Operating System Practice

Che-Wei Chang

chewei@mail.cgu.edu.tw

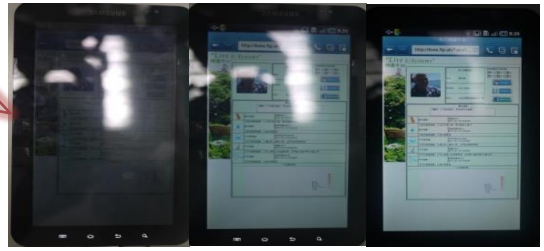
Department of Computer Science and Information Engineering, Chang Gung University

Energy Saving Designs

Two Approaches to Reduce the Power Consumption of Devices

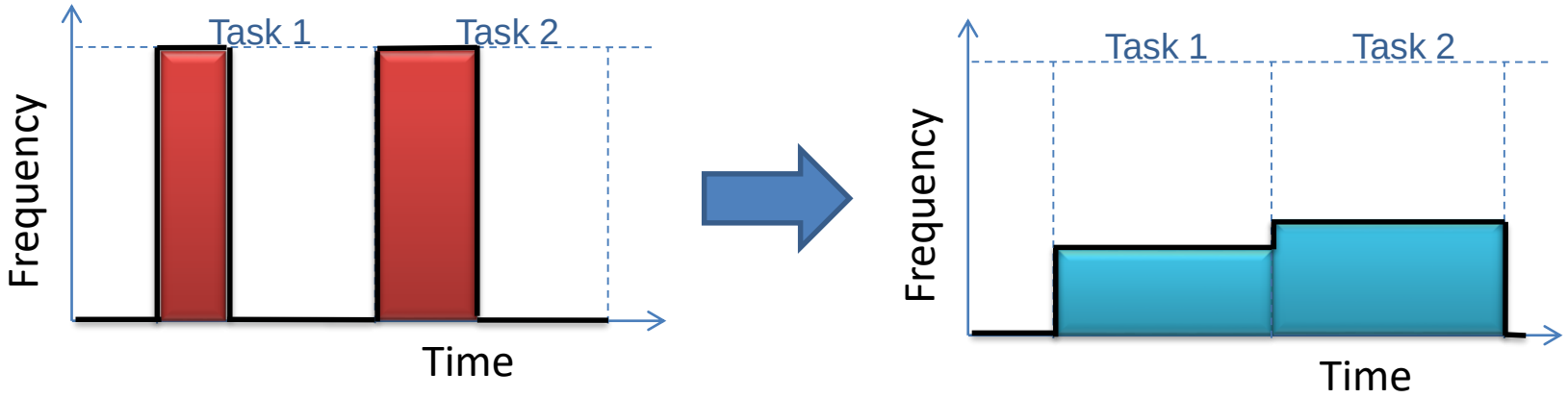
- Dynamic Voltage and Frequency Scaling (DVFS)
  - Scale down the voltage and/or frequency to reduce the processor power consumption
  - An example: reducing 17% of the energy consumption for H.264 decoding over TI DaVince DM6446
- Dynamic Power Management (DPM)
  - Change to an energy-efficient state to reduce the power consumption of peripheral devices
  - An example: reducing 15% of the energy consumption for the browser over Android Galaxy Tab

Power Consumption	1625mW	2023mW	3166mW

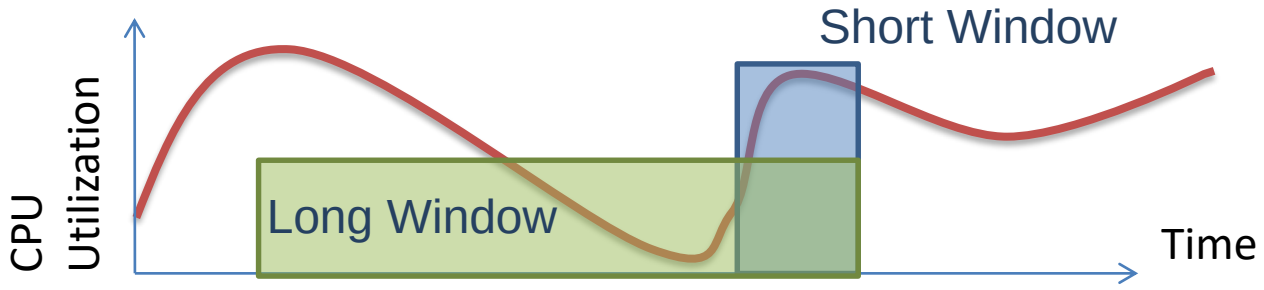


Dynamic Frequency and Voltage Scaling Designs

- The Observation: It is energy-efficient to scale down the processor frequency dynamically to fit current workloads



- The Approach: It keeps a window or buffer to estimate the workload



Energy-Efficient Multimedia Platforms

- Energy-Efficient ARM-DSP Platforms for H.264 Decoding
  - Analyze the workload variance of multimedia jobs
  - Take real platforms for an ARM+DSP design
  - Achieve more than 17% energy saving for the ITRI PAC SoC

