Process-Variation Aware Mapping of Real-Time Streaming Applications to MPSoCs for Improved Timing Yield

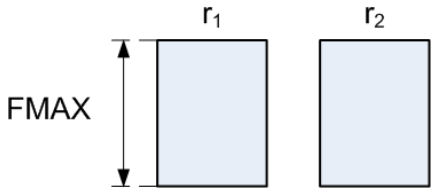
Davit Mirzoyan, Benny Akesson, Kees Goossens

Background

- Inability to precisely control the manufacturing process in deep-submicron technologies
  - Variation in key transistor parameters (i.e. V_{th})
  - Identically designed cores in a chip and across chips have different maximum supported frequencies (FMAX)
    - Up to 40% variation in FMAX of a VLIW processor manufactured at 32nm technology [M. Miranda *et al.*, ISQED, 2009]
- To efficiently design MPSoCs
  - Variation-aware performance analysis is essential in the process of system-level task allocation

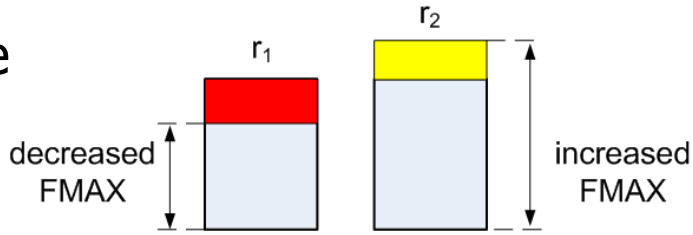
Concept Demonstration

- Hardware resources (i.e. cores) with nominal FMAX



  - Sufficient performance for the timing requirement of an application

- Variation affected hardware

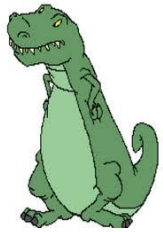


  - Still sufficient performance for the timing requirement of an application

- Variation-aware performance analysis can increase the timing yield
  - The number of chips satisfying the timing requirement of an application

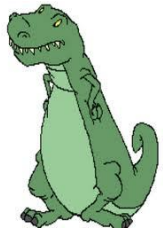
Outline

- Positioning the work
- Modelling
- Optimization problems
- Implementation algorithms
- Experimental results
- Conclusions



Outline

- **Positioning the work**
- Modelling
- Optimization problems
- Implementation algorithms
- Experimental results
- Conclusions

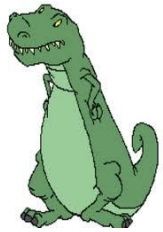


Positioning the work

- Existing solutions that propose variation-aware task allocation and scheduling
  - Use acyclic task-graphs for application modelling
  - Are based on latency requirements
- Our approach
  - Allows arbitrary task graphs with cyclic dependencies
  - Is based on throughput requirements of real-time streaming applications
    - Primary timing requirement

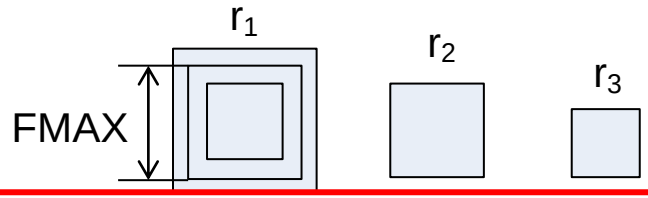
Outline

- Positioning the work
- **Modelling**
- Optimization problems
- Implementation algorithms
- Experimental results
- Conclusions



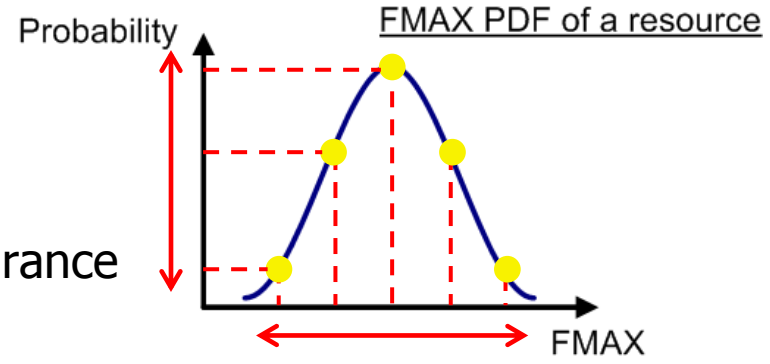
Hardware platform

- Set of hardware resources (processor, DSP, accelerator)



