System-Level Power-Performance Trade-offs in Task Scheduling for Dynamically Reconfigurable Architectures

Juanjo Noguera
InkJet Commercial Division (ICD)
Hewlett-Packard Company
jnoguera@bpo.hp.com

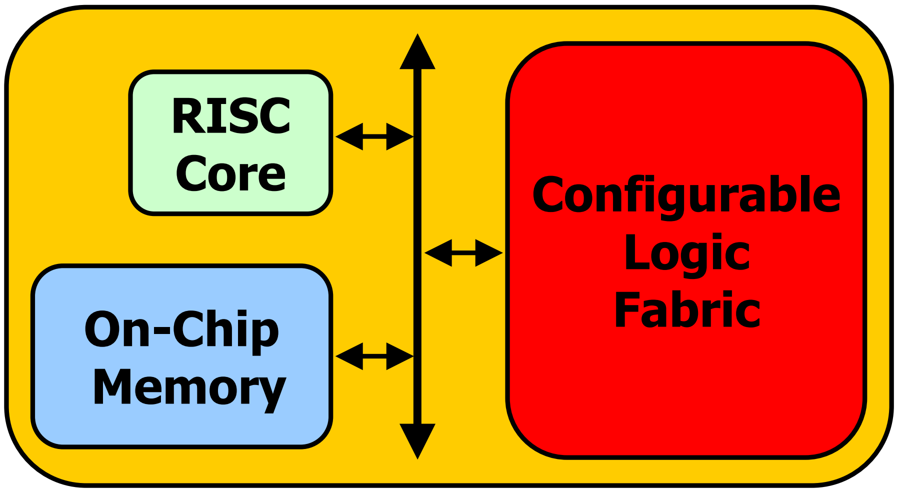
Rosa M. Badia
Computer Architecture Dept.
Technical University of Catalonia (UPC)
rosab@ac.upc.es

Outline

- Introduction & Motivation
- Design Methodology for Embedded Systems
- Dynamically Reconfigurable Architecture
- Dynamic Task Scheduling Approach
  - Used Strategies for Power Minimization
- Experiments & Results
- Conclusions

Introduction & Motivation

- **Configurable/Programmable System-On-Chip is Becoming a Reality**
  - **Xilinx Virtex II Pro (with an embedded PowerPC)**



- **Low-Power / Energy-Efficiency is a Challenge for Embedded Systems Design**

Previous Work

- **Reconfiguration Latency is a Challenge in Reconfigurable Computing**
  - **Configuration Pre-fetching (overlaps reconfiguration with computation)**
- **Scheduling for Reconfigurable Architectures is based on Static Scheduling Techniques**
  - **Recent Work in Operating Systems Support**
  - **OS consumes a big amount of power of an embedded system**
- **Research Efforts on:**
  - **Power Consumption Distribution in FPGA devices**
  - **Moving into configurable hardware parts of software to reduce power consumption**

Objective of this Work

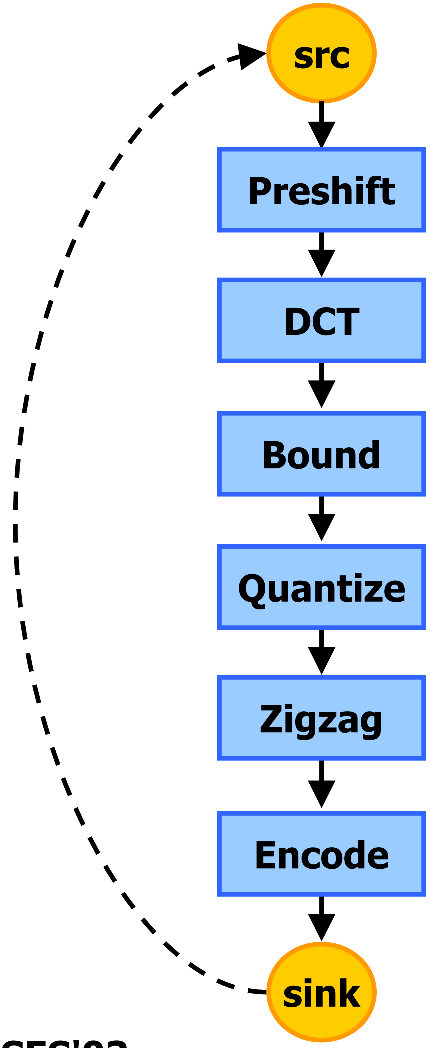
Explore Power-Performance Trade-Offs in Dynamic Task Scheduling Techniques for DRL Architectures

- Configuration Reuse & Configuration Pre-fetching
- Clock-gating & Frequency Scaling
- Hardware Support for Configuration Loading & No OS Support for Dynamic Scheduling

Outline

- Introduction & Motivation
- Design Methodology for Embedded Systems
  - Application Specification
  - Design Methodology Overview
- Dynamically Reconfigurable Architecture
- Dynamic Task Scheduling Approach
- Experiments & Results
- Conclusions

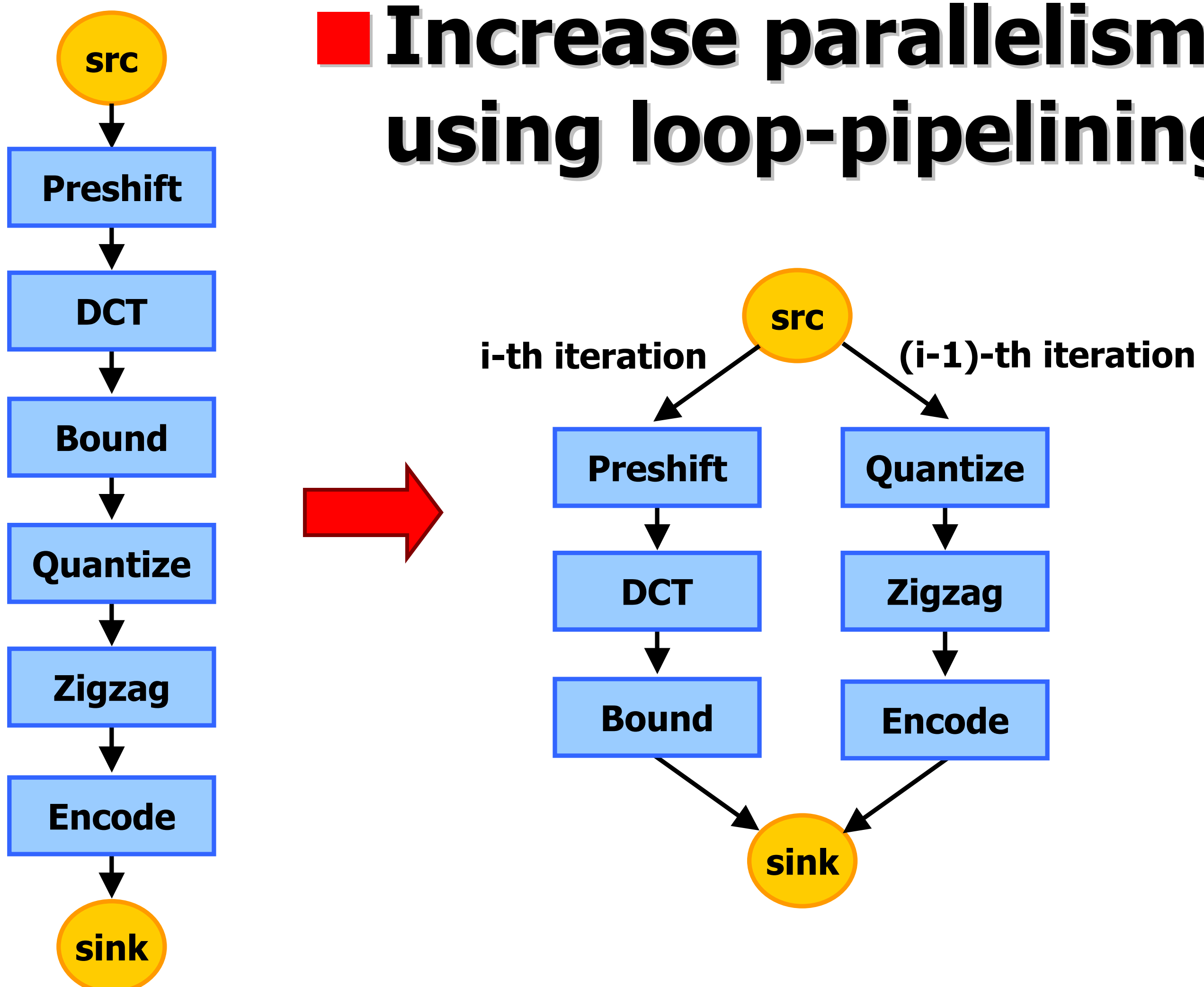
Application Specification



- Data-Flow Graph
- Coarse-grain tasks
- Data dependencies between tasks
- Data stream based processing
- Ex: JPEG enc