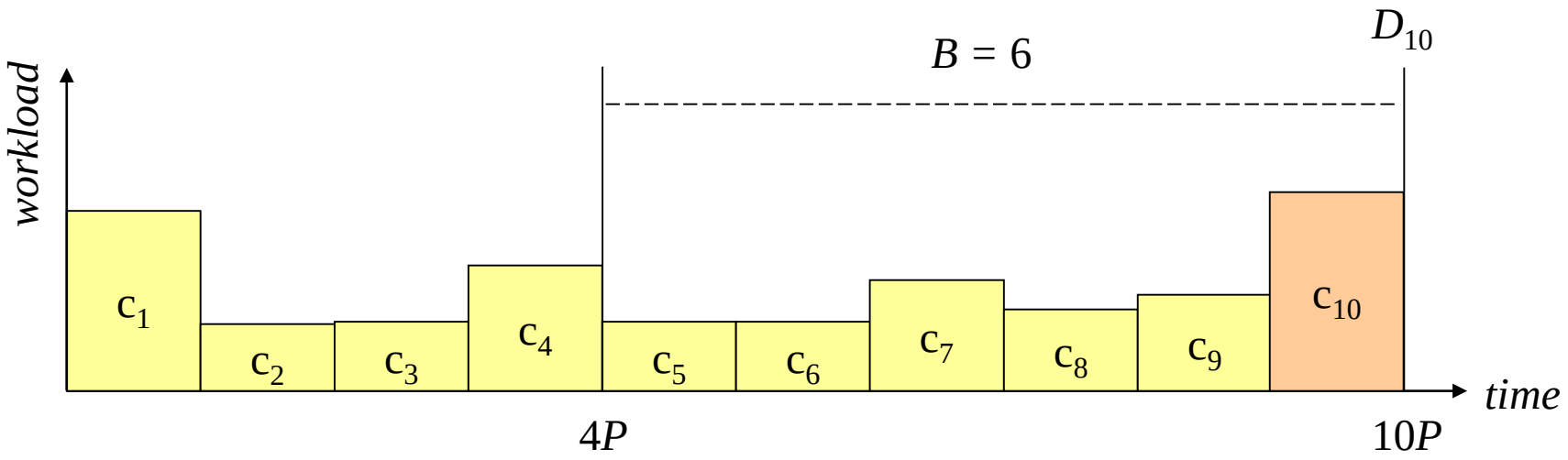
Implementation on a PC with Phenom II

Implementation on an Android Phone

Implementation on the PAC Soc

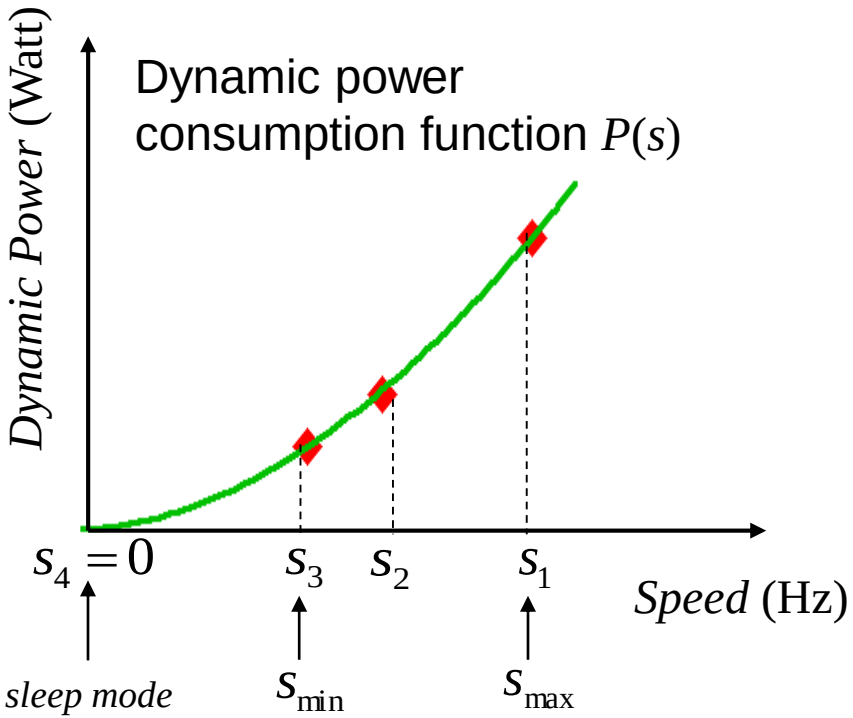
Examples of Energy Saving Designs

Task Model of Multimedia Applications



- A task consists of a sequence of jobs
- Jobs arrive by a period: P time units
- The workload of future job can be predicted based on the log
- Because of the buffer size limitation, the number of pending jobs at any time point is bounded by a fixed integer B

Processor Power Model



$$P(s) = C_{ef} V_{dd}^{\ 2} s,$$

$$s = k \frac{(V_{dd} - V_t)^2}{V_{dd}}$$

where

C_{ef} is the effective switch capacitance

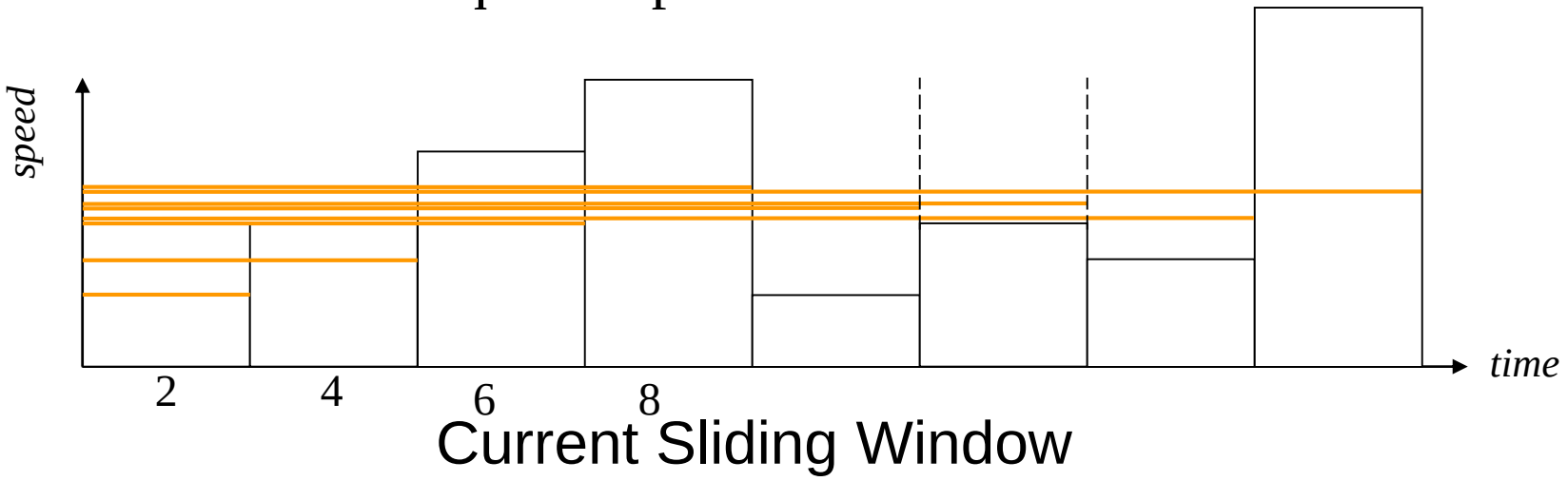
V_t is the threshold voltage

V_{dd} is the supply voltage

k is a hardware-design-specific constant

Concepts of DVFS Algorithm

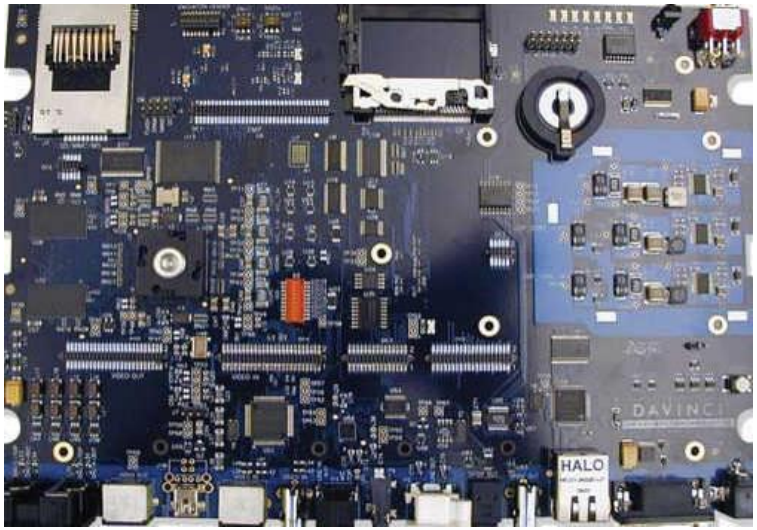
- An Energy-Efficient Multimedia Application
  - ➔Make the frequency as low as possible
  - ➔Meet all performance constraints
  - ➔Find the critical (most demanding) interval by calculating the maximum required speed in all intervals



Hardware Platform

- Platform: Texas Instruments DaVinci DVEVM (The Digital Video Evaluation Module)

