MODLEX: A Multi Objective Data Layout EXploration Framework for Embedded SoCs

Presented By – Udaya Seshua

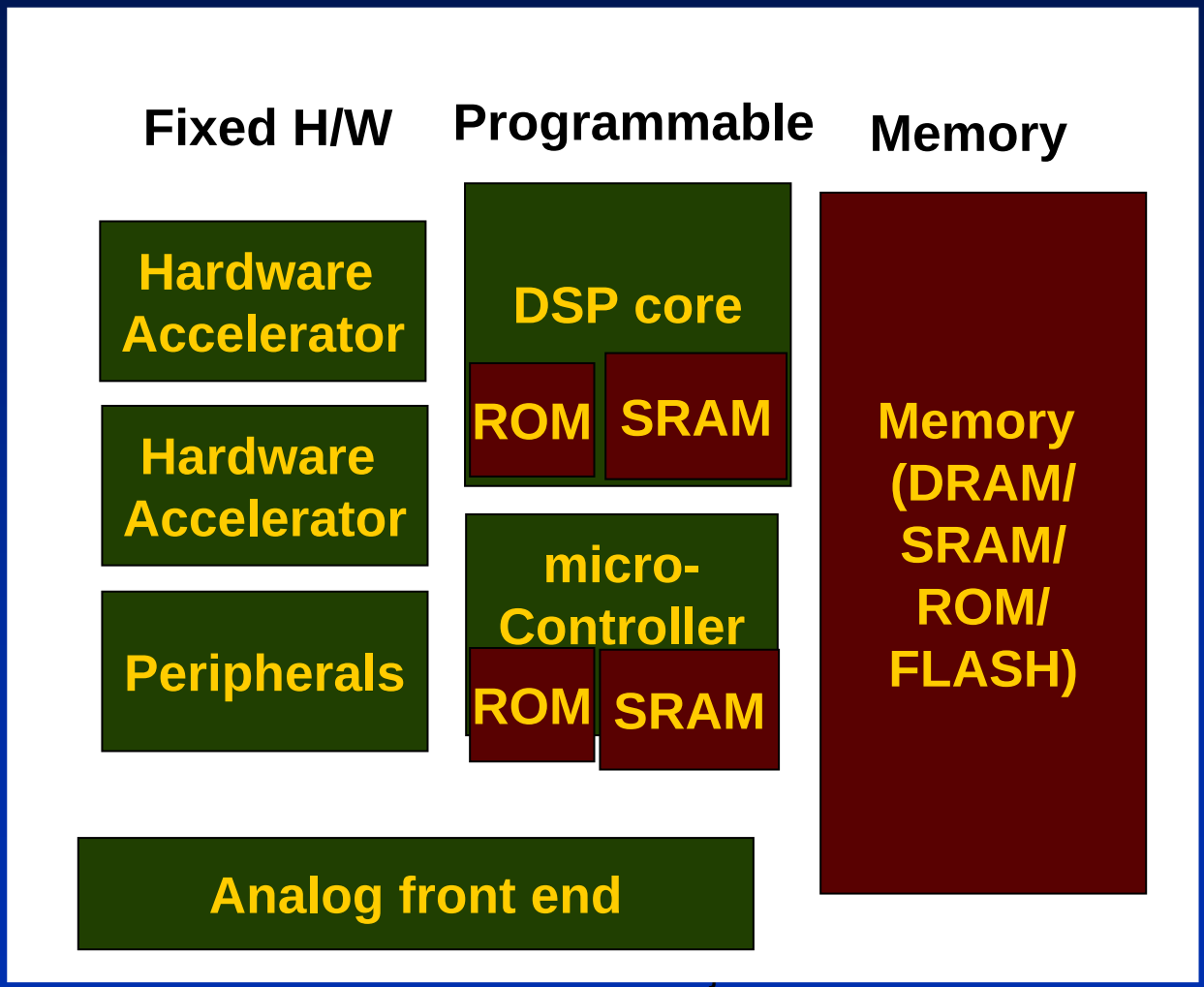
T.S.Rajesh Kumar,
C.P. Ravikumar
Texas Instruments India
Bangalore

