone.
- Wątek może „zasnąć” i zostać automatycznie wybudzony gdy nadejdzie zdarzenie.
- Po odebraniu zdarzenia, wątek powinien je wyłączyć.
- Zdarzenie eliminuje konieczność cyklicznego sprawdzania czy są nowe dane.

Przerwania

- Przerwanie sprzętowe (*hardware interrupt*) jest sygnałem generowanym przez sprzęt, np. gdy pojawią się nowe dane na interfejsie.
- Obsłużenie przerwania – przez wątek, który jest przypisany do przerwania; pobiera on dane z wejścia.
- Przerwania programowe (*software interrupt*) – generowane przez programistę.
- Przerwania mają wyższy priorytet niż wątki, dlatego obsługa przerwania wywłaszcza inne wątki.
- Przerwania niemaskowalne – nie mogą zostać wyłączone, np. sygnał RESET.

Zakleszczenie wątków

Sytuacja, której należy bezwzględnie unikać:

Wątek 1:

- blokuje Muteks A
- czeka na Muteks B

Wątek 2:

- blokuje Muteks B
- czeka na Muteks A

- Oba wątki zostają zawieszone w oczekiwaniu na zasoby, których nie mają szansy uzyskać.
- Powstaje zakleszczenie wątków (*deadlock*).
- Zwykle prowadzi to do zawieszenia się programu.

Problemy programowania współbieżnego

- Mówi się, że programy wielowątkowe są niedeterministyczne, ponieważ ich działanie zależy od zależności czasowych pomiędzy wątkami, np. kiedy zostanie przełączony kontekst.
- Wyścig (*race*) do zasobów – sukces w pobraniu zasobu zależy od tego, czy wątek zdąży przed innym wątkiem.
- Zakleszczenie może zdarzyć się np. raz na 20 uruchomień.
- Debugowanie programów wielowątkowych jest bardzo trudne.
- Programista musi minimalizować ryzyko wystąpienia konfliktowych sytuacji.

Systemy operacyjne na DSP

Przykład – procesory Texas Instruments

- System operacyjny czasu rzeczywistego, pod różnymi nazwami: DSP/BIOS, SYS/BIOS, TI-RTOS.
- Umożliwia uruchamianie wielowątkowych programów na jednym rdzeniu DSP.
- Zapewnia mechanizmy synchronizacji wątków.
- System jest łączony z kodem programisty w jeden program wykonywalny.