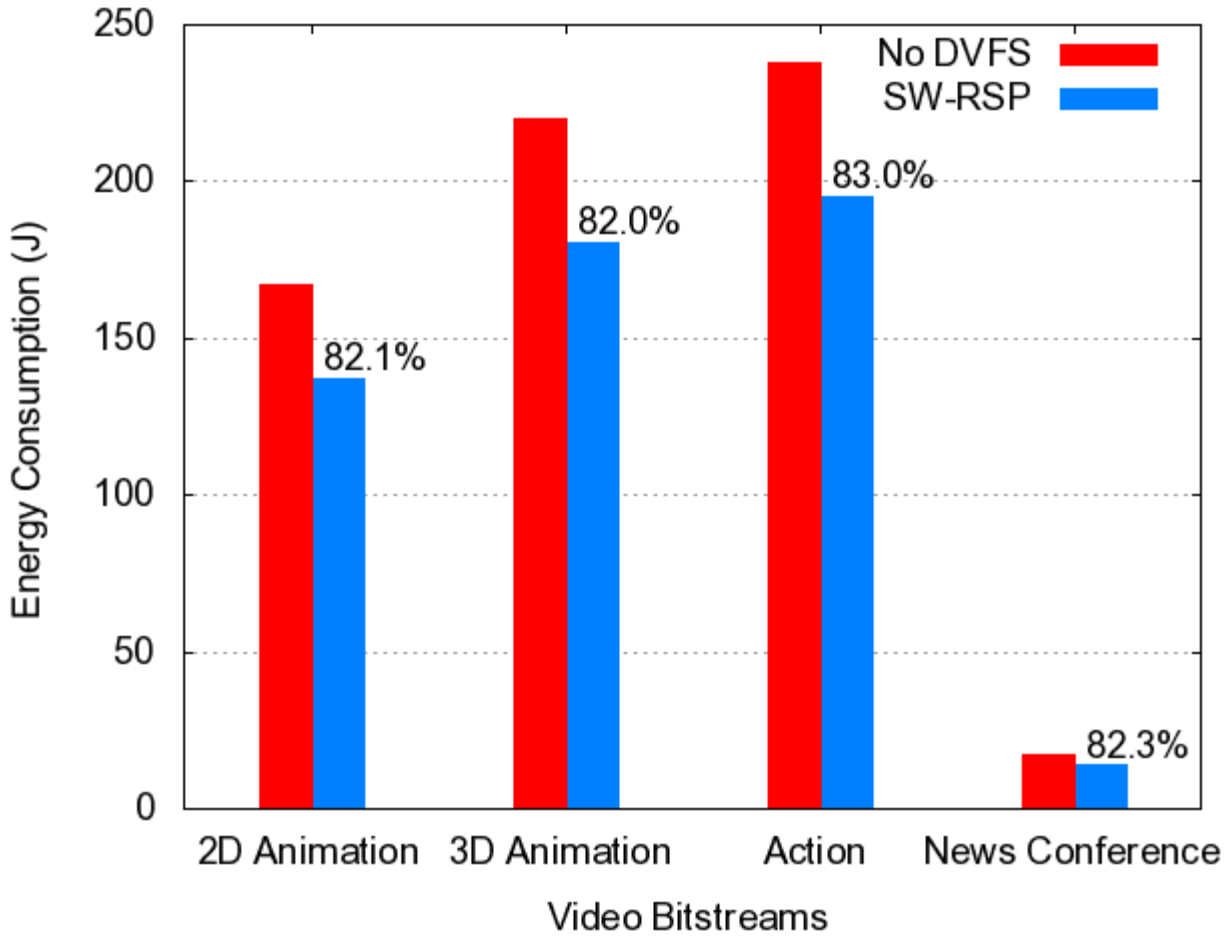
Speed Mode	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
Frequency (MHz)	594	567	540	513	486	459	432	405	0



Input Video Streams

	2D Animation	3D Animation	Action	News Conf.
				
Number of frames	17985	19182	19182	1751
FPS	29.97	29.97	23.98	23.98
Bit-rate (kbps)	754.99	1237.58	1798.87	631.27
PSNR	42.44	41.85	40.04	42.49
Resolution	720×480	720×480	720×480	720×480

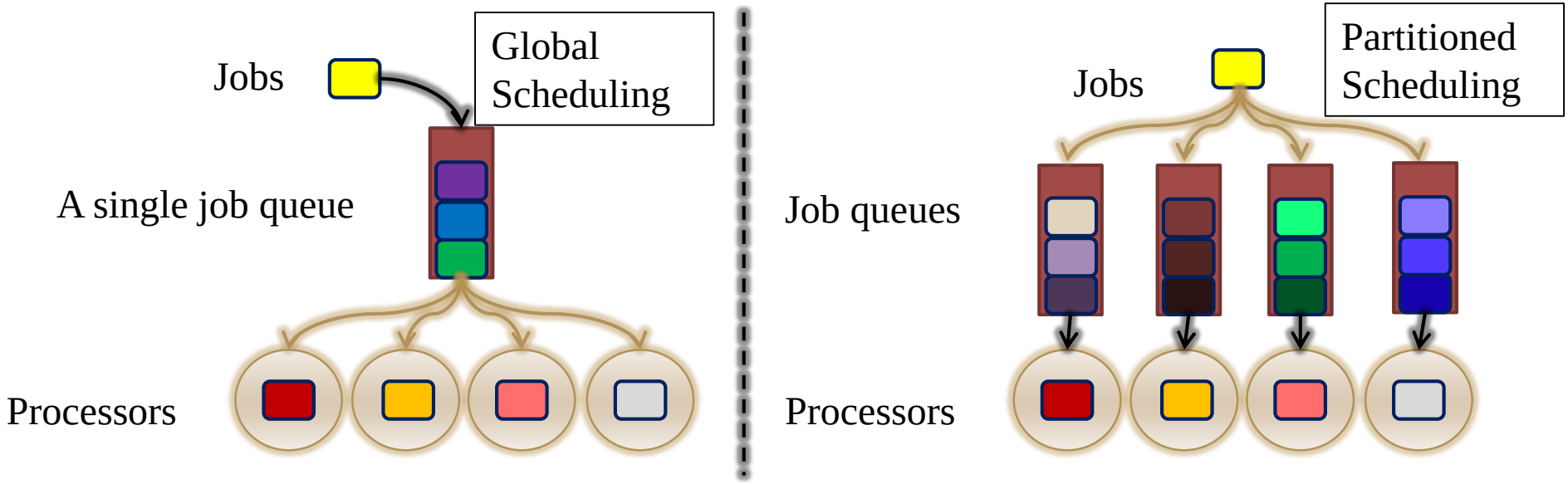
Experimental Results



Multi-Core Scheduling

Scheduling on Multiple Cores (1/2)

- Real-Time Multi-Core Scheduling Algorithms [1]
  - Global scheduling algorithms [2]
  - Partitioned scheduling algorithms [3]



[1] Robert I. Davis and Alan Burns, "A Survey of Hard Real-Time Scheduling for Multiprocessor Systems," in *ACM Computing Surveys*, 2011.
[2] Hennadiy Leontyev and James H. Anderson, "Generalized Tardiness Bounds for Global Multiprocessor Scheduling," in *RTSS*, 2007.
[3] Sanjoy Baruah and Nathan Fisher, "The Partitioned Multiprocessor Scheduling of Sporadic Task Systems," in *RTSS*, 2005.

Scheduling on Multiple Cores (2/2)

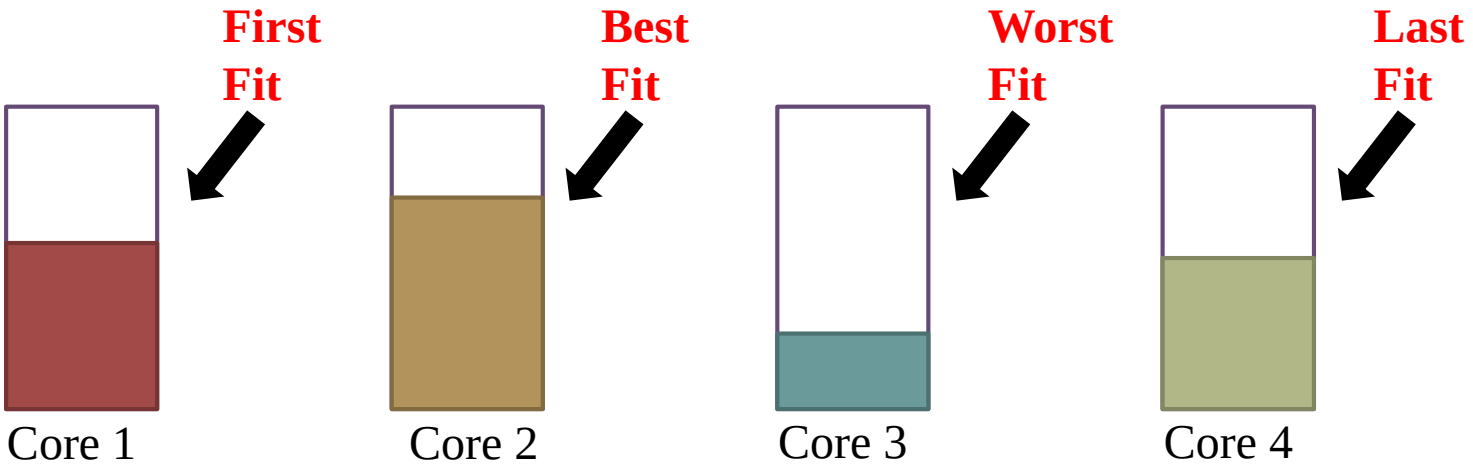
- Settings:
  - 2 cores
  - 2 tasks with execution time 10 arrive at time 0
  - 1 task with execution time 19 arrives at time 1

