15th International Forum on Embedded MPSoC and Multicore

MPSoC 2015

NanoProcessor Cluster: Naturally Minimizing Active Portion for Dark-Silicon Era

July 13th, 2015

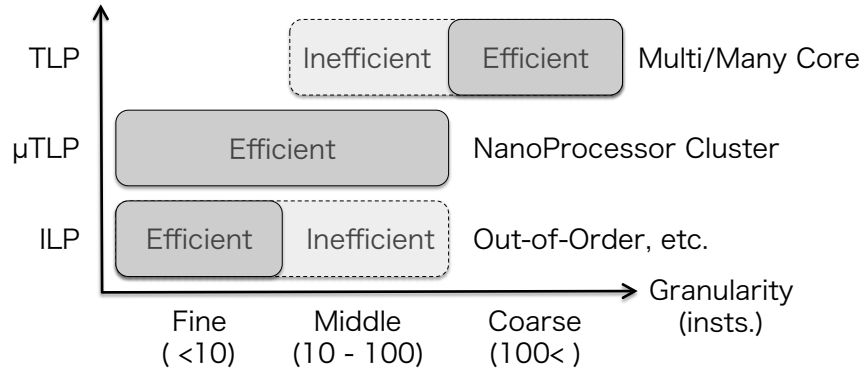
Fumio Arakawa
Nagoya Univ., SH Consulting

Motivation: Total Efficiency

- *Multi-Many Cores* : excellent for High-TLP portions
- *Amdahl's Law* : High-speed core for Low-TLP portions.
- *Pollack's Law* : High-speed core is inefficient.
  - Conventional High speed core
    => (large, complicated, fast) => inefficient
  - High ILP => large scope => large fast buffers
  - Large but high frequency => many pipeline latches
- Nanoprocessor + overrun buffer
  => (small, fast, active) + (large, dense & slow, inactive)
  => Efficient => good for *embedded systems*

Granularity of Parallelism

- Fine/Coarse Grain: fit to ILP/TLP *), respectively
- Middle Grain: Bad for conventional Arch.
- µTLP of NanoProcessor Cluster fits to Middle Grain

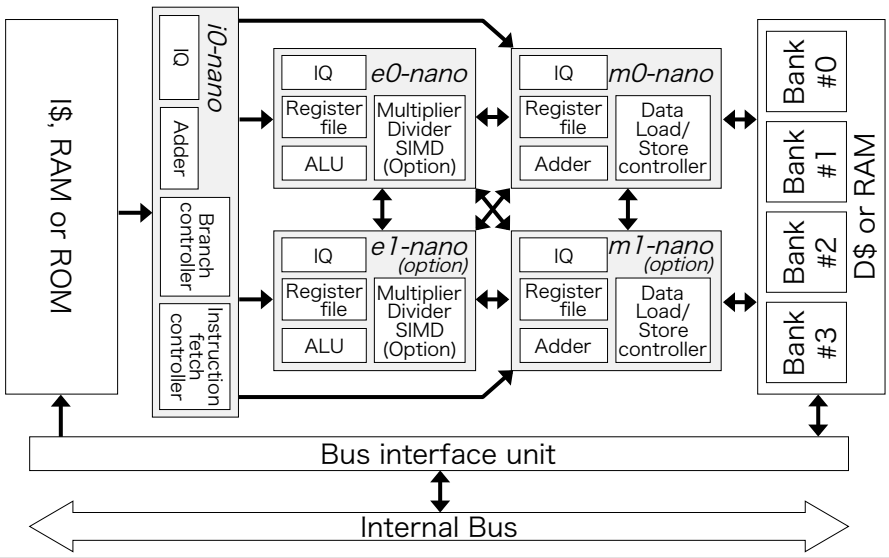


*) ILP / TLP : instruction/thread level parallelism

Conversion from ILP to µTLP

- High speed microprocessor core utilizes ILP.
  - *Pipelined* Architecture: Parallelism = # of the pipeline stages
  - *Superscalar* Architecture: Parallelism = # of issue slots
  - Good for Fine Grain parallelism
- Conversion from ILP to µTLP
  - *µ-Thread* : a functionally divided flow of original single thread
    Flows are tightly coupled.
- from High speed core to "*NanoProcessor Cluster* "
  - *NanoProcessor* : Subset of Conventional CPU
    Execute a µ-Thread
    Scalar short-pipeline processor: for limited ILP of µ-Thread
  - Its *Cluster* : equivalent to microprocessor
    Good for both Fine and Middle Grain parallelism
    highly efficient and high performance