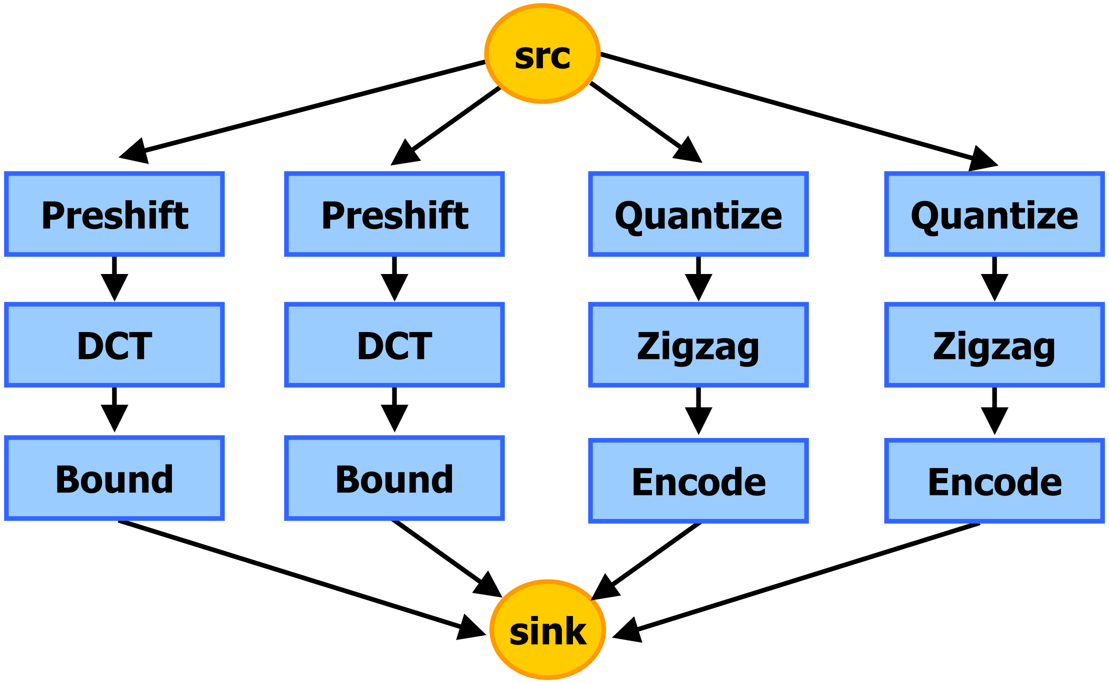
Application Specification (II)

- Increase parallelism using loop-pipelining



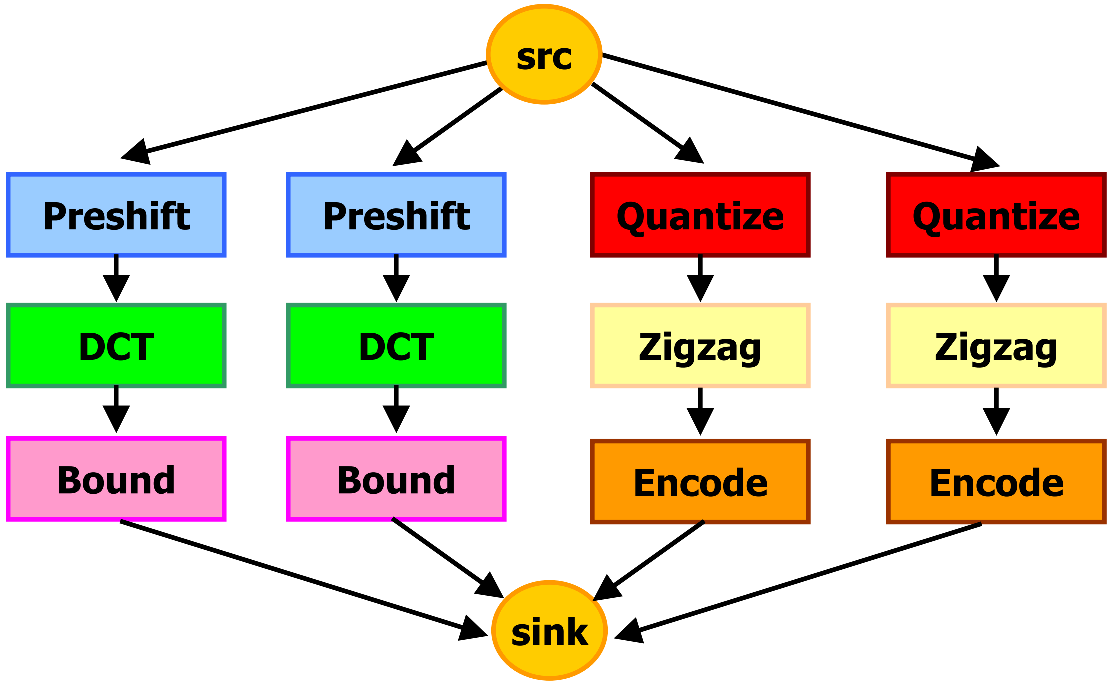
Application Specification (III)