- Global Scheduling
  - Core 1: 
  - Core 2: 
- Partitioned Scheduling
  - Core 1: 
  - Core 2: 

Different Fitting Approaches for Partitioned Scheduling

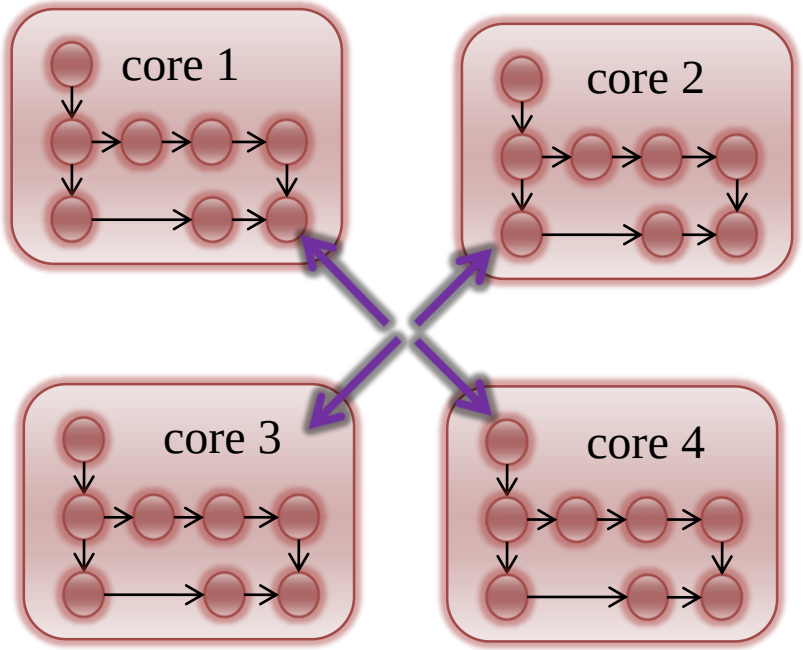
- First-Fit: choose the one with the smallest index
- Last-Fit: choose the one with the largest index
- Best-Fit: choose the one with the maximal utilization
- Worst-Fit: choose the one with the minimal utilization