R. Govindarajan
SERC,
Indian Institute of Science,
Bangalore

Outline

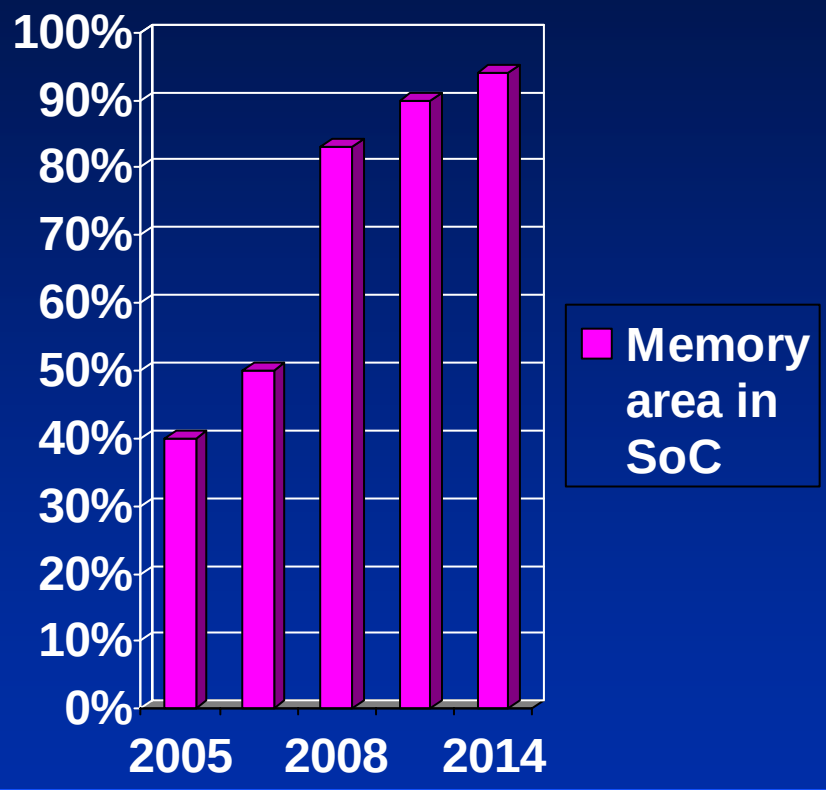
- **Motivation**
- **Problem Summary**
- **Solution Overview**
- **Multi Objective GA for Data Layout exploration**
- **Experimental Results**
- **Conclusion**

Embedded SOC architecture



- Size of on-chip RAM controls system cost
- On-chip SRAM essential for real-time performance
- Memory occupies 50-70% of silicon area in current day SoC

Memory Architecture



Designing Custom Memory Architectures for Applications!

- Cost of Embedded memory is smaller (man power and time to market)
- Power density for memories for embedded memories is lower (than logic)
- Heterogeneous Architecture:
  - On-chip SRAM
    - Organized as multiple banks
    - single/dual port memories
    - Non-uniform bank sizes
  - SDRAM/DDR (multiple banks, burst mode)
  - Flash (NOR/NAND)
- Memory Architecture choice is ***very critical*** for performance