- … or further parallelism performing loop-unrolling

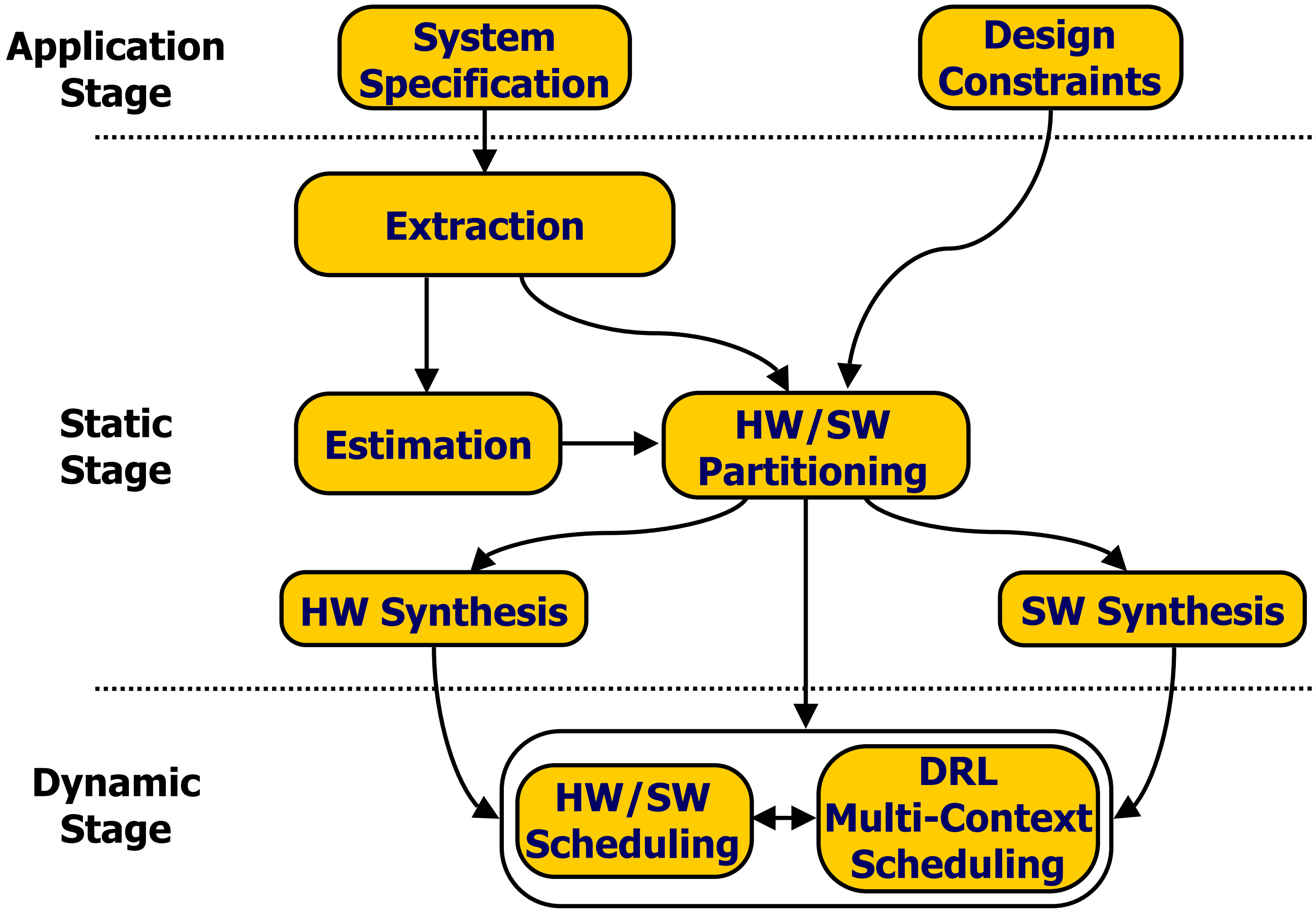


Application Specification (IV)

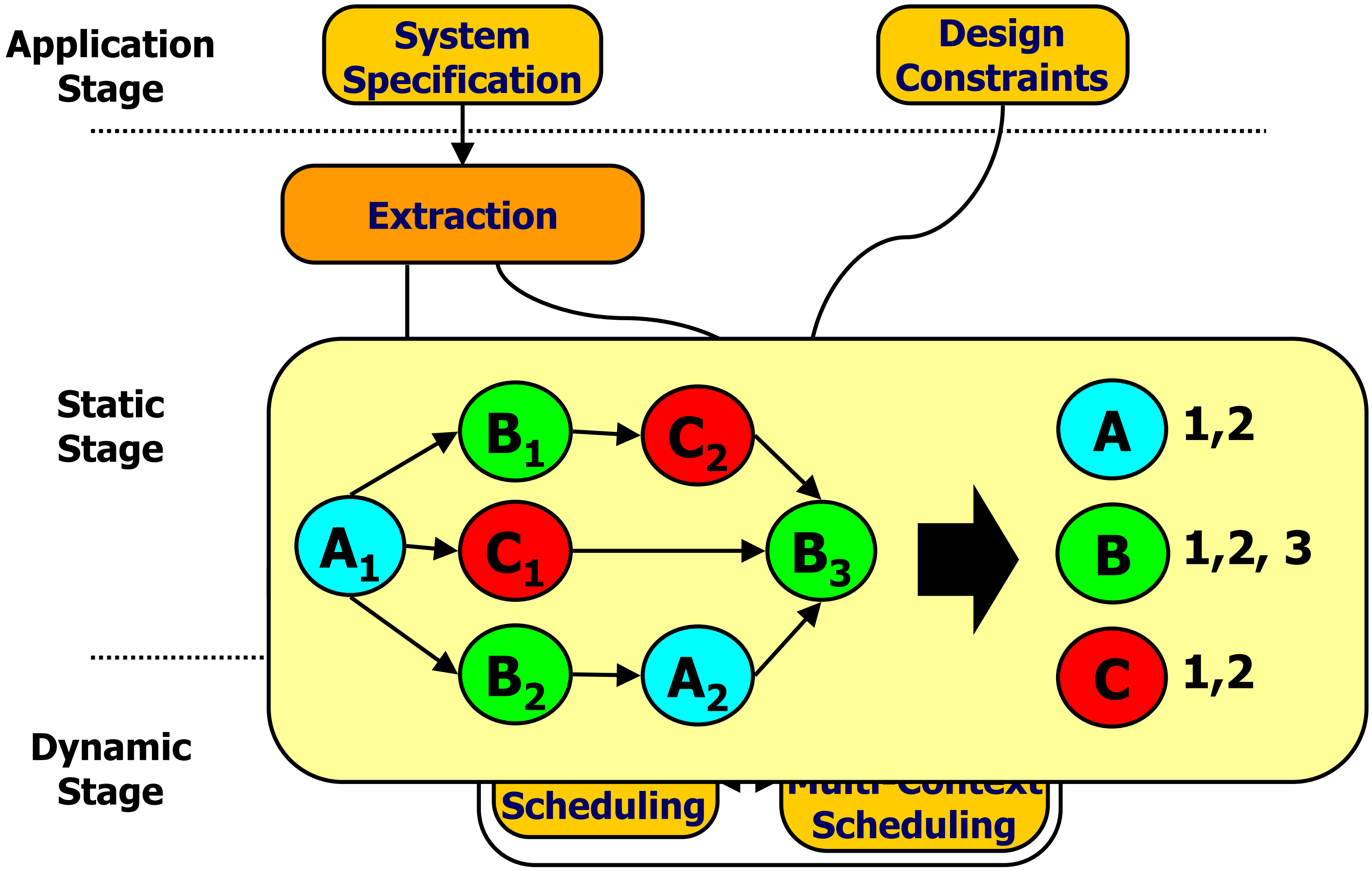
- **Modeling of applications:**
  - Acyclic Task Graph (Data-Flow Graph)
  - Task & Task Types