Assigning a task with utilization equal to 0.1



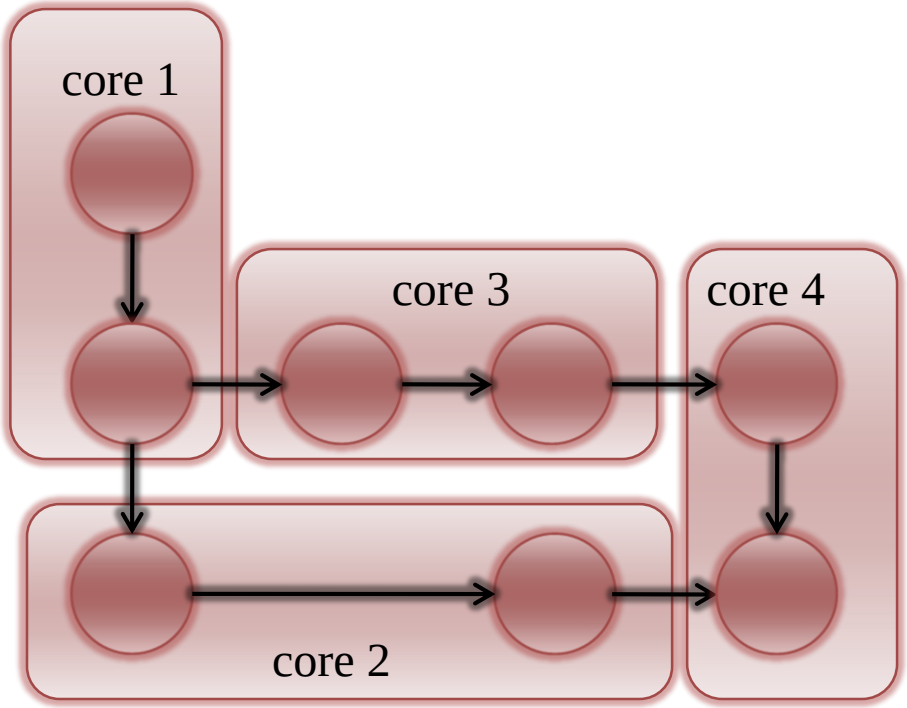
Two Approaches for Using Multiple Cores

- Data Partition



- Flexibility
- Scalability
- Load-balance

- Functional Partition

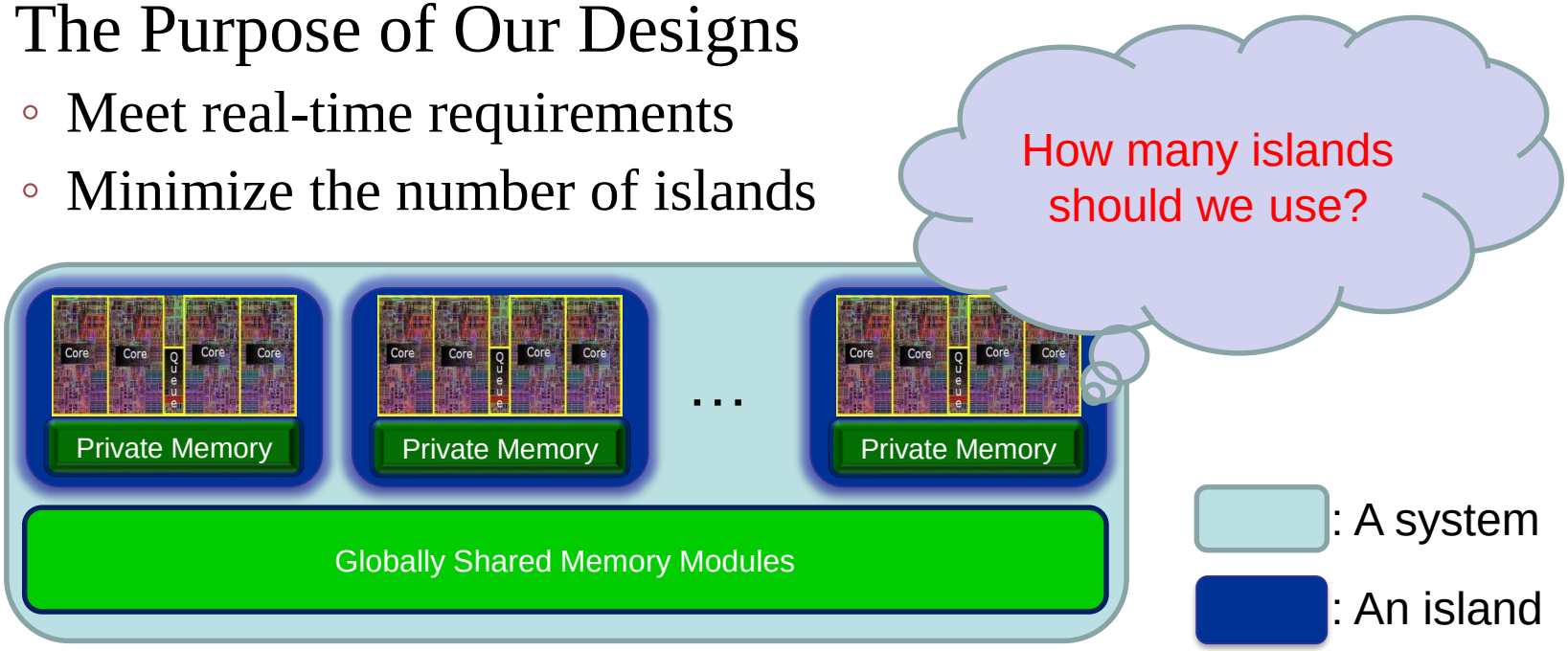


- Hardware/software co-design
- Parallelism in sequential program

Studies of Multi-Core Scheduling with Heterogeneous Memory

System Model

- An Island-Based Multi-Core Platform
  - A global memory pool and multiple islands
  - A policy to turn on/off islands to manage resources
- The Purpose of Our Designs
  - Meet real-time requirements
  - Minimize the number of islands

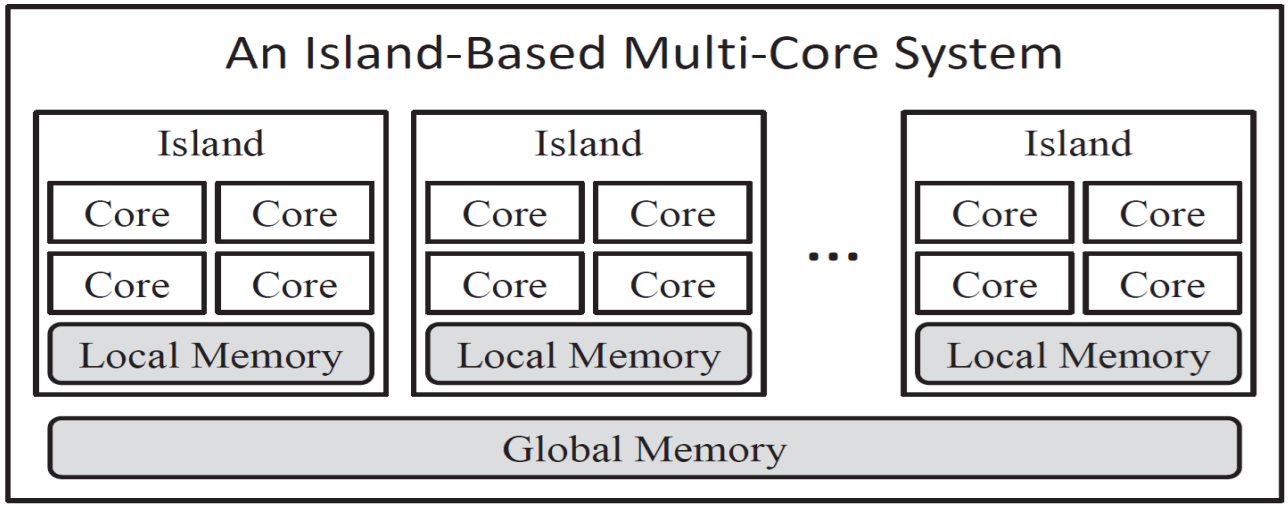


Problem Definition– Real-Time Tasks

- Goal:
  - Minimize the number of required islands
- Constraints:
  - All implicit-deadline sporadic tasks meet their deadlines
  - No memory space limitation is violated
- Task Model– Implicit-Deadline Sporadic Tasks:
  - A relative deadline
  - The minimum inter-arrival time
  - The worst case execution time

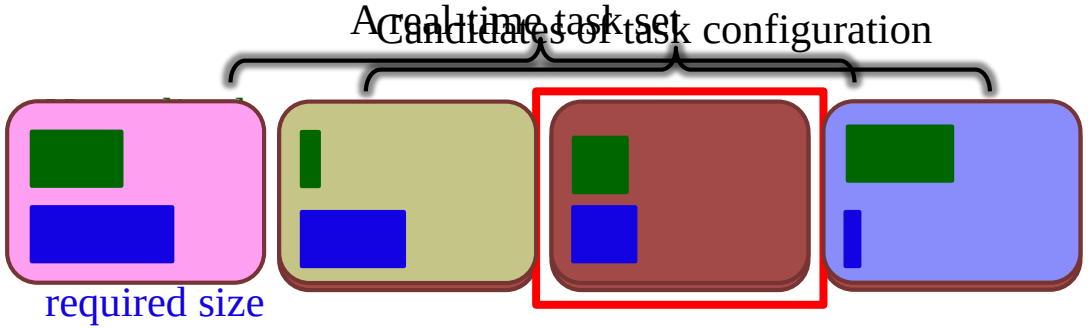
Problem Definition– Hardware Platforms

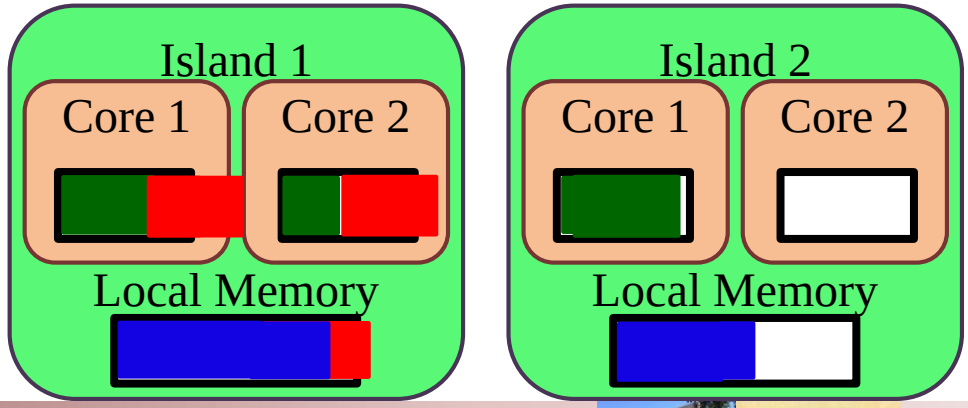
- Hardware Model– Island-Based Multi-Core Platforms:
  - Cores in an island share a local memory pool
  - An island consists of multiple homogeneous cores
  - A system consists of multiple islands
  - A system consists of a large global memory pool



Proposed Algorithms

1. Minimize the value of $\frac{U_i^j}{M} + \frac{j}{B_L}$ for each task
2. Sort all tasks in a non-increasing order by the required number of the local memory blocks
3. Partition all tasks to islands by the first fit strategy
4. Assign tasks onto cores by the first fit strategy
5. Allocate a new island if there is no feasible assignment for a task