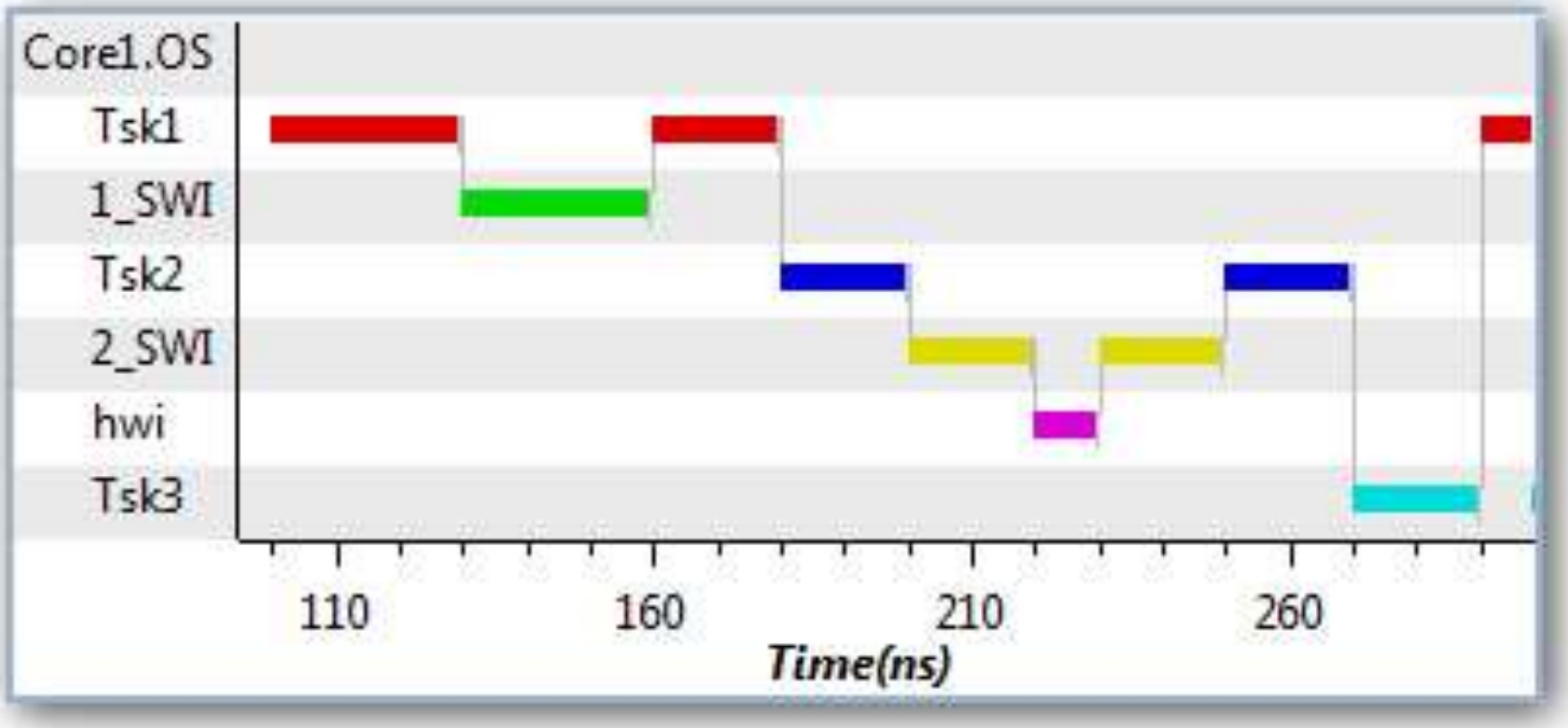
Terminologia

Terminy stosowane przez DSP/BIOS i SYS/BIOS:

- *Interrupt* (przerwanie) – wątek obsługi przerwania,
- *Task* (zadanie) – wątek nie związany z przerwaniami,
- *Idle loop* – wątek tła o najniższym priorytecie, uruchamiany tylko gdy wszystkie inne wątki są bezczynne,
- *Semaphore* – semafor, *Gate* (bramka) – muteks,
- *Mailbox* (skrzynka pocztowa) – struktura do przekazywania komunikatów między wątkami,
- *Queue* – kolejka dla danych,
- *Memory section manager* – moduł zarządzania dynamicznie alokowaną pamięcią,
- *Pipe* (potok), *stream* (strumień) – wymiana danych między interfejsami a pamięcią.

Przykład działania

Wątki obsługi przerwań sprzętowych (hwi) i programowych (SWI) oraz wątki utworzone przez programistę (Tsk)