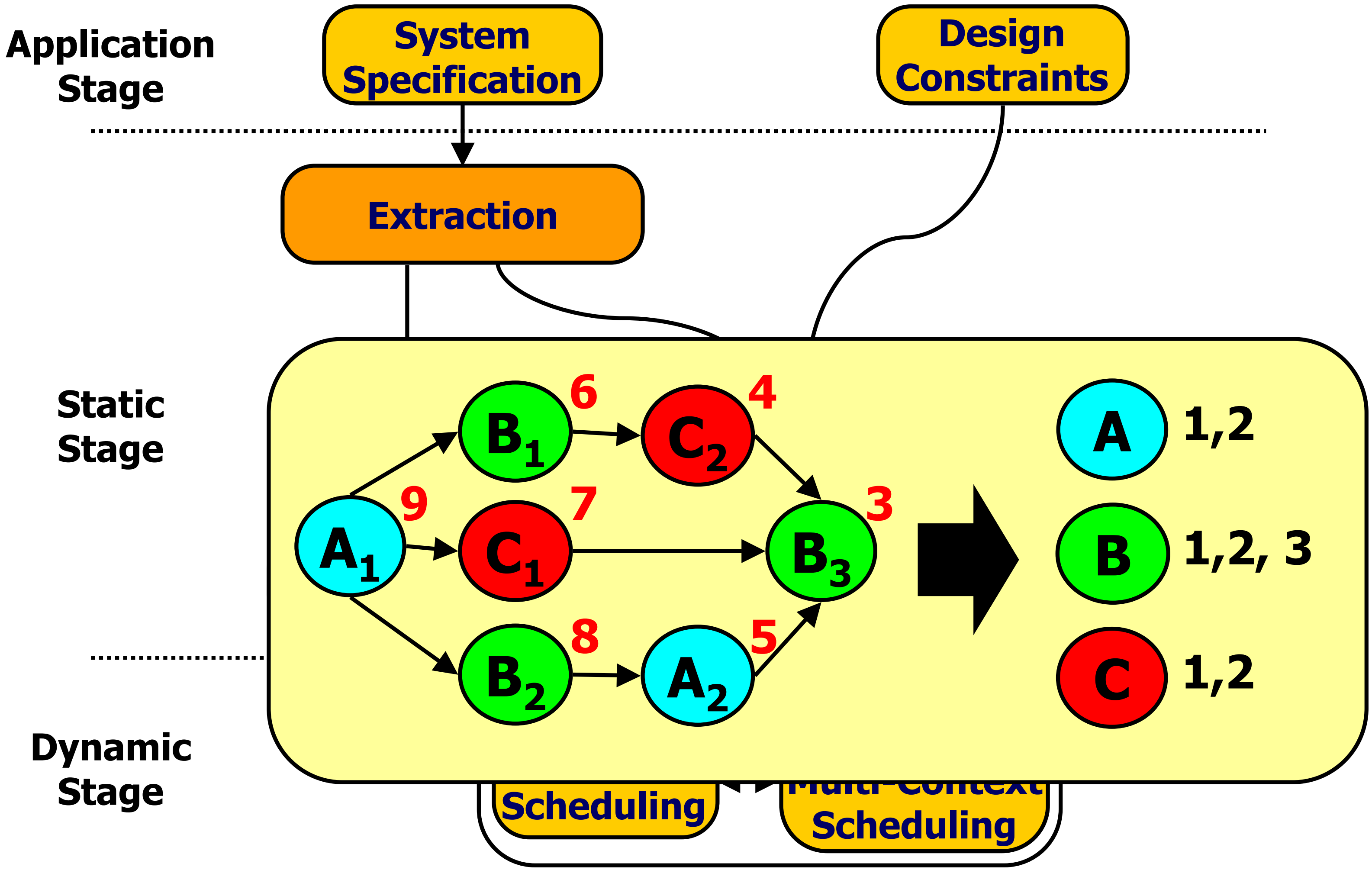
Design Methodology Overview



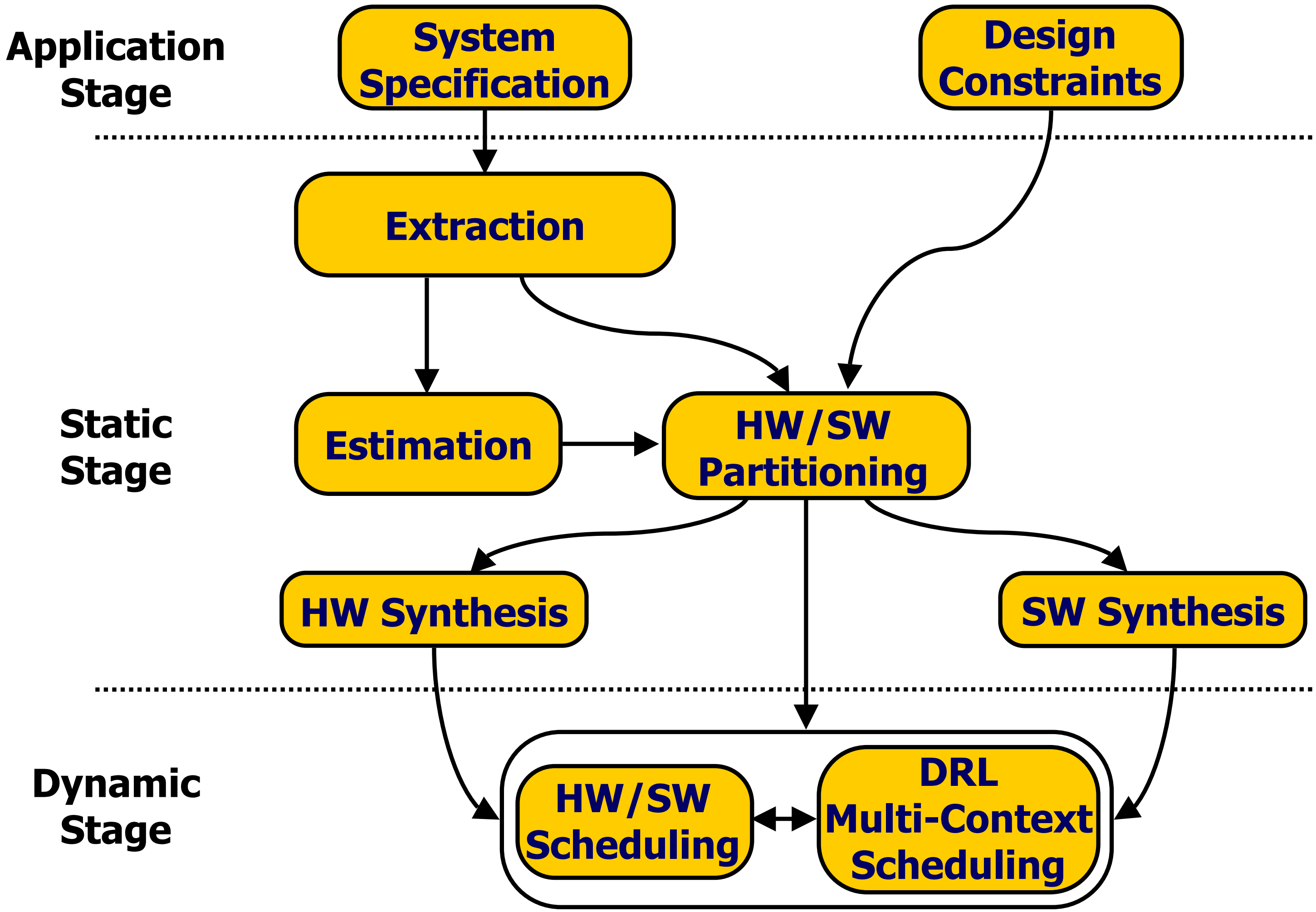
Design Methodology Overview



Design Methodology Overview



Design Methodology Overview

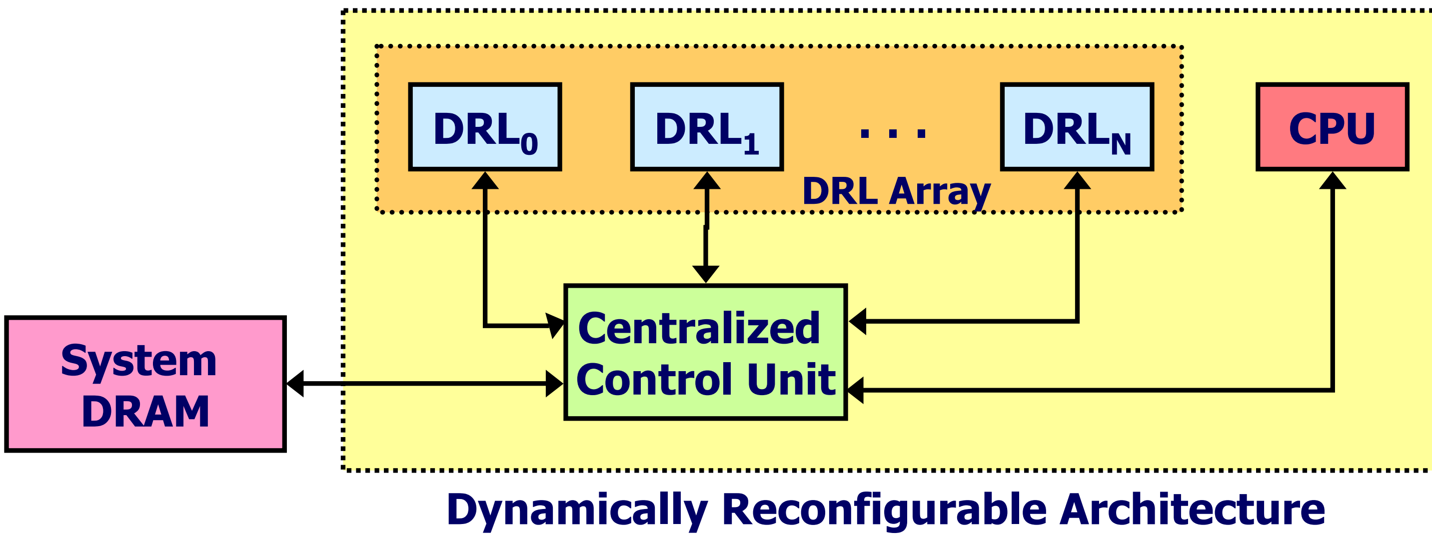


Outline

- Introduction & Motivation
- Design Methodology for Embedded Systems
- Dynamically Reconfigurable Architecture
  - General Overview
  - Galapagos: A First Prototype
  - Central Control Unit Architecture
  - DRL Controller Architecture
- Dynamic Task Scheduling Approach
- Experiments & Results
- Conclusions