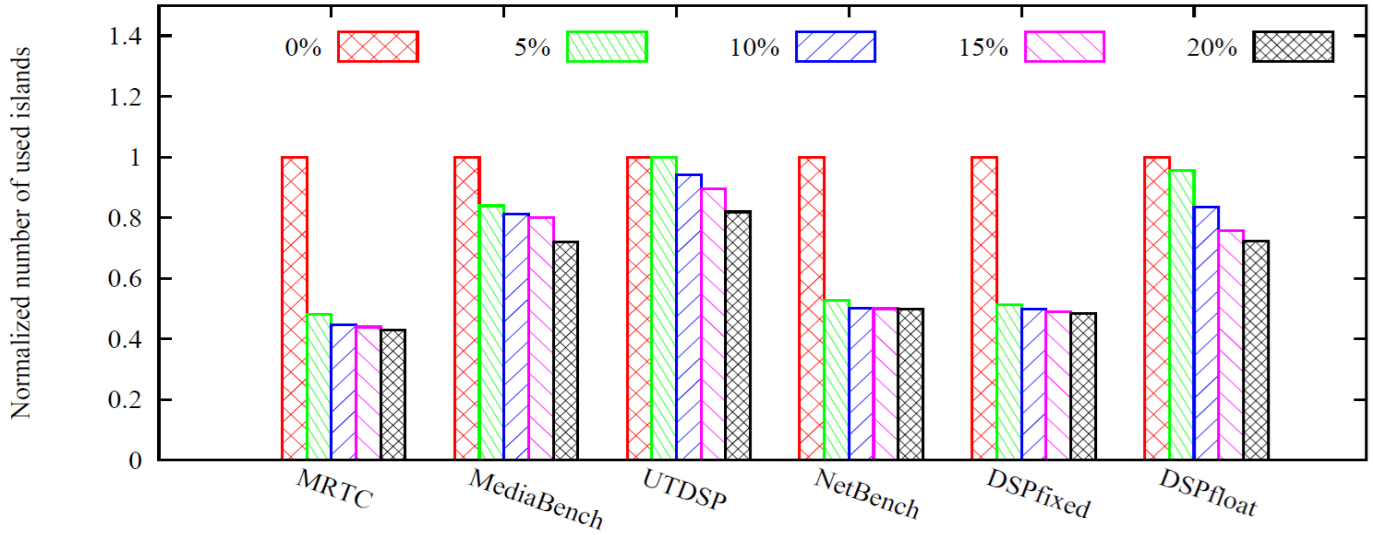
22

Analysis of the Algorithm

- ***Theorem 3***: the properties of the algorithm
  - The time complexity is $O(|\mathbf{T}|^2 + \gamma)$, where $|\mathbf{T}|$ is the number of task, and γ is the number of the candidates of memory allocation
  - The derived result (number of the required islands) is bound by $4OPT + 2$, where OPT is the result of the optimal solution
- The value of the proposed algorithm
  - Provide the lower bound for an IBRT problem instance
  - Provide a bounded solution for island-based multi-core system synthesis

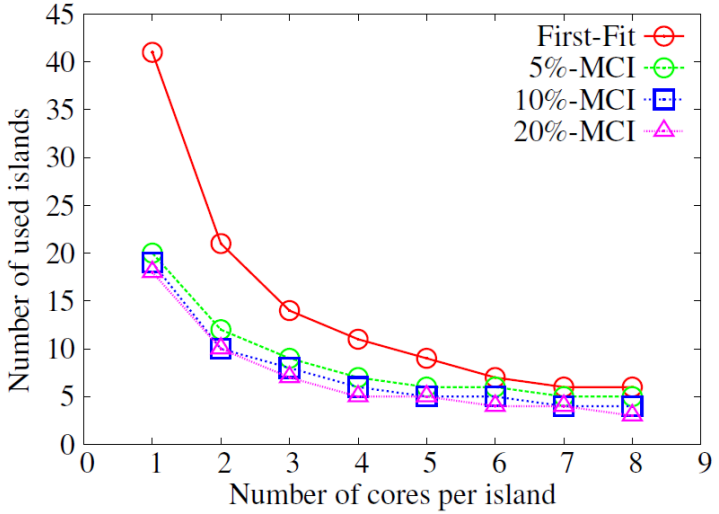
Performance Evaluation (1/2)

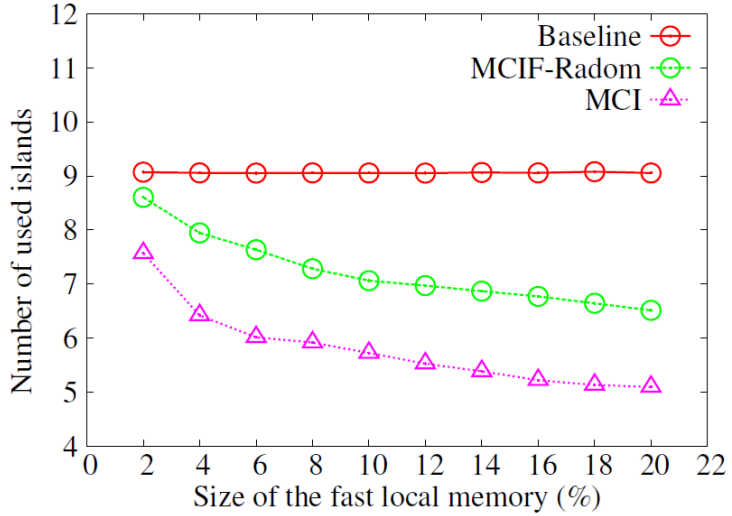
- 82 real-life benchmarks from MRTC, MediaBench, UPDSP, NetBench, and DSPstone
- The worst-case execution time of each task is generated by aiT which is based on Infineon TriCore TC1797
- The number of cores in an island is 4



Performance Evaluation (2/2)

- Include the 82 benchmarks for the two experiments
- Vary the number of cores per island for the left experimental results
- Vary the size of the local memory for the right experimental results

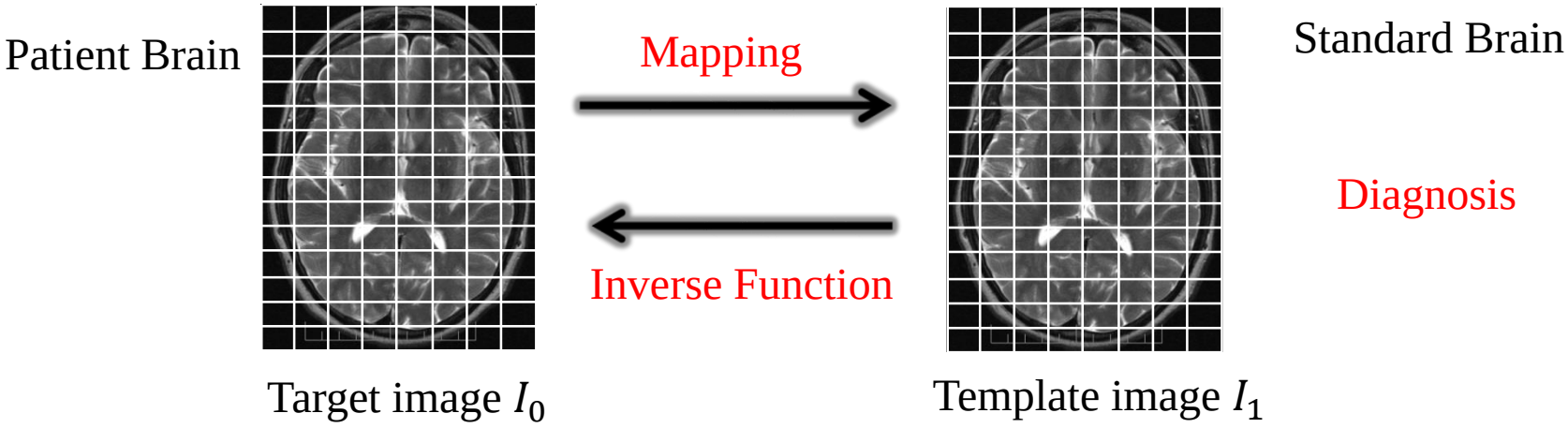




Studies of Heterogeneous Computing Resources

Image Registration

- LDDMM-DSI conducts image registration while preserving the topology conservation and invertibility
- However, the precision of the outcome is at the expense of the execution time
- Why we need image registration?