Block Diagram



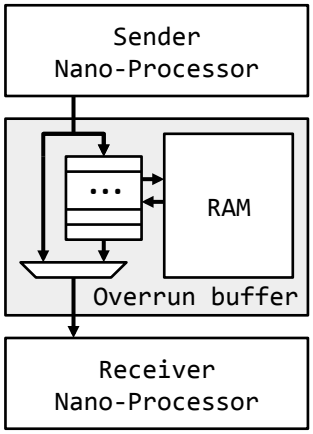
Overrun buffer

- Toward small register file
  - register: most expensive storage, top of memory hierarchy
  - long life for long pipeline => must be large
  - allocate => idle => write => n-time (n≧0) read => idle => retire
- with “Overrun buffer” & “Keep” flag
  - allocate register at write back stage
    - fail to allocate for write after read => write to overrun buffer
  - retire just after the last read (check “Keep” flag)
    - not to wait overwrite (conventional retire sign)
  - allocate & write => n-time (read & keep) => read & retire
  - Small register file is enough.
- Small core + Overrun buffer

Overrun buffer (continued)

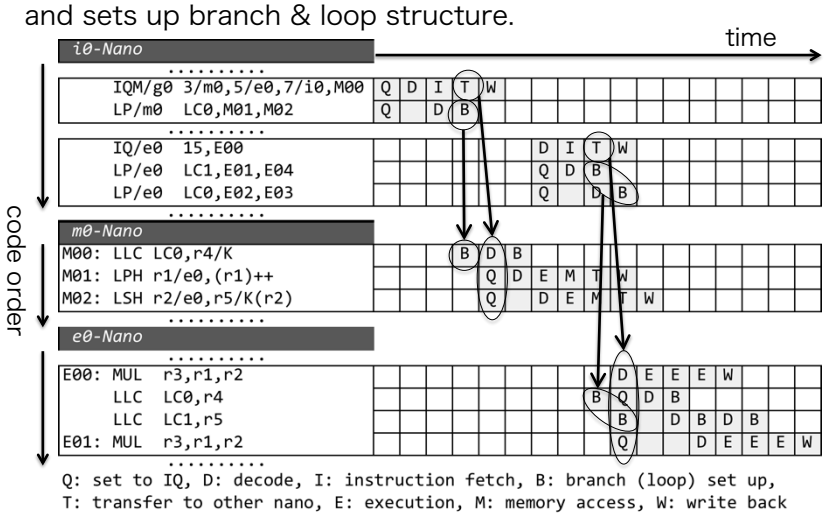
- To absorbs write/read timing gap of NanoProcessors
  - While sender and receiver paces are well synchronized, overrun buffer is idle.
  - Once receiver stalls, it requests to stop data sending and keeps all the on-the-fly data in overrun buffer.
  - On-the-fly data can be huge in a high-throughput long-latency case, and a RAM is useful.
  - Either the sender/receiver or overrun buffer is active.

    => fit to **Dark silicon era**



Instruction fetch & branch set up

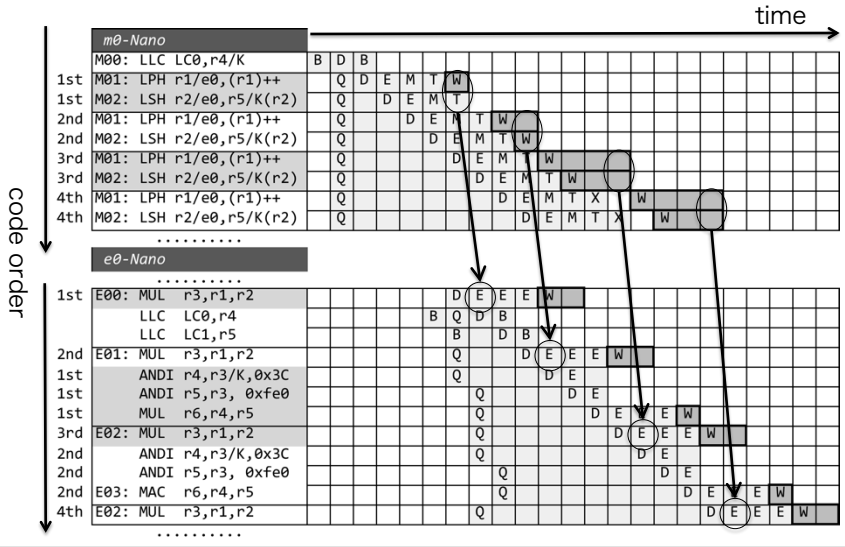
- i0-nano Processor explicitly fetches instructions, and sets up branch & loop structure.