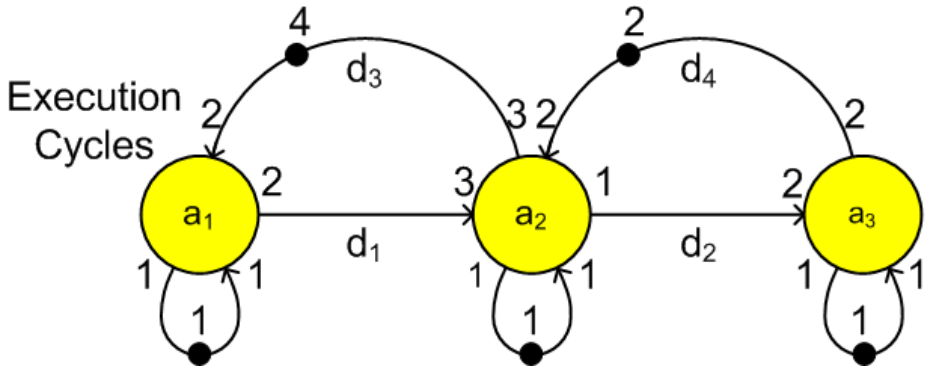
- Each resource has a set of possible operating points (FMAX) with associated probabilities

- Chips with various operating point sets
  - $os = (FMAX_i{}^{r1}\ FMAX_j{}^{r2}\ FMAX_k{}^{r3})$
  - Each chip (os) has a probability of occurance



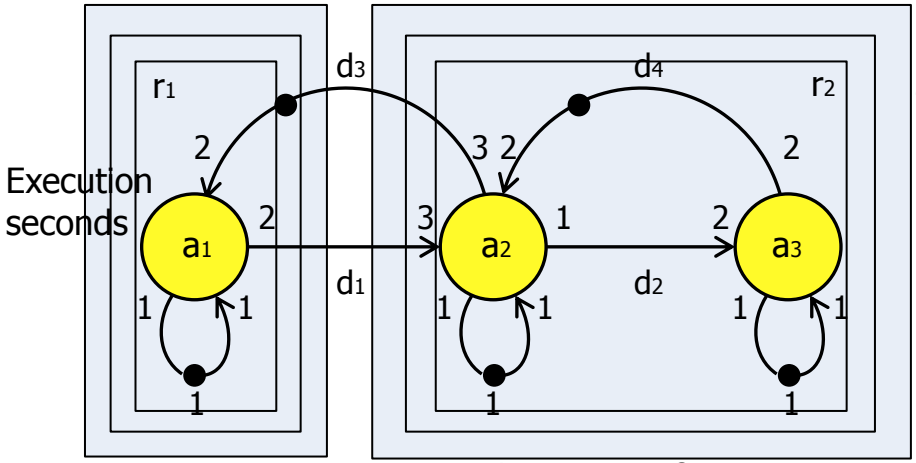
Unbound graph

- Synchronous Data Flow Graph (SDFG) model of an application
  - Binding unaware
  - Execution times of actors are given in clock cycles on the resources
- This graph is decoupled from hardware variation