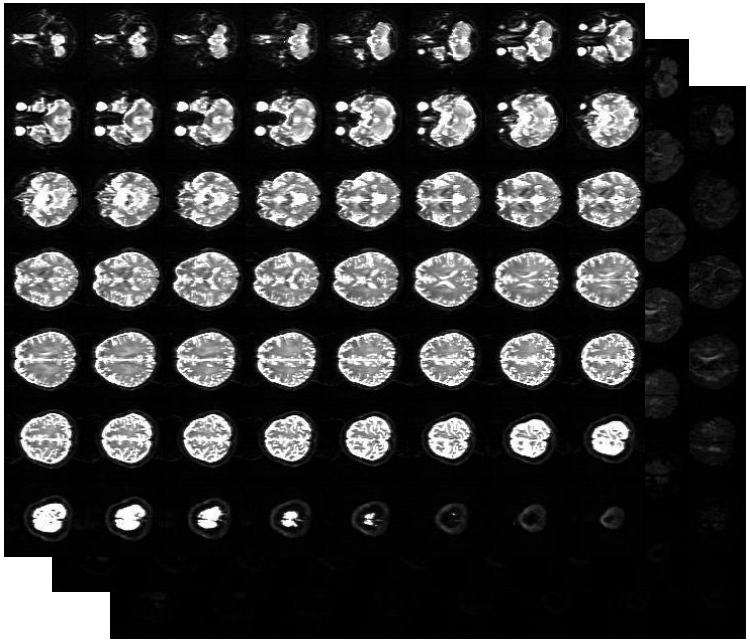


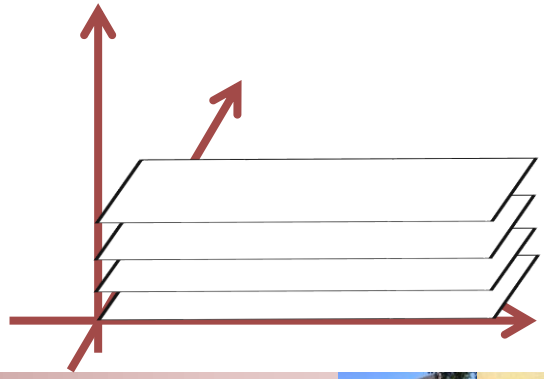
LDDMM-DSI

- LDDMM-DSI
  - A large deformation diffeomorphic metric mapping solution for diffusion spectrum imaging datasets
- Motivation
  - Medical Image Center
    - Huge amount of medical images are received for processing
    - How to exploit the throughput becomes important
  - Instant checking of many different possible diseases
    - Different weighted MRI images target different soft tissues
    - Thus, different registration processes will be required to conduct thorough checking

Input/Output Data

- The input data contain both i-space (3D) and q-space (3D)
- The output data contain mainly the velocity files, which is around 2 GB in total
- The intermediate data are larger than 30 GB

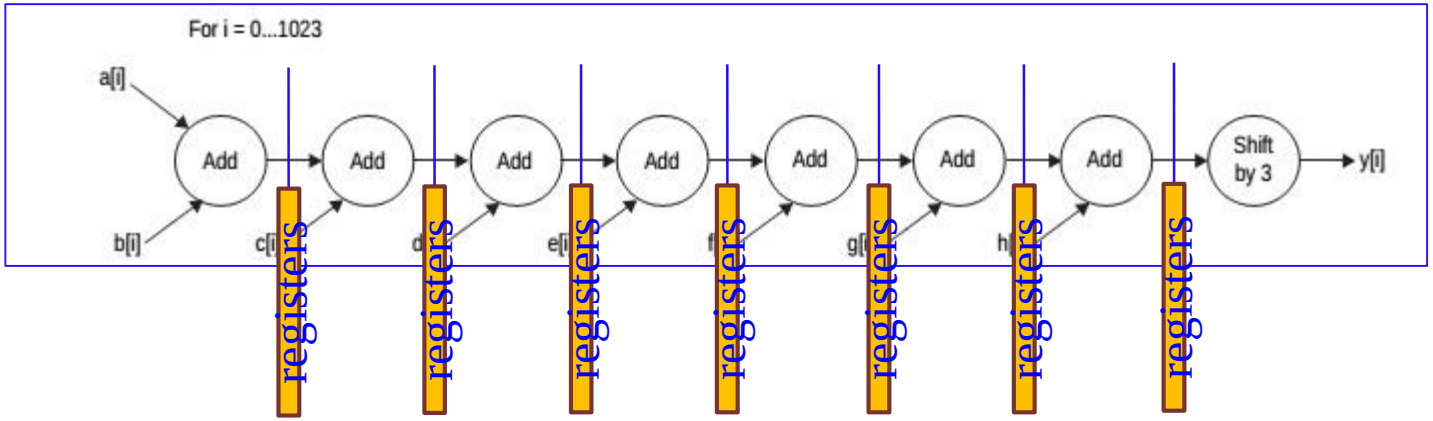




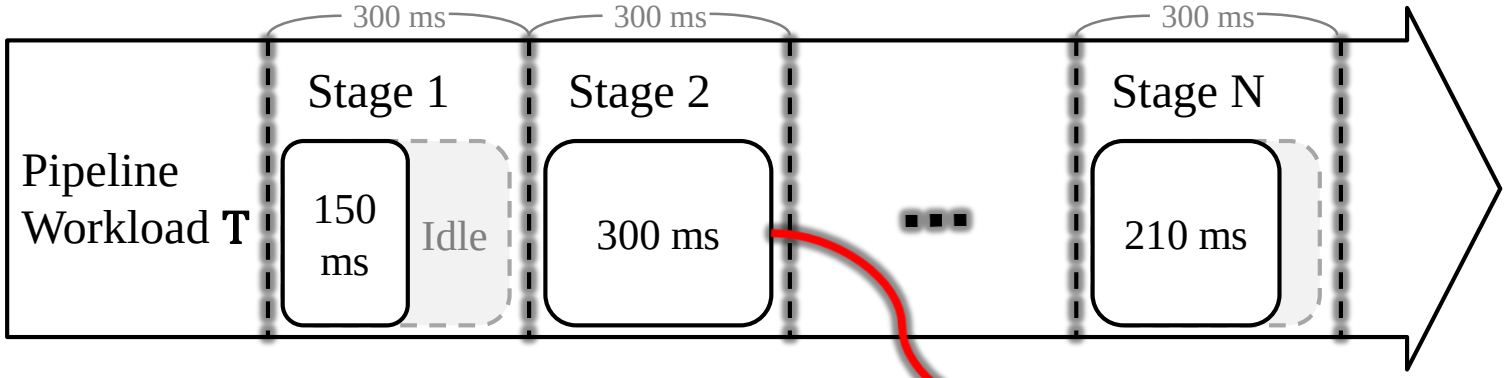
Pipeline Designs of FPGA

- Acceleration Techniques on FPGA
  - Customized Datapath
  - Pipelined Execution

```
for(i = 0; i < 1024; i++)
{
   y[i] = (a[i] + b[i] + c[i] + d[i] + e[i]+ f[i] + g[i] + h[i]) >> 3;
}
```