General Overview

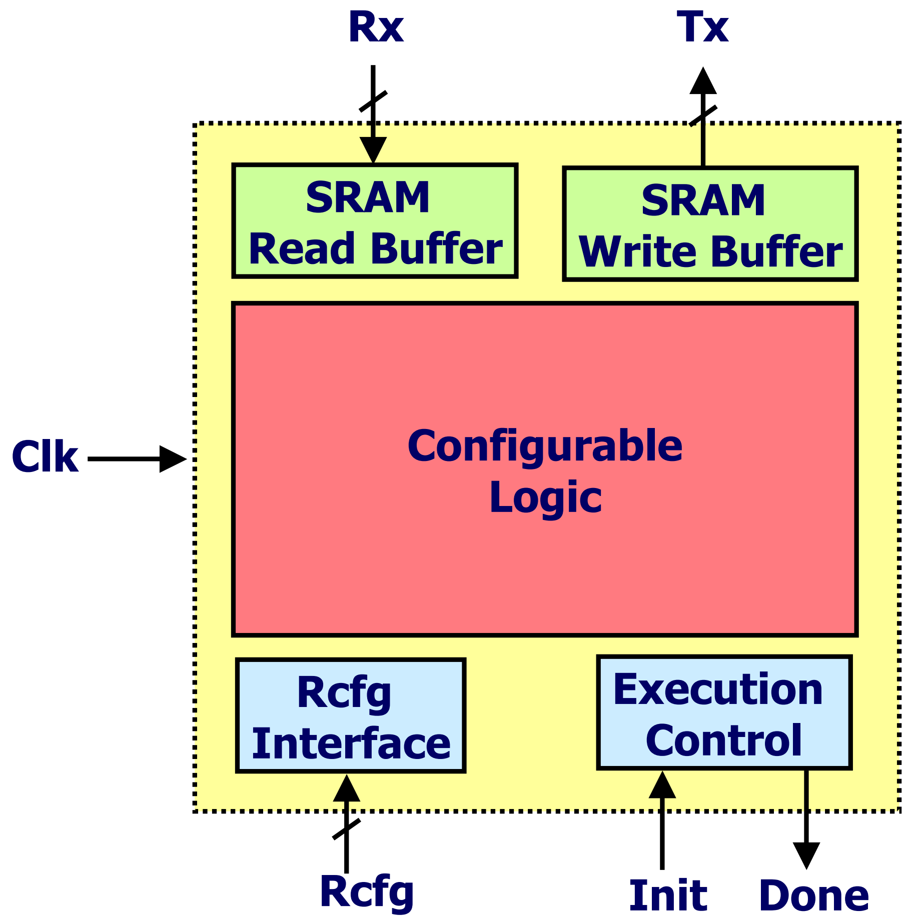
- **Centralized Control Scheme**
  - Dynamic Scheduling Algorithms Implementation
  - High-Bandwidth External DRAM Interface



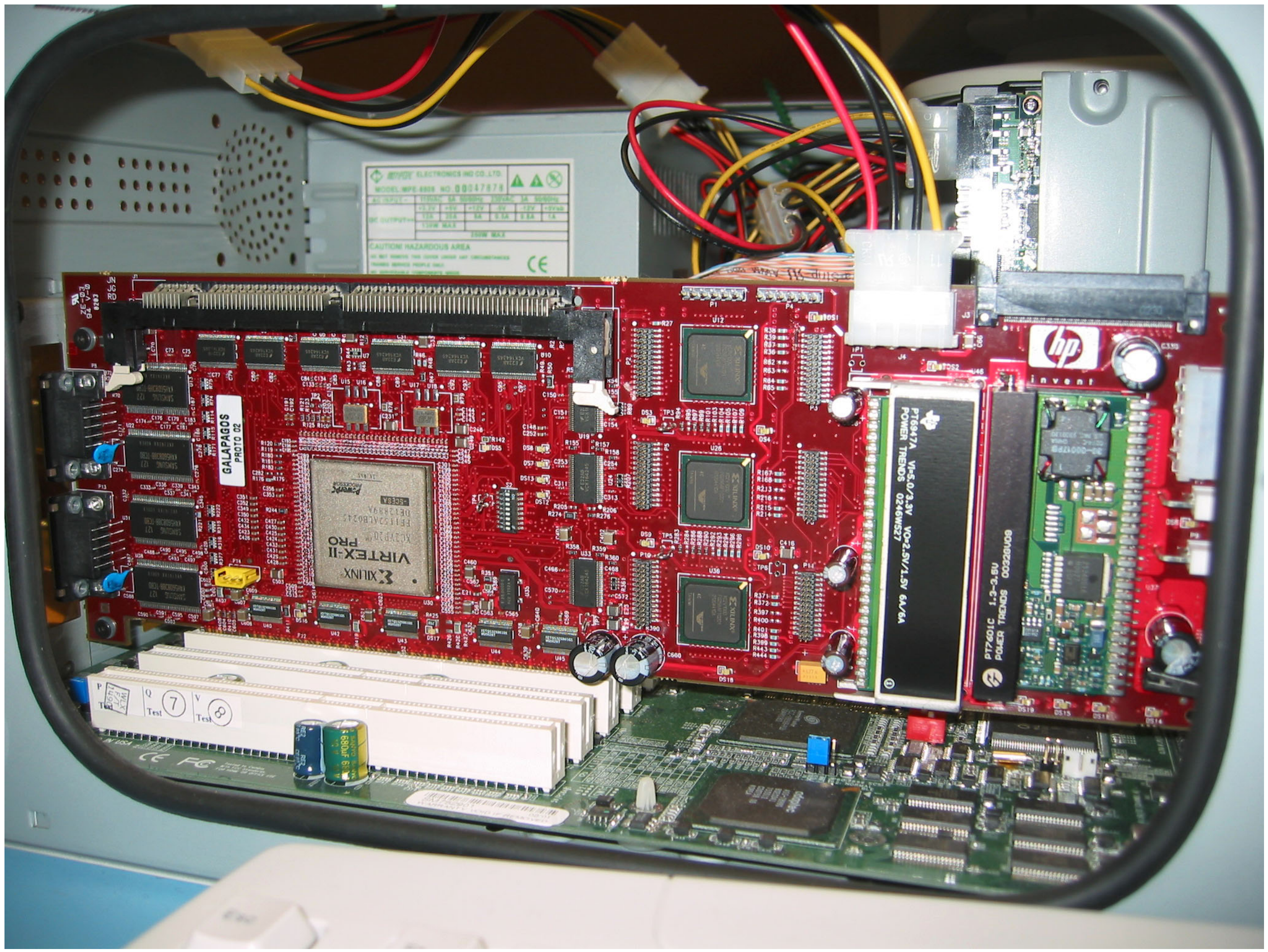
Dynamically Reconfigurable Architecture

General Overview (II)