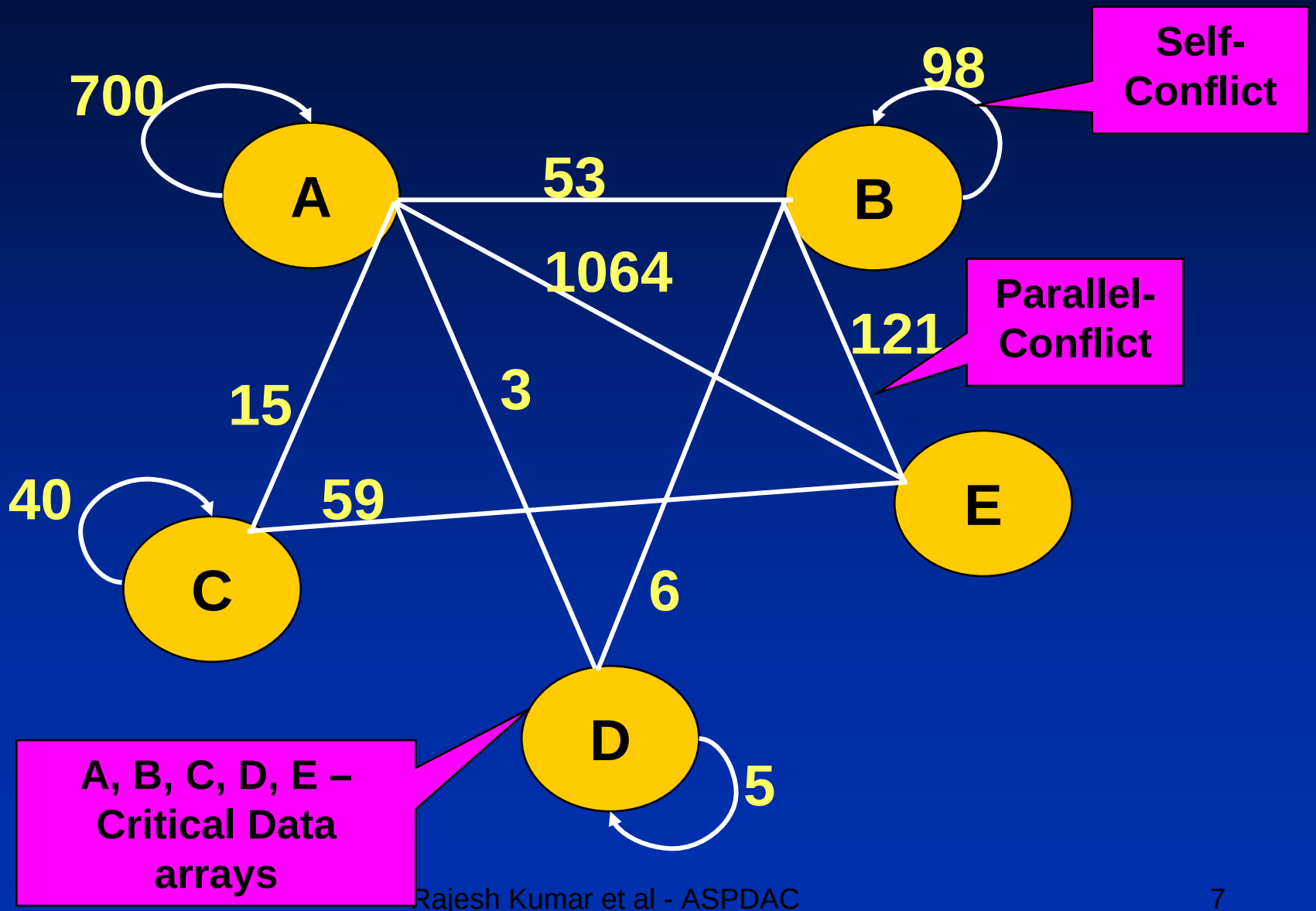
DSP Application Characteristics

- Typically hand-written in assembly
- Very data intensive
- Very high bandwidth requirements
  - **Average 1.4 to 1.8 data memory accesses per cycle**
- For real-time performance, the following are critical
  - On-chip memory architecture
  - **mapping of critical data to on-chip memory**

Memory Latency and conflicts

- Data accesses to off-chip memory incur additional processor cycles due to slower off-chip memory
- On-chip memory bank conflicts:
  - Parallel conflicts
    - For data involved in parallel access (y = x[i] * c[i]), ‘x’ and ‘c’ if placed in the same memory single-port bank, causes bank conflicts
  - Self conflicts
    - For data that are self-accessed multiple times, (y = x[i] * x[k-i]), placing x in single port memory bank results in a conflict
- It is very critical that these additional memory stalls are resolved to improve application performance