Challenges

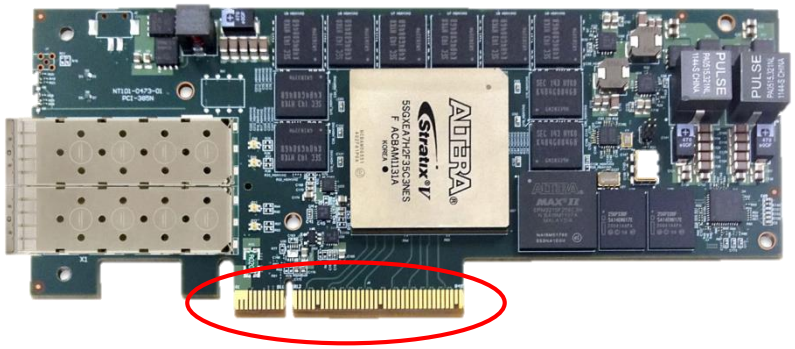


- Since throughput = $\frac{1}{\text{longest stage time}}$

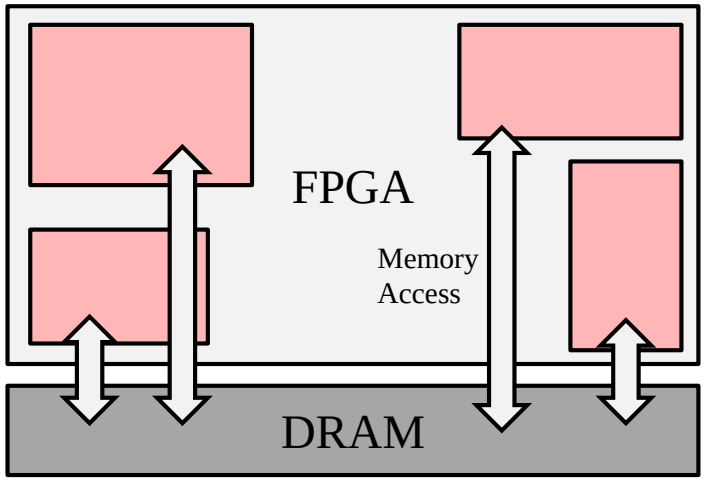
- To achieve optimal performance:
  - Minimize idle time
  - Trade **area** and **bandwidth** for performance
    - Area and bandwidth are limited resource!

FPGA Device Model

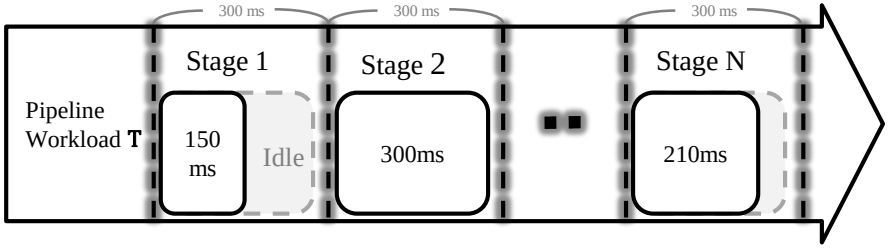
- Example: Nallatech PCIE 385n (Altera Stratix V A7 FPGA)
  - Logic element count: 622000
  - Memory bandwidth: 25 GB/s



- Model:
  - A: total area (gate count)
  - B: total memory bandwidth



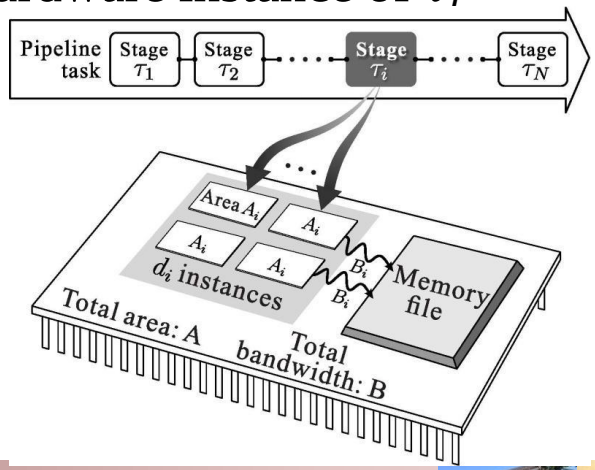
Task Model



- A **pipeline** workload **T** has N stages of execution
  - The i-th stage τ_i has a degree D_i of ***data parallelism***
- When a **hardware instance** of τ_i is programmed on FPGA
  - Consumes FPGA ***area***: A_i
  - Requires ***memory bandwidth***: B_i
  - Requires time T_i to execute a single work-item
- Let d_i be the **number of** programmed hardware instance of τ_i
  - Required area: $A_i d_i$
  - Required bandwidth: $B_i d_i$
  - Execution time: $T_i \left\lceil \frac{D_i}{d_i} \right\rceil$

Time to process a batch

Number of batches



Problem Definition

- METM (**M**aximum **E**xecution **T**ime **M**inimization problem)
- Problem definition:

Minimize

$$\max_{\forall \tau_i \in \mathbf{T}} \left\{ \Gamma_i \left\lceil \frac{D_i}{d_i} \right\rceil \right\}$$

Stage execution time

Pipeline stage time

subject to

$$\sum_{\forall \tau_i \in \mathbf{T}} A_i \cdot d_i \leq A$$

Area constraint

$$\sum_{\forall \tau_i \in \mathbf{T}} B_i \cdot d_i \leq B$$

Bandwidth constraint

$$1 \leq d_i \leq D_i, \forall \tau_i \in \mathbf{T}$$

Dynamic Programming Solution

- DP-based algorithm: DP-METM (Maximum Execution Time Minimization)

min time of first x-1 stages

exec time of x-th stage

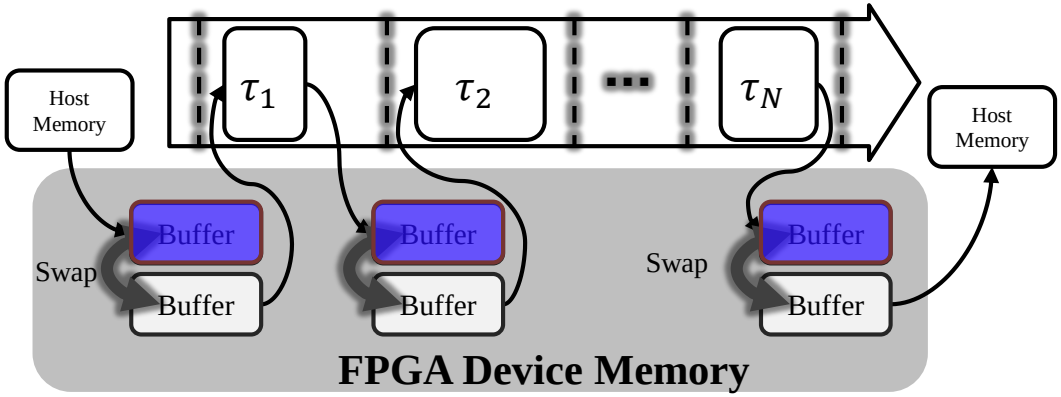
$$v(x,a,b) = \begin{cases} \min\limits_{1 \le d_x \le D_x} \left\{ \max\left\{ v(x-1, a', b'), T_x \left\lceil \frac{D_x}{d_x} \right\rceil \right\} \right\}, \text{if } \begin{cases} 0 \le a \le A \\ 0 \le b \le B \end{cases} \\ \infty \quad , \text{otherwise} \end{cases}$$

Where $a' = a - A_x \cdot d_x$, $b' = b - B_x \cdot d_x$ and $x \geq 1$

- Algorithm DP-METM is optimal
- Complexity of DP-METM is: $O(|\mathbf{T}| \cdot A^2 \cdot B)$

Implementation Remarks

- Input parameter measurement
  - A_i, B_i can be obtain from the compiler output
  - T_i need to be measured manually
- Runtime library support
  - Dispatch stage workload
  - Exchange data between stages



- Example platform: Nallatech PCIE 385n [1] FPGA card
  - Using OpenCL toolchain

[1] N. Coporation, "Nallatech 385 Product Brief," Feb 2014.