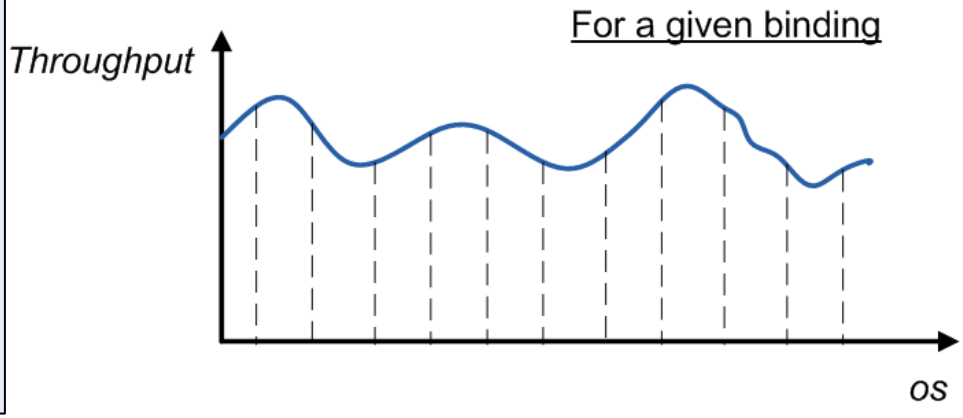


Bound graph

- SDFG model of an application
  - Binding aware
    - Multiple ways of binding application actors to the resources
  - Execution times of actors are given in seconds
- Bound graph is not decoupled from hardware variation
  - Impact of hardware variation on throughput for various bindings



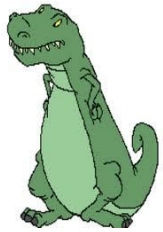
os={FMAX$_i^{r1}$,FMAX$_j^{r2}$}



Changing the *os* we can observe variation in throughput

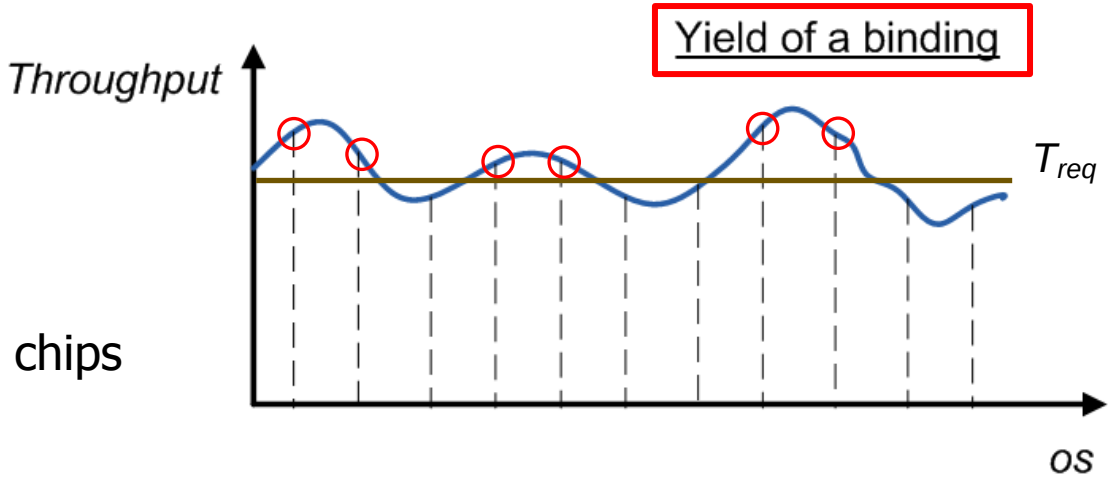
Outline

- Positioning the work
- Modelling
- **Optimization problems**
- Implementation algorithms
- Experimental results
- Conclusions



Single-binding optimization

- Two optimization approaches are presented
- Objective is to maximize the timing yield
  - The number of chips satisfying application throughput requirement T_{req}
- Single-binding optimization
  - Find a binding at design time such that the timing-yield is maximized
  - Yield of a binding
  - Identical binding for all chips

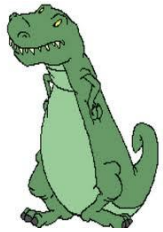


Multiple-bindings optimization

- Multiple-bindings optimization
  - Find and store a binding for each chip at design time
    - Based on each manufactured chip, the right binding is selected at the run-time configuration stage
  - Advantage over single-binding
    - Per-chip binding selection results in higher timing-yield
  - Disadvantage
    - Diverse software instances for the same product
    - Multiple bindings are stored