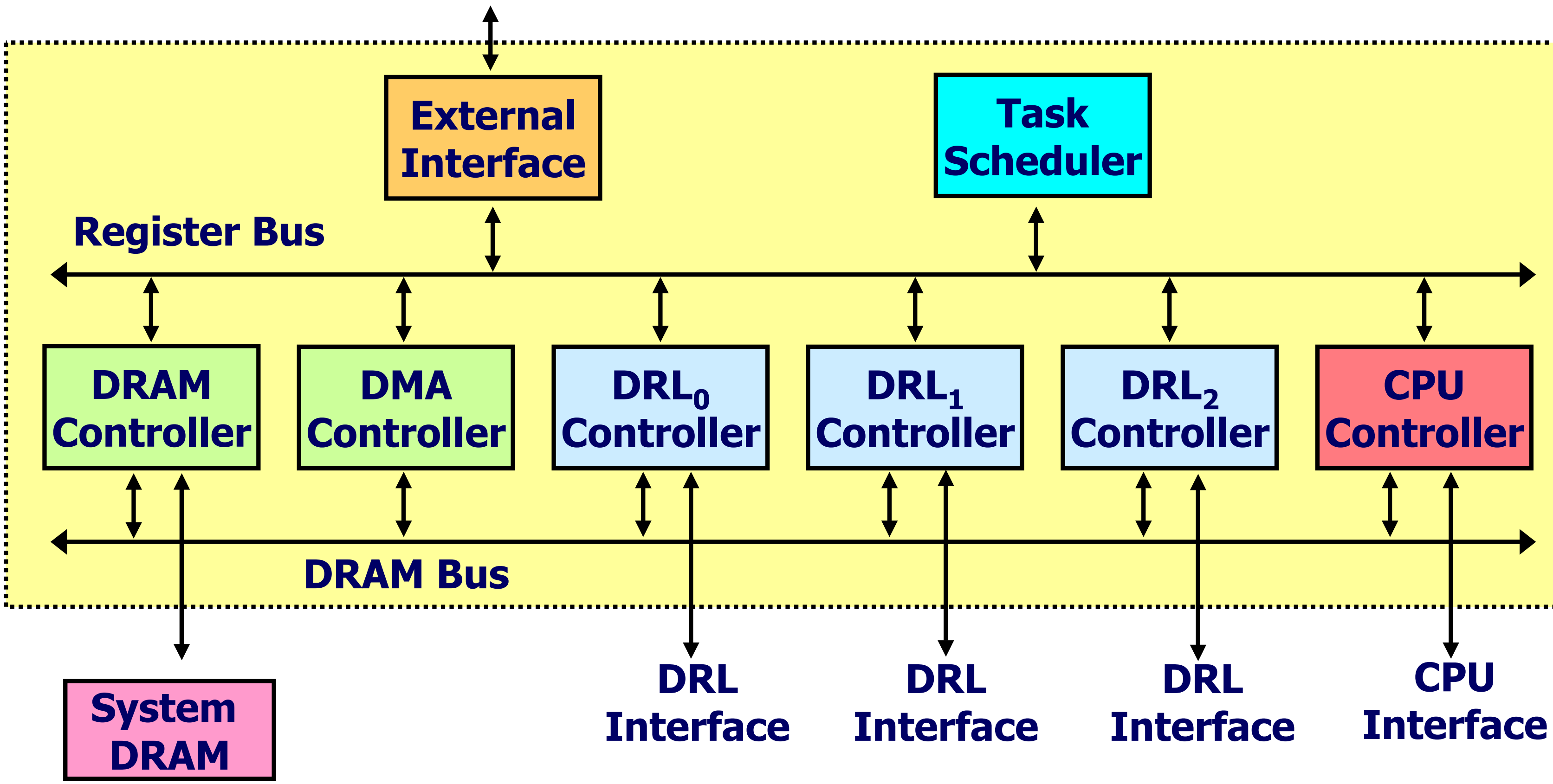
- **High-Bandwidth for Communication**



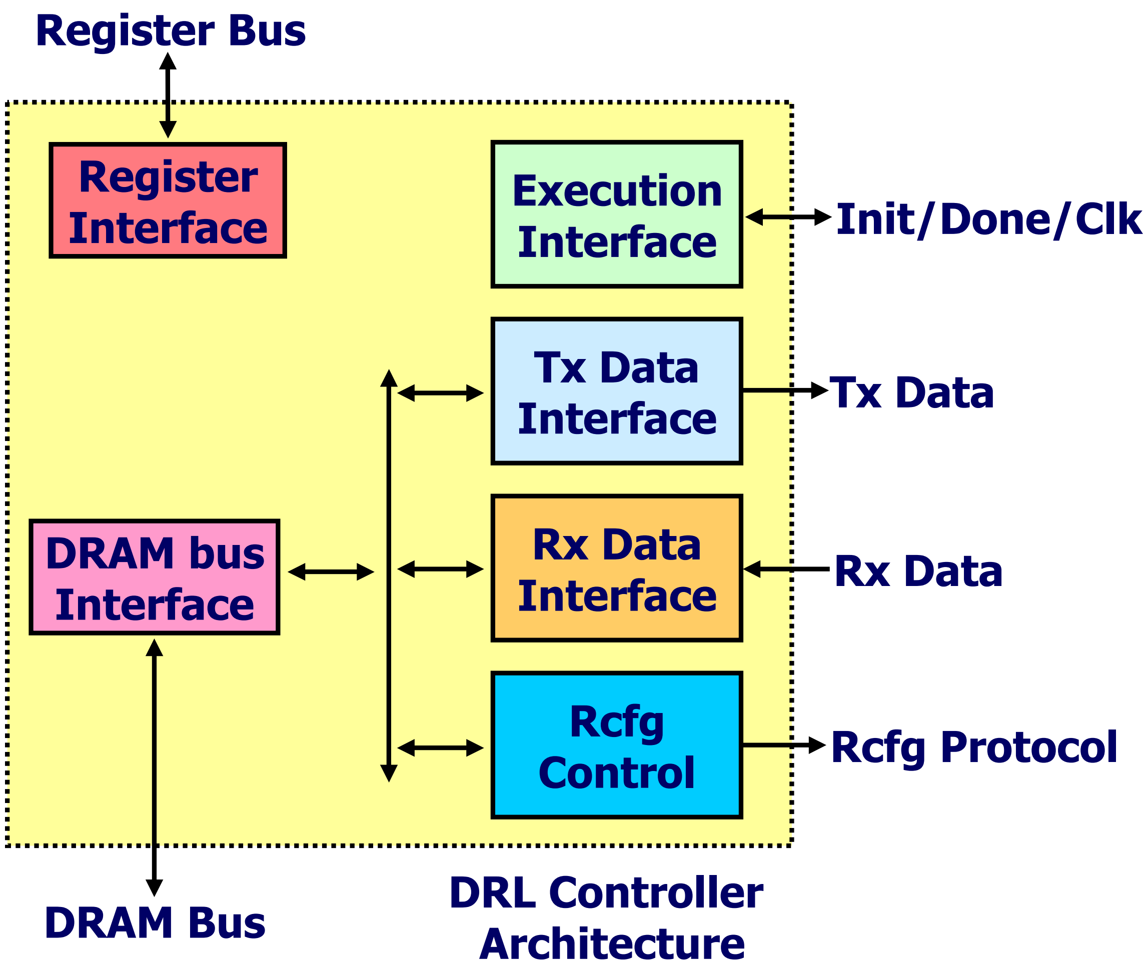
Galapagos: A First Prototype



Central Control Unit Architecture



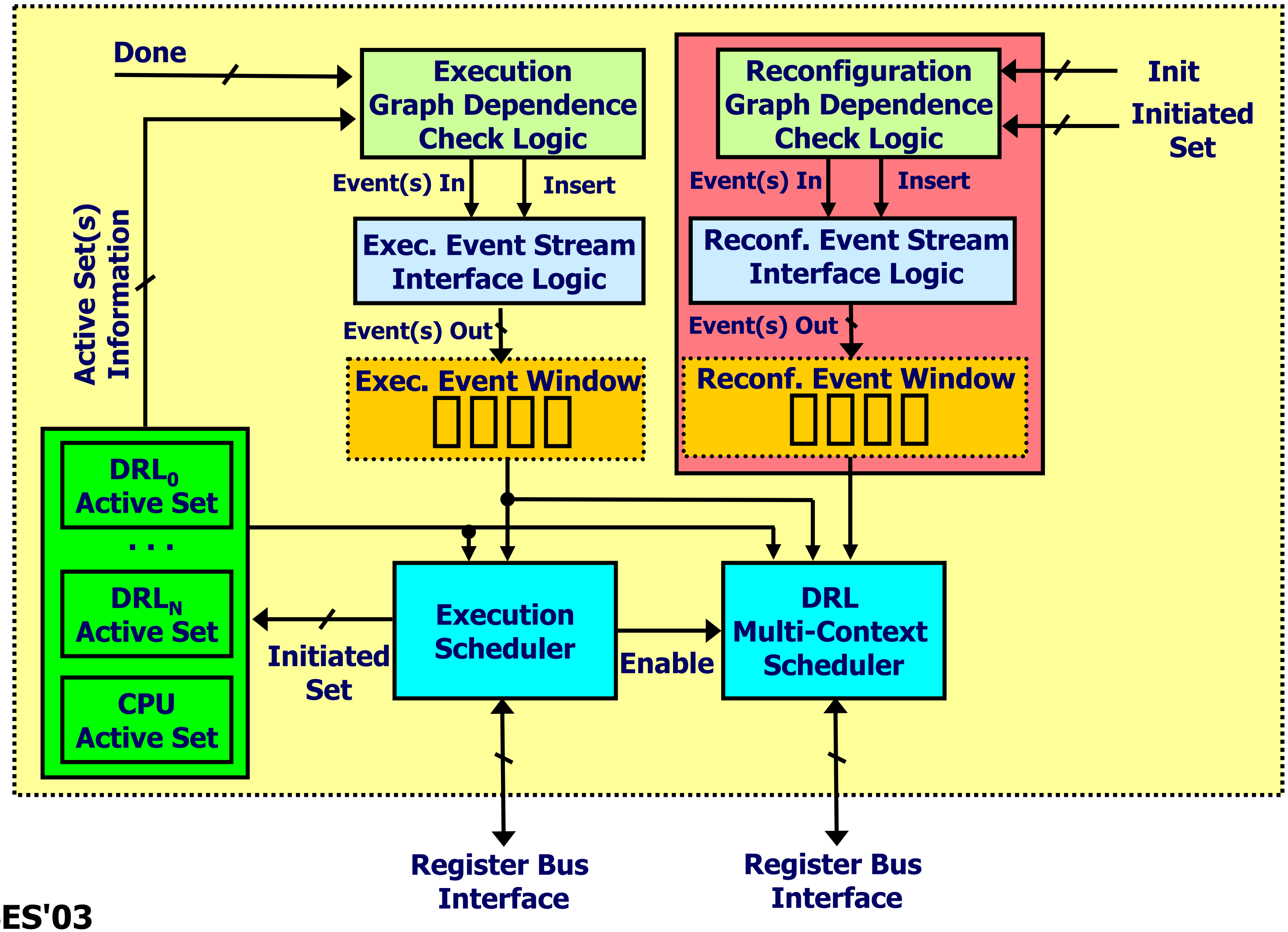
DRL Controller Architecture



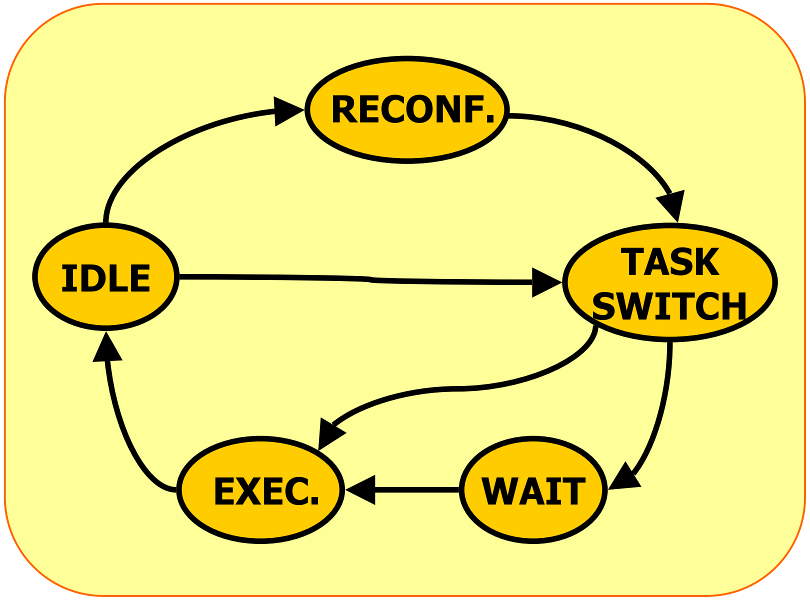
Outline

- Introduction & Motivation
- Design Methodology for Embedded Systems
- Dynamically Reconfigurable Architecture
- **Dynamic Task Scheduling Approach**
  - Task Scheduler Architecture
  - Support for Clock Gating and Frequency Scaling
  - DRL/CPU States
  - Scheduling Algorithms
- Experiments & Results
- Conclusions