Example: DSP Algorithm Data Access



Data Layout - Overview

- Application is profiled to obtain the below info:
  - Access Frequency (Profile data)
  - Conflict Matrix (representation of conflict graph)

$$\begin{pmatrix} 700 & 53 & 15 & 2 & 1064 \\ & 98 & 0 & 6 & 121 \\ & & 40 & 0 & 59 \\ & & & 5 & 0 \\ & & & & 0 \end{pmatrix}$$

• Data Layout: Based on Access frequency and conflict matrix data placement is optimized for a given memory architecture with the objective to reduce memory stalls/bank conflicts

Performance oriented Data Layout Optimizations

- Based on Access Frequency, partition data between on-chip and off-chip memory
- Place simultaneously accessed data in multiple on-chip memory banks to avoid parallel bank conflicts
- Place data with self-accesses in dual-port memory bank to avoid self conflict

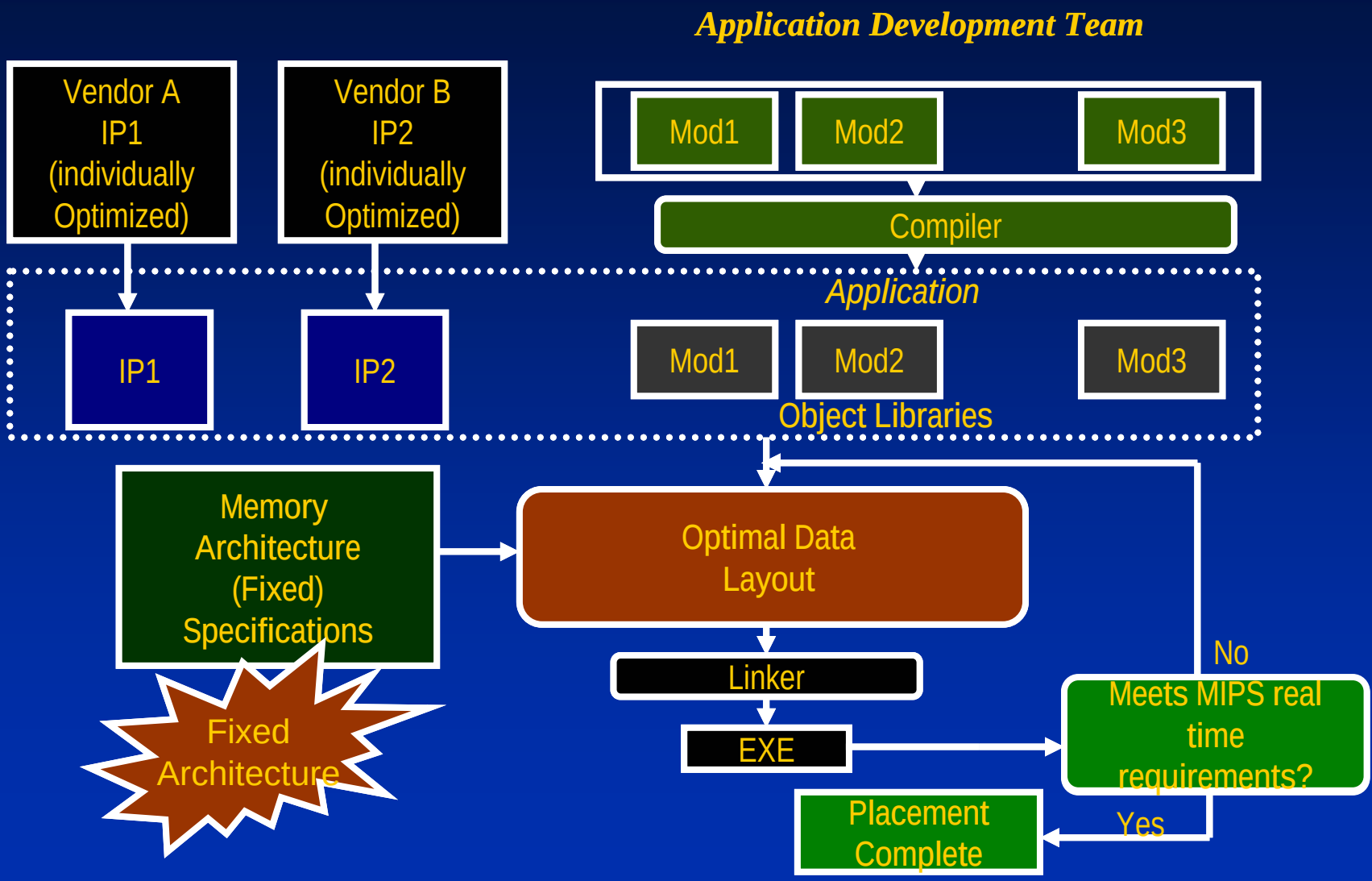
Power oriented Data Layout Optimizations