Outline

- Positioning the work
- Modelling
- Optimization problems
- **Implementation algorithms**
- Experimental results
- Conclusions



Exhaustive algorithm

- Exhaustive and Heuristic algorithms are presented
- Exhaustive
  - All binding possibilities are evaluated
    - Number of all bindings: $|R|^{|A|}$
      - Exponential complexity
  - Gives the optimum solution
    - Maximum improvement in yield
  - Too computationally expensive for large problems

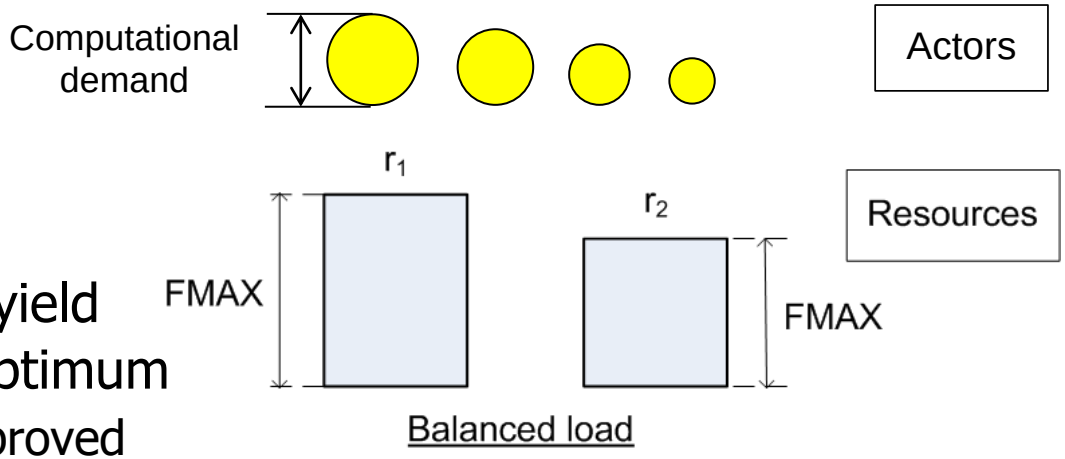
Heuristic algorithm

- Small number of bindings are evaluated
  - |A|*(|R|-1)
  - Exponential complexity is reduced to polynomial
- The bindings are generated by two phase procedure

1) Initial resource allocation
   - An initial binding is derived
2) Allocation optimization
   - Actors are moved from a resource to another to improve the yield

Initial resource allocation

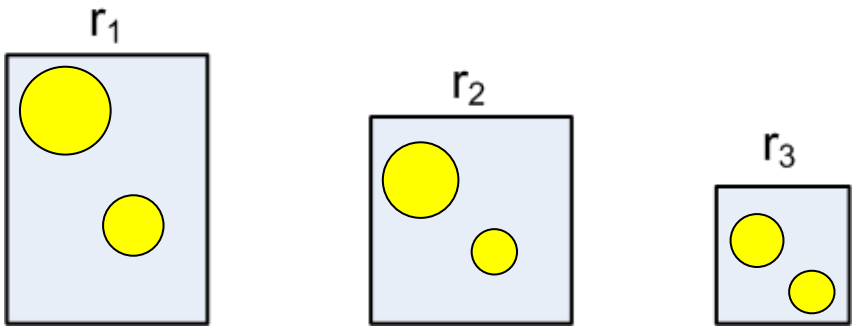
- Actors with high computational demands are considered first
- Load balancing on the resources
- When allocating an actor
  - Select a resource with the lowest load
  - Not allocated resources?
    - Select the fastest resource



- Potentially results in high yield
- Doesn't necessarily give optimum
  - The allocation can be improved

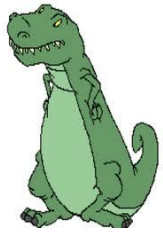
Allocation optimization

- Allocation of actors is reconsidered
  - To find a binding with higher yield for all chips or throughput per chip
- Strategies
  - Pair-wise swapping of actors (all combinations)
  - Moving each actor from a resource to another (all combinations)
    - Gives better results
- Still may not give the optimum as not all bindings are evaluated