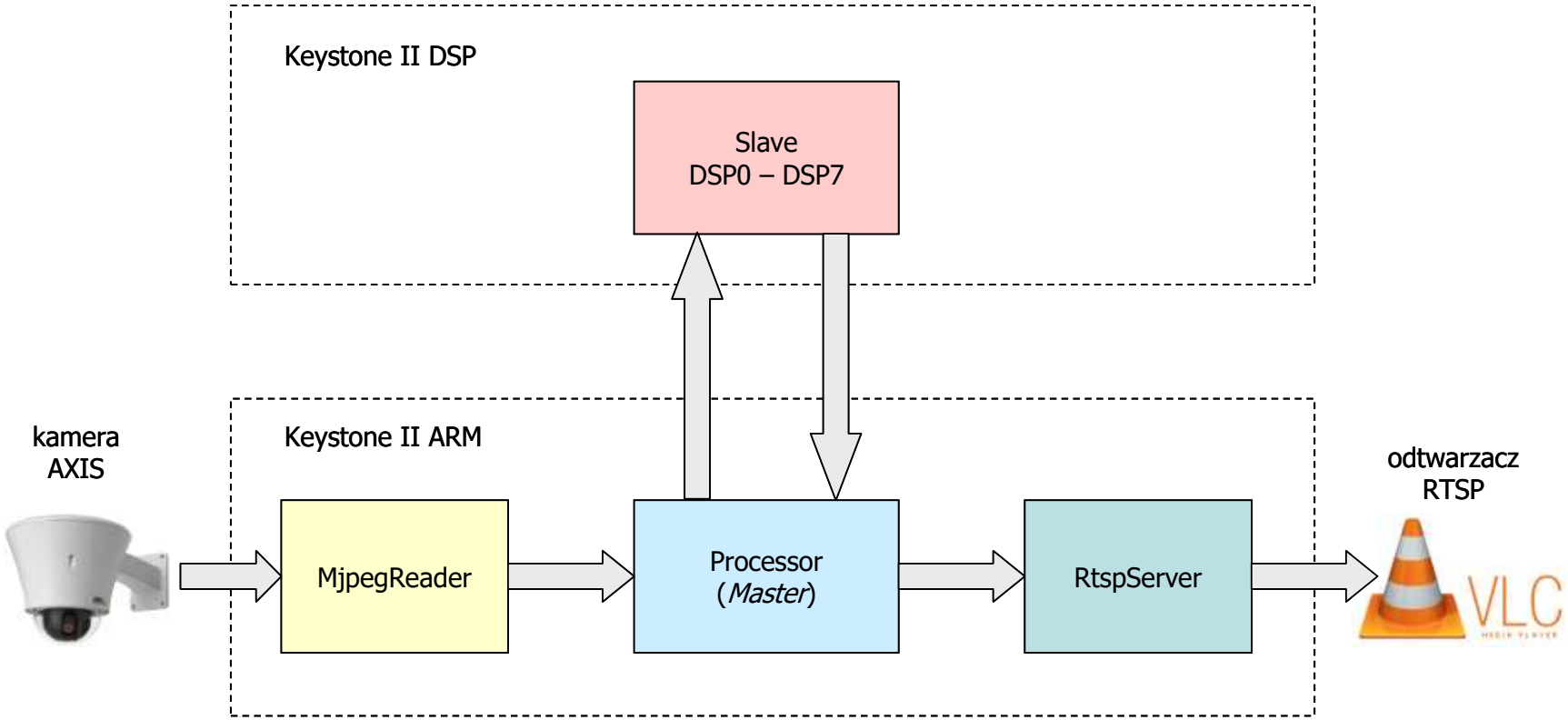


Systemy na procesorach hybrydowych

- Hybrydowe procesory DSP – zawierają rdzenie ARM i DSP.
- Przykład: TI Keystone C66x+ARM.
- Przetwarzanie na takim procesorze odbywa się w trybie *master-slave* (zarządca i wykonawca).
- Na procesorze ARM jest uruchamiany system operacyjny:
  - dedykowany dla procesora (np. TI-RTOS),
  - lub Linux z jądrem czasu rzeczywistego (np. Yocto).
- Program na rdzeniu ARM: zarządca, kieruje dane do wykonawców, odbiera wyniki.
- Program na rdzeniach DSP: wykonawca, przetwarzanie danych.
- Do wymiany danych M-S służą kolejki.