- Based on Access Frequency, partition data between on-chip and off-chip memory
- Smaller Memory Banks consume lesser power than larger memory banks
  - 16KB bank consumes 2X power compared 4KB bank
- Dual port memories consume more power than single port memories
  - 2KB Dual port consumes 1.8X to 3X power compared to 2KB Single port

Performance Vs Power Trade-off in Data layout

Memories	Impact on Performance	Impact on Power
On-chip Memory	Better performance	Better Power
Multiple Memory Banks	Uniform Sized memory banks better for performance	Non-uniform sized memory banks are better for power optimizations
Dual Port memories	Better for performance	Bad for power

Data Layout and Application dev Flow



Our Solution based on multi objective Genetic Algorithm (GA)

- Problem –
  - Finding ***Optimal data layouts*** that maps the data variables to a given memory architecture with the objective of ***minimizing memory stalls and memory power.***
  - Find all ***Pareto Optimal points*** (each point is a data layout) with respect to power and performance
- Formulated the problem as a multi-objective Genetic Algorithm (GA) problem

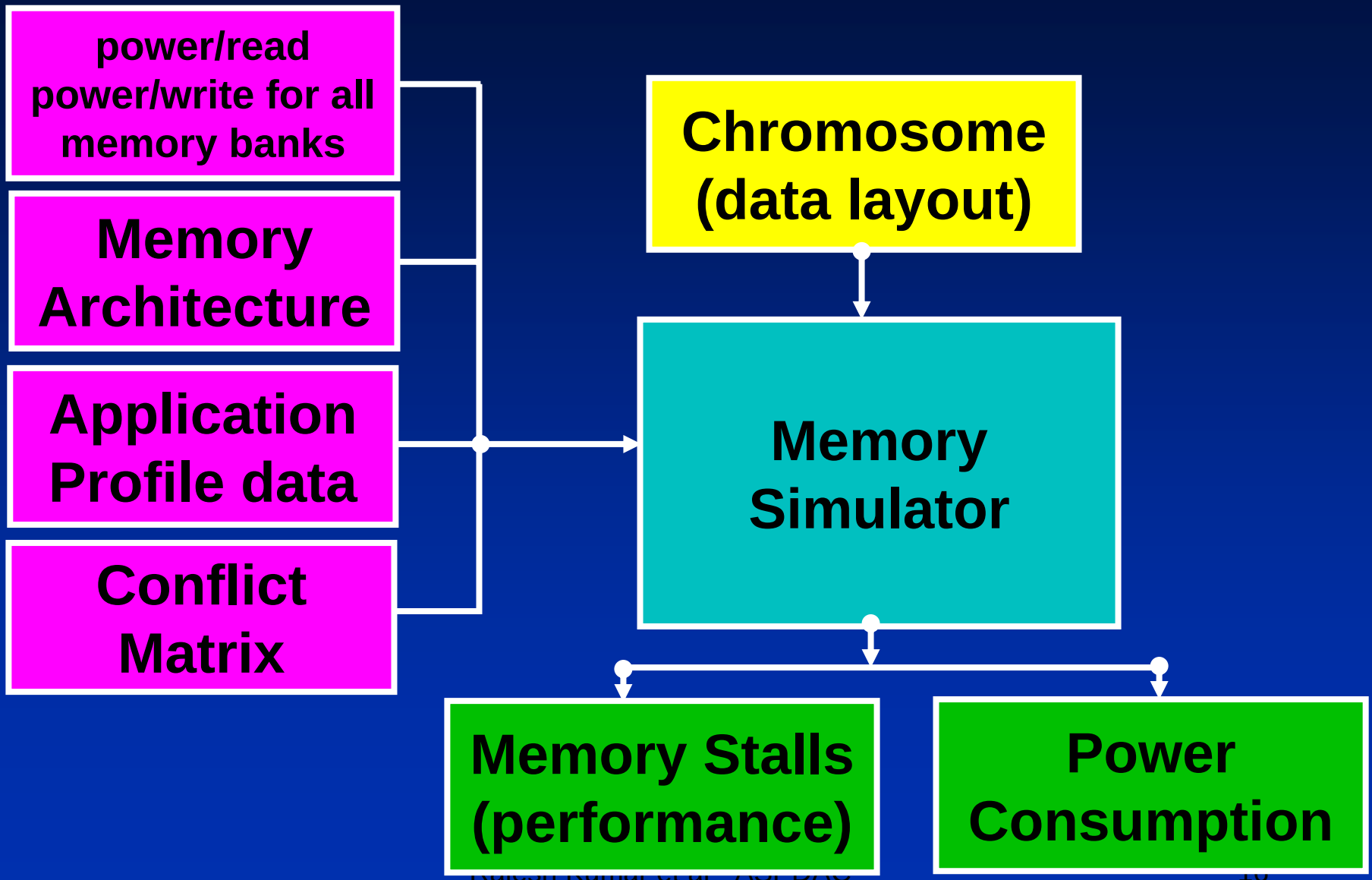
Genetic Algorithm Formulation

- Each chromosome represents a data layout
  - Representation: data placement info
  - d_1<bank-num> d_2<bank-num>… d_n<bank-num>
    - $d_1..d_n$ represents the data variables
    - <bank-num> is the memory bank on which data variable ‘d’ is placed
    - <bank-num> ranges from 1 to maximum number of banks
- Fitness Function: Performance (number of memory stalls), and Power
  - Both objectives are minimizing functions

Genetic Algorithm Formulation

- Fitness Function Computation
  - Performance (number of memory stalls)
    - Data layout obtained from the chromosome
    - Data layout represents a placement
    - For the given placement the number of memory stalls computed by a memory simulator
    - Memory simulator reads the memory architecture, profile data, and conflict matrix as inputs and based on the data layout (memory placement) computes the number of memory stalls
  - Power
    - Based on the profile data (Access Frequency) and the data layout, the memory simulator computes the total power consumption