Outline

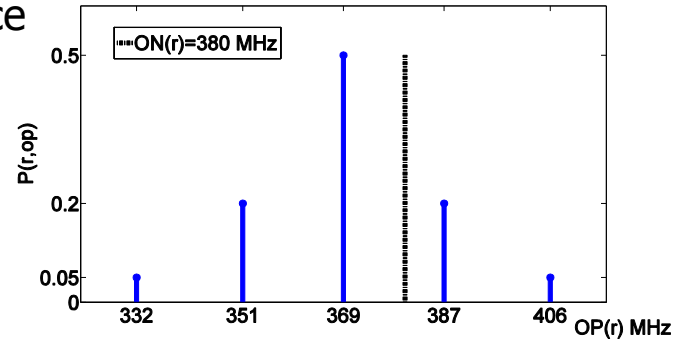
- Positioning the work
- Modelling
- Optimization problems
- Implementation algorithms
- **Experimental results**
- Conclusions



Experimental setup

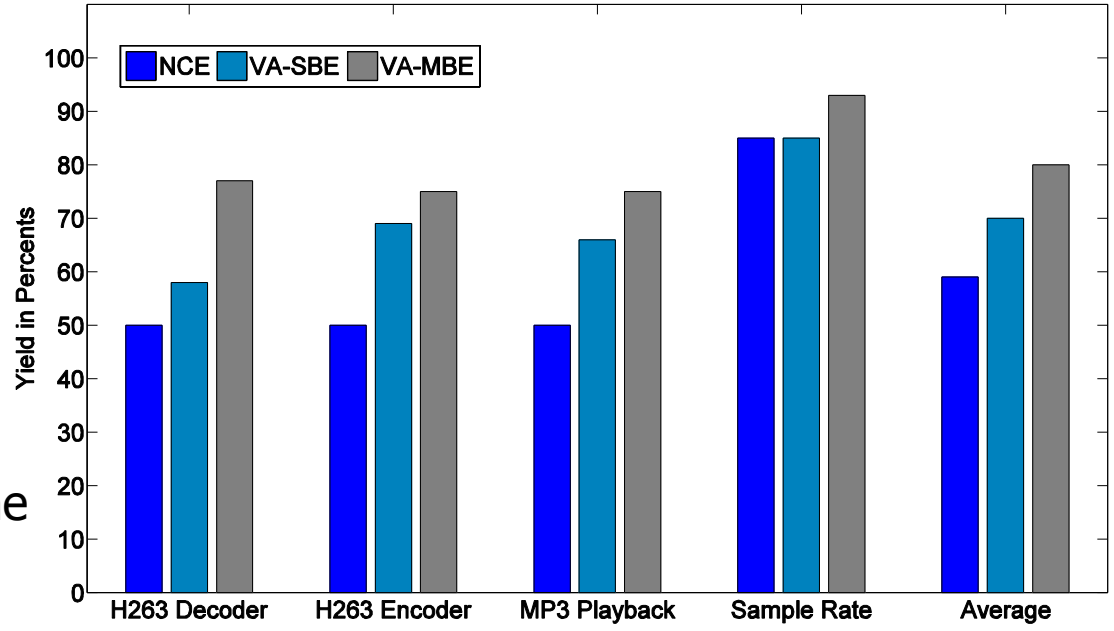
- Applications
  - H.263 Encoder, H.263 Decoder, MP3 Decoder, Modem, Satellite Receiver, MP3 Playback and a Sample Rate Converter
- Resources
  - 2-5 resources with 380, 380, 380, 440, 500 MHz nominal FMAX
  - Impact of process variation [M. Miranda *et al.*, ISQED, 2009]
    - 3%, 6% and 15% FMAX mean degradations (intra-die)
    - 3.3% standard deviation (3σ=10%) for all resources (inter-die)
    - 5 discrete operating points for each resource
- SDF3 for throughput analysis

  [S. Stuijk *et al.*, ACSD, 2006.]