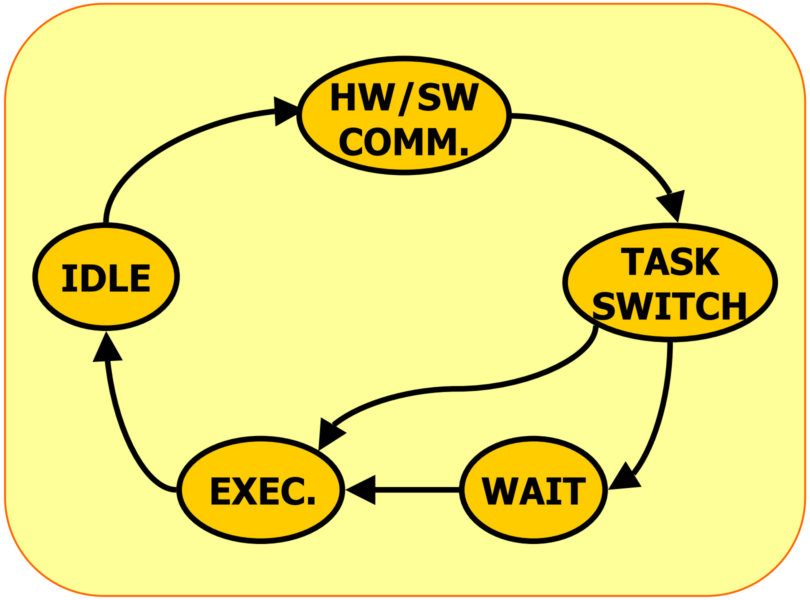
Task Scheduler Architecture



DRL/CPU States

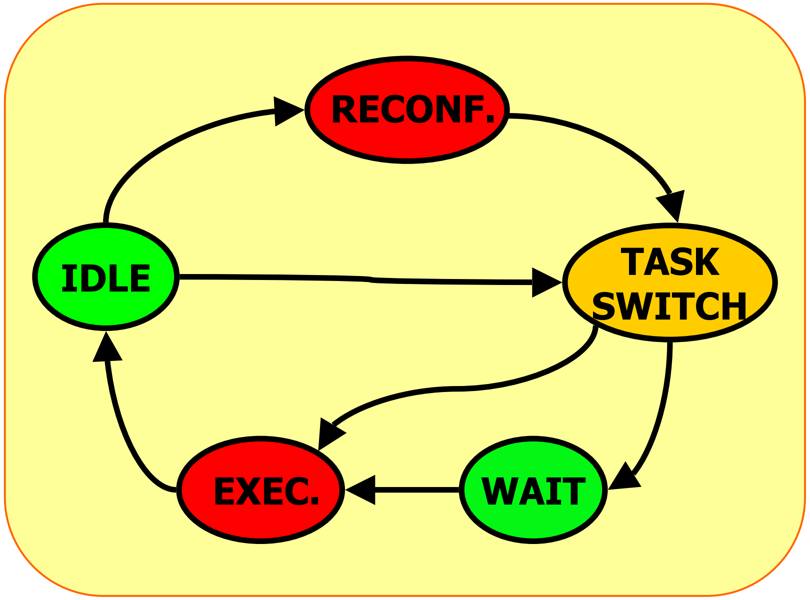


DRL States



CPU States

DRL/CPU States

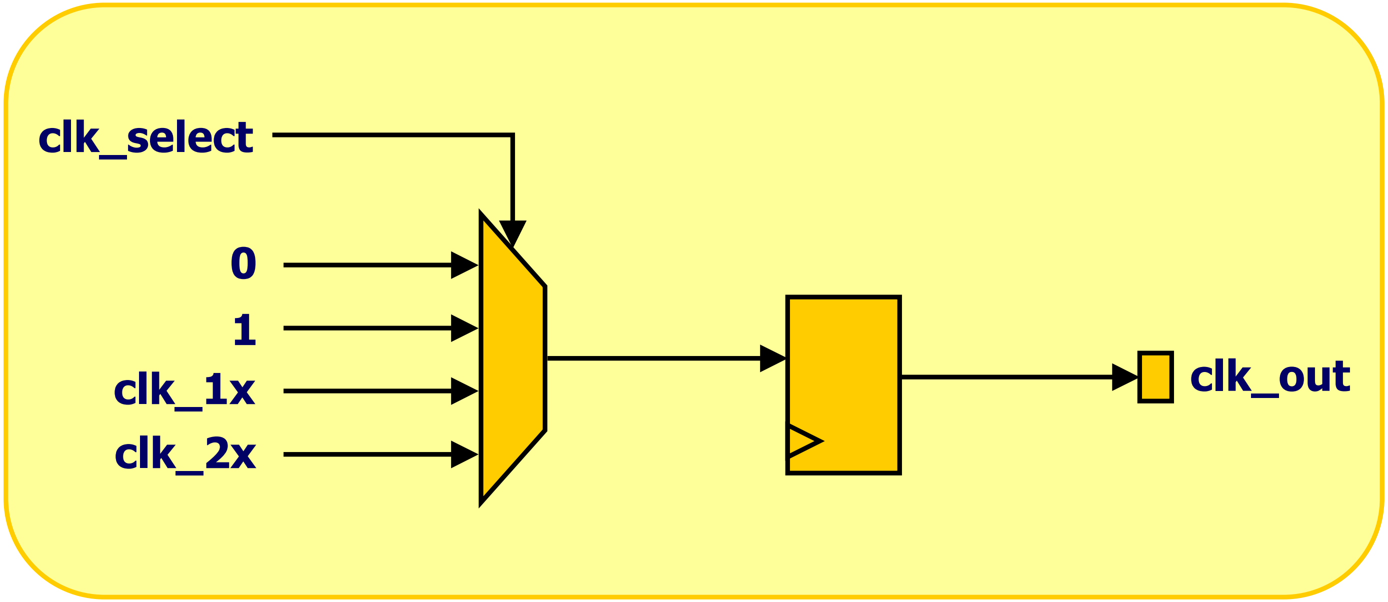


DRL States

- Reconfiguration is a high-power consumption state
- Power consumption of Execution state is minimized using frequency scaling
- Clock-gating is used in Idle and Wait states

Low-Power Strategies Support

- **Used Low-Power Strategies:**
  - **Clock-Gating (IDLE, WAIT)**
  - **Frequency-Scaling (EXECUTION)**



Scheduling Algorithms

- **Configuration Reuse as Selection Criteria. Select:**
  - **There is an Active Reconfiguration Context within the DRL Array ready for Execution**
  - **Within the Execution Event Window select the event with the Highest Priority**
- **DRL Multi-Context Scheduler based on a Configuration Pre-Fetching approach**
  - **Within the Reconfiguration Event Window select the event with the Highest Priority**

Outline

- Introduction & Motivation
- Design Methodology for Embedded Systems
- Dynamically Reconfigurable Architecture
- Dynamic Task Scheduling Approach
- **Experiments & Results**
  - **Implementation Results**
  - **Experiments Set-up**
  - **Power-Performance Results**
- Conclusions

Implementation Results

- The proposed architectures have been implemented on a Virtex-II Pro device
- Reconfigurable Control Module:
  - Used Hardware Resources
    - 634 FF's + 835 FF's + 2 SRAM Blocks
  - Power Consumption
    - Hardware = 55mW @ 66 MHz
    - Software (PowerPC) = 267mW @ 266MHz
  - A Virtex-II device is reconfigured in 8.4ms
    - Three Virtex-II devices can be reconfigured in parallel
- No Operating System Support for the Dynamic Scheduling Algorithms

Experiments Set-Up

- **Objective**: Study the System-Level Energy Consumption of the Proposed Dynamic Scheduling Algorithms
- Test Cases Generated using TGFF. Parameters:

Test Case	Exec. Time = x Rcfg. Time	Num. Task Types	Num. Task Successors	Power (mW)	Freq. (MHz)
1	1	10	5	500	33
2	1/2	10	5	1000	66
3	1	5	5	500	33
4	1/2	5	5	1000	66
5	1	10	2	500	33
6	1/2	10	2	1000	66
7	1	5	2	500	33
8	1/2	5	2	1000	66

Experiments Set-Up (II)

- Powering Galapagos from an External Current Source Device
- Power Consumption of a Virtex-II Device is ~450mW
- Total Energy-Consumption is Obtained Adding the Individual Devices Energy
- Two Scheduler Implementations:
  - Scheduler V1 : Does Not Support Configuration Pre-Fetching
  - Scheduler V2 : Supports Configuration Pre-Fetching

Power-Performance Results

- Scheduler V1 vs Scheduler V2
  - Increasing the #DRL's reduces the execution time and energy consumption
  - Frequency scaling reduces execution time but increases energy-consumption

	DRL=2		DRL=3			
Test Case	Exec. Time (uS)	Energy (mJ)	Exec. Time (uS)	Energy (mJ)	Dec. Exec. Time %	Dec. Energy %
1	221640	167.14	181523	167.14	18.10	0.00
	187591	167.15	147.147	167.15	21.56	0.00
2	178581	196.38	144312	196.37	19.19	0.01
	152432	192.78	118670	189.17	22.15	1.87
3	208471	157.32	163898	146.50	21.38	6.88
	178662	153.70	132658	146.51	25.75	4.68
4	152505	179.64	125690	168.79	17.58	6.04
	136217	176.01	105545	168.82	22.52	4.09

Power-Performance Results

Energy Distribution

- Energy for Execution is constant (independent of the used scheduler, #DRL's and Freq. Scaling factor)

	SCHEDULER V1 DRL=2		SCHEDULER V1 DRL=3		SCHEDULER V2 DRL=2		SCHEDULER V2 DRL=3	
Test Case	Energy Rcfg (mJ)	Energy Exec (mJ)	Energy Rcfg (mJ)	Energy Exec (mJ)	Energy Rcfg (mJ)	Energy Exec (mJ)	Energy Rcfg (mJ)	Energy Exec (mJ)
1	64.79	102.35	64.79	102.35	64.80	102.35	64.80	102.35
2	64.80	131.58	64.80	131.57	61.20	131.58	57.59	131.58
3	50.40	106.92	39.58	106.92	46.79	106.91	39.60	106.92
4	46.80	132.83	35.99	132.80	43.20	132.81	35.99	132.83
5	64.79	103.18	61.19	103.21	64.80	103.19	64.81	103.19
6	64.81	127.69	57.60	127.70	68.39	127.68	61.20	127.70
7	46.80	104.76	43.20	104.75	46.80	104.75	39.60	104.75
8	50.40	130.01	43.19	130.00	57.59	130.00	43.20	130.03

Power-Performance Results

- **Energy Distribution**
  - Differences in the energy for reconfiguration (increasing the number of DRL reduces the number of reconfigurations, thus reducing energy)

	SCHEDULER V1 DRL=2		SCHEDULER V1 DRL=3		SCHEDULER V2 DRL=2		SCHEDULER V2 DRL=3	
Test Case	Energy Rcfg (mJ)	Energy Exec (mJ)	Energy Rcfg (mJ)	Energy Exec (mJ)	Energy Rcfg (mJ)	Energy Exec (mJ)	Energy Rcfg (mJ)	Energy Exec (mJ)
1	64.79	102.35	64.79	102.35	64.80	102.35	64.80	102.35
2	64.80	131.58	64.80	131.57	61.20	131.58	57.59	131.58
3	50.40	106.92	39.58	106.92	46.79	106.91	39.60	106.92
4	46.80	132.83	35.99	132.80	43.20	132.81	35.99	132.83
5	64.79	103.18	61.19	103.21	64.80	103.19	64.81	103.19
6	64.81	127.69	57.60	127.70	68.39	127.68	61.20	127.70
7	46.80	104.76	43.20	104.75	46.80	104.75	39.60	104.75
8	50.40	130.01	43.19	130.00	57.59	130.00	43.20	130.03

Power-Performance Results

- **Comparison between Scheduler V1 and Scheduler V2**
  - Effect of frequency scaling combined with configuration pre-fetching
  - Configuration Pre-Fetching helps to obtain similar performance while reducing energy (clock freq.)

	SCHEDULER V1 ExecTime = FAST (1/2x Rcfg. Time)		SCHEDULER V2 ExecTime = SLOW (1x Rcfg. Time)			
Test Case	Exec. Time (uS)	Energy (mJ)	Exec. Time (uS)	Energy (mJ)	Inc. Exec. Time %	Dec. Energy %
1-2	144312	196.37	147.147	167.15	1.96	17.48
3-4	125690	168.79	132658	146.51	5.54	15.21
5-6	138839	185.30	152734	167.99	10.01	10.30
7-8	120770	173.19	145513	144.35	20.49	19.98

Conclusions

- Configuration Reuse and Configuration Pre-fetching Techniques help to reduce both energy and execution time
- Configuration Pre-Fetching jointly with Frequency Scaling and Clock-Gating help to reduce energy consumption
- Implementation details also help to increase performance while minimizing power consumption
  - Hardware Support for Configuration Loading
  - No Operating System Support for Dynamic Scheduling

Thanks!

Questions & Answers