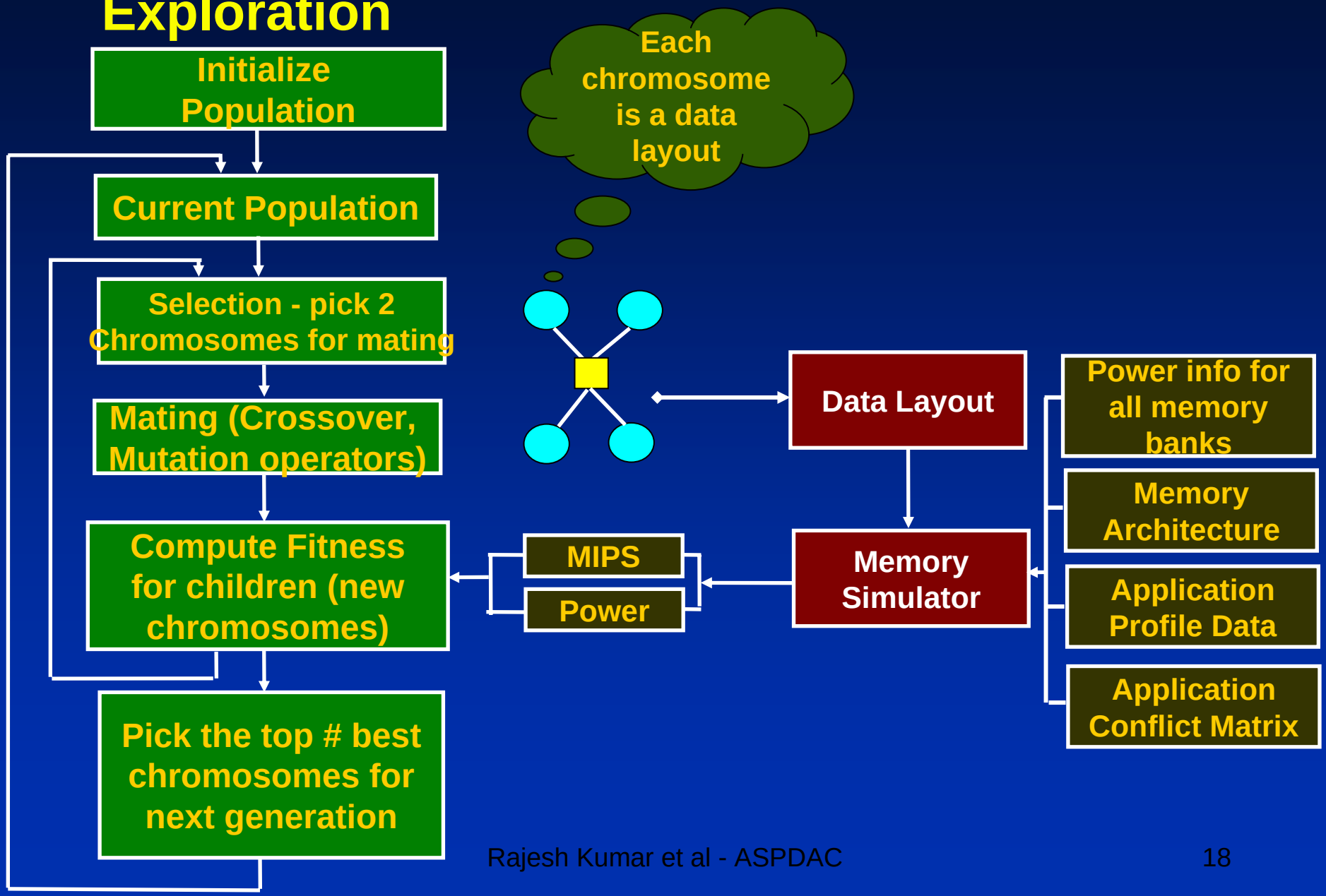
Performance and Power Computation



Genetic Algorithm Formulation

- Crossover and Mutation: Random operators
- Selection: fitness criteria based on Pareto Optimality
  - Let A(mip, pow) and B(mip, pow) are two chromosomes (data layouts)
  - A dominates B iff: ***((A.mip <= B.mip) && (A.pow <= B.pow)) && ((A.mip < B.mip)*** || ***(A.pow < B.pow))***
  - In at least one objective A must be better than B and for all the other objectives it can be equal to or better than B