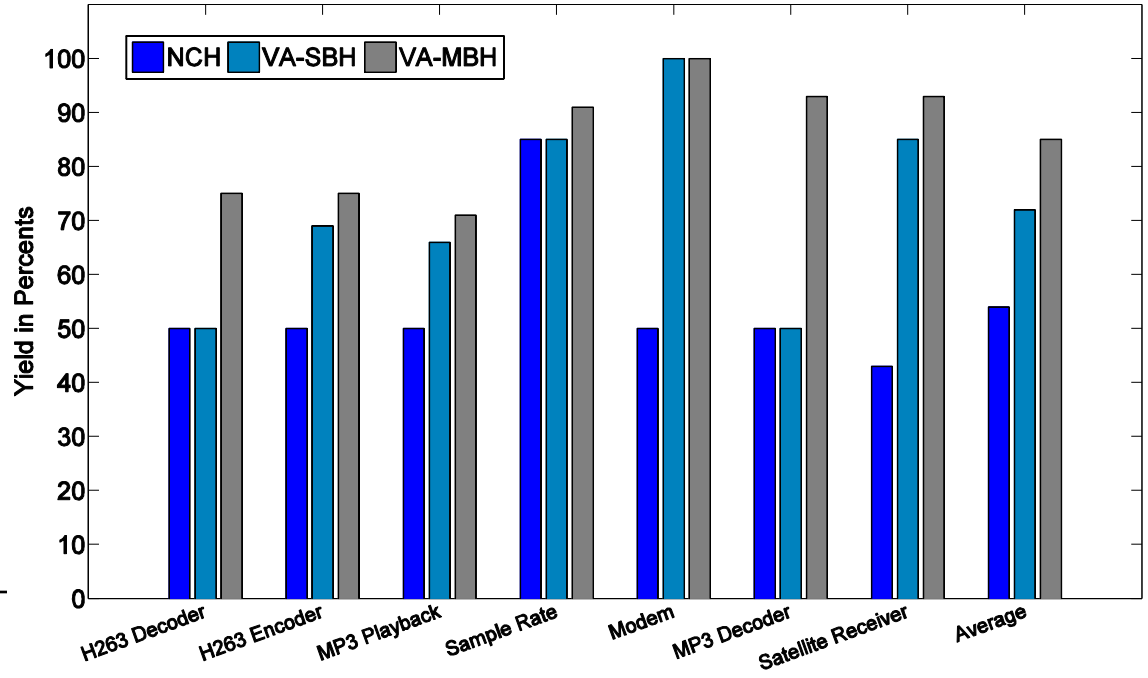
Exhaustive algorithm

- Comparison to variation-unaware nominal frequency-based mapping approach
- Medium sized applications: 6 actors the largest
- Average improvements in yield
  - VA-SBE: 11%
  - VA-MBE: 21%
    - Better than VA-SBE
- Run time: ~1hour
  - On P4 2.8 GHz machine



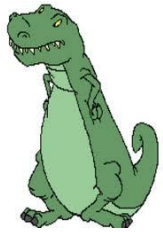
Heuristic algorithm

- 4% average reduction in yield as compared to optimum
- Applied to large applications (22 actors the largest)
  - Higher yield improvements: up to 50%
- Run time: ~15 min. (for large applications)
  - Exhaustive approach is infeasible (for large applications)
- Not more than 10 bindings are stored for VA-MBH



Outline

- Positioning the work
- Modelling
- Optimization problems
- Implementation algorithms
- Experimental results
- **Conclusions**



Conclusions

- Single-binding and multiple-bindings approaches for mapping cyclic task graphs to MPSoC for improved timing yield under process variation
- Exhaustive and Heuristic algorithms that implement the optimization approaches
- Variation awareness is important in resource allocation
  - Up to 50% yield improvements (31% average)
- Heuristic algorithm scales well to large problems while giving slight reduction in yield (4% average)
- Only a few bindings are stored for the run-time configuration

Questions?