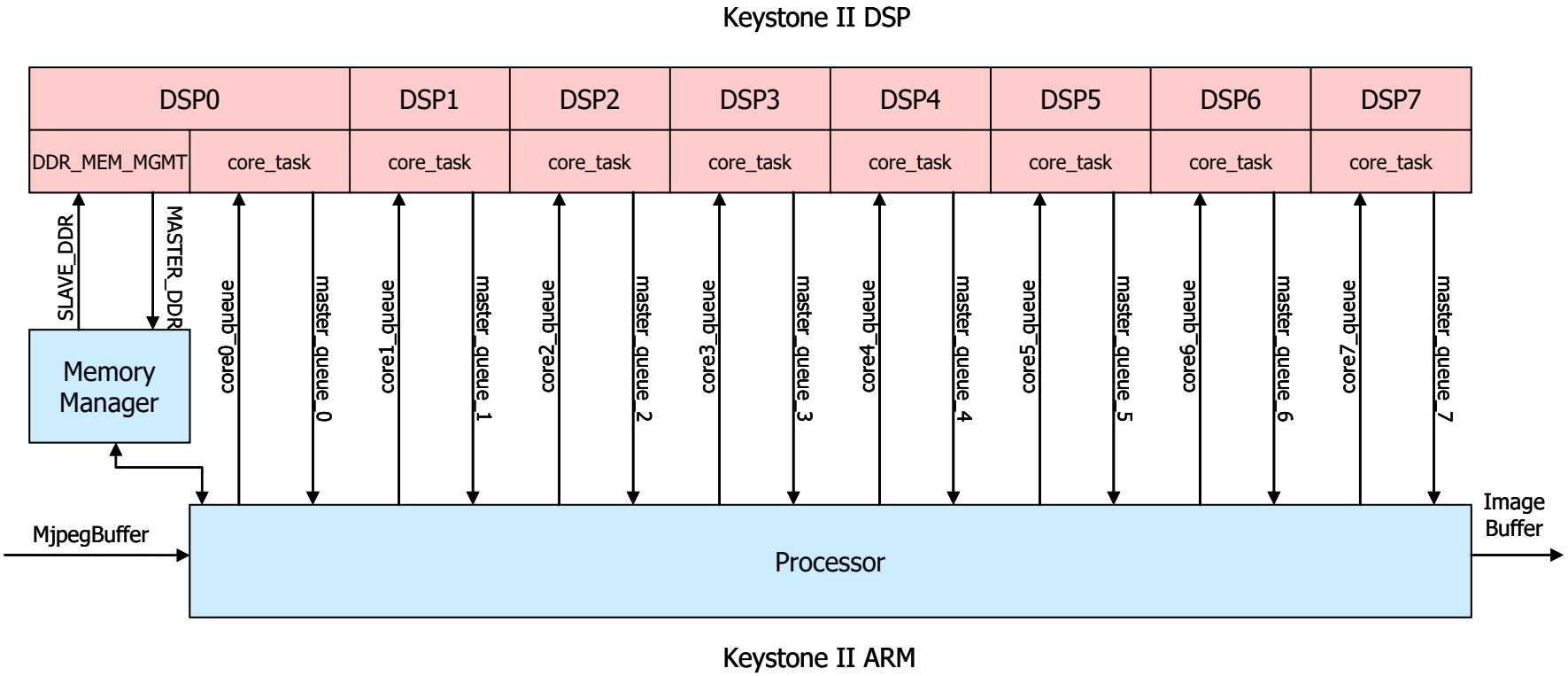
Systemy na procesorach hybrydowych

Przykład: przetwarzanie obrazu z kamery. Każdy rdzeń DSP dostaje swój „kawałek” obrazu.



Systemy na procesorach hybrydowych

Wymiana danych między rdzeniami za pomocą kolejek:



Podsumowanie

- Proste programy na DSP, takie jak na projekcie ZPS - programowanie *bare metal* wystarcza.
- Jeżeli program spędza zbyt dużo czasu czekając na dane i brakuje cykli do obliczeń – potrzebny system operacyjny.
- Programowanie wielowątkowe jest trudniejsze niż klasyczne programy jednowątkowe – wiele rzeczy może pójść źle.
- Należy dobrze przemyśleć podział zadań między wątki.
- System operacyjny wykonuje za nas zarządzanie wątkami.
- Procesory DSP+ARM dają potencjalnie większą wydajność i elastyczność, kosztem zwiększonego narzutu przetwarzania.