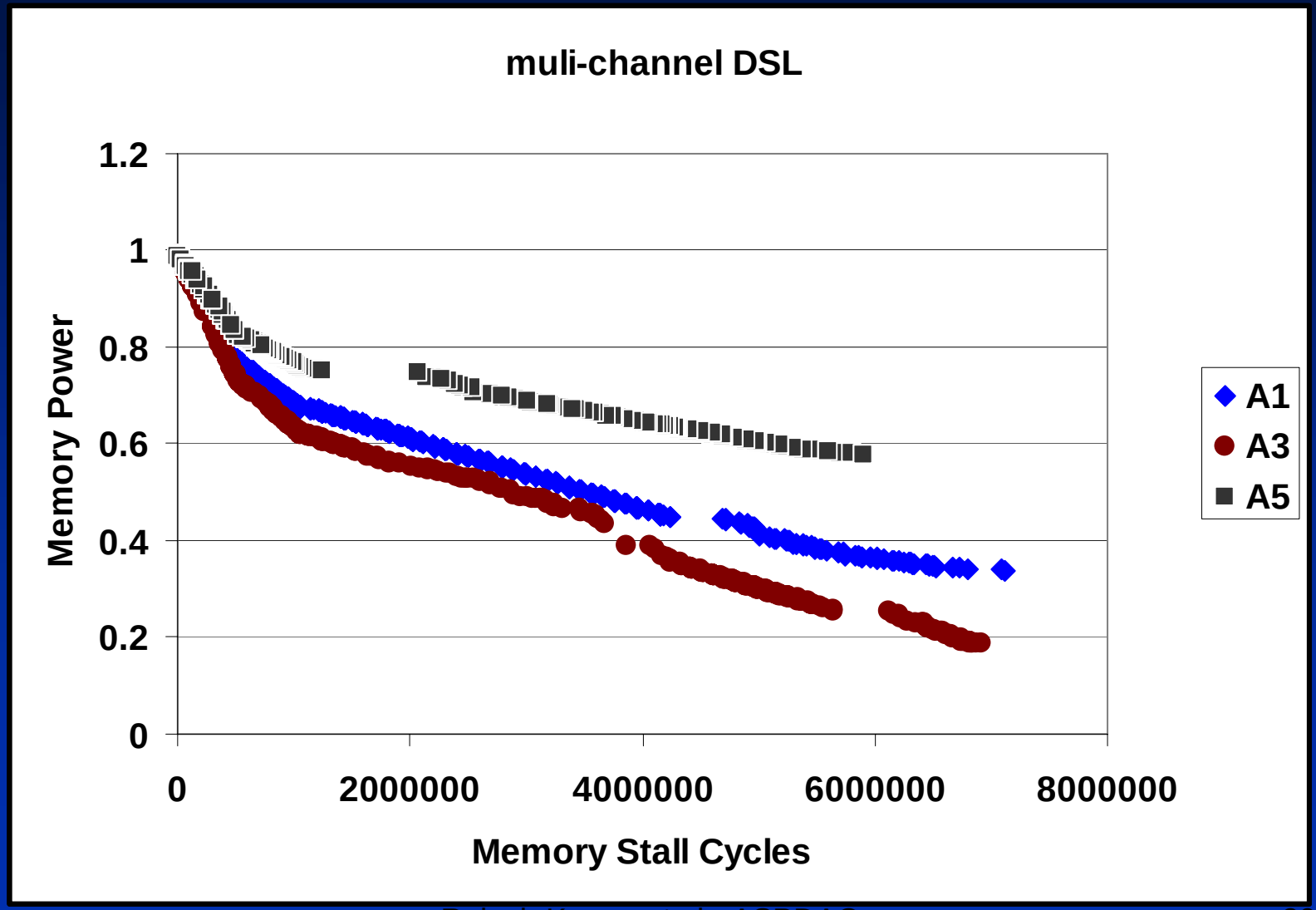
Multi-Objective GA based Memory Exploration



Memory Architecture Used for Experiments

Arch ID	Memory Architecture		Area
	Logical Memory	Physical Memory	
A1	**2 x 8K SPRAM;** **20 x 4K DPRAM**	**2 x 8192 x 16 bit** **20 x 4096 x 16 bit**	**1**
A3	**8 x 4K SPRAM** **1 x 32K SPRAM** **8 x 4K DPRAM**	**8 x 4096 x 16 bit (1P)** **1 x 32768 x 16 bit (1P)** **8 x 4096 x 16 bit (2P)**	**0.82**
A5	**2 x 32K SPRAM** **8 x 4K DPRAM**	**2 x 32768 x 16 bit (1P)** **8 x 4096 x 16 bit (2P)**	**0.72**

Experimental Results (1)



Conclusions

- Presented a ***Data Layout Exploration Framework***
- Explore the whole solution space and present a set of ***optimal design choices*** based on memory ***power*** and run-time ***performance***
- Results on real-world applications show that there is a ***wide range of operating points that trade off power/performance*** and can be used for efficient system level power management
- Fully automated flow for the data layout exploration

Thank You!

Please send your Questions to:

Rajesh Kumar (tsrk@ti.com)