i0.m0.e0 Cluster case, extracted from matrix_mul_matrix_bitextract of Coremark 1.0

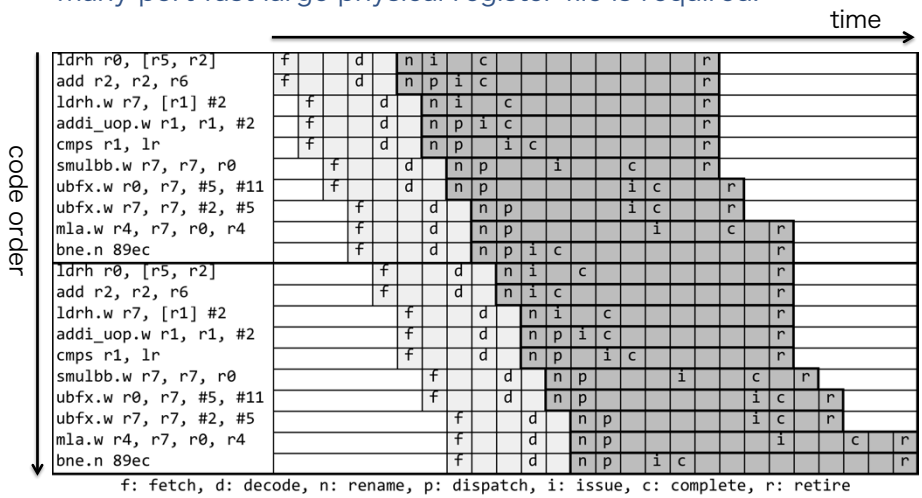
Minimize life time of register values

- Overrun buffers & keep flag => register life: write to last read



Long life time of register values

- Out-of-Order processor => rename to retire
- Many-port fast large physical register file is required.



f: fetch, d: decode, n: rename, p: dispatch, i: issue, c: complete, r: retire

Instruction Examples

Instruction	Operation
`IQM/g0 3/m0,5/e0,7/i0,M00`	Load 3/5/7 instructions. from label M00 to multiple IQs of m0-/e0-/i0-nano, respectively
`LP/m0 LC0,M01,M02`	Set loop from M01 to M02 of m0-nano
`LLC LC0,r4/K`	Load r4 value to loop counter #0 ("/K" indicates keep r4 value.)
`LPH r1/e0,(r1)++`	Load half word from memory pointed by r1 to r1 of e0-nano, and increment r1.
`LSH r2/e0,r5/K(r2)`	Load half word from memory pointed by r2 to r2 of e0-nano, and add r5 to r2.
`IQ/e0 14,E00`	Load 14 instructions. From label E00 to IQ of e0-nano
`MUL r3,r1,r2`	Multiply r1 and r2, and write result to r3
`ANDI r4,r3/K,0x3C`	Logical AND of r3 and 0x3C, and write result to r4
`MAC  r6,r4,r5`	Multiply r4 and r5, and accumulate it to r6

Summary

- NanoProcessor + overrun buffer = fit to dark silicon Era
  - (small, fast, active) + (large, dense, slow, inactive)
- Evaluation: matrix_mul_matrix_bitextract (Coremark 1.0)

	Out-of-Order	NanoProcessor Cluster	
memory latency	2	2	10
execution cycles	17,393	13,736	13,744
ratio to Out-of-Order		78.97%	79.02%

Future Work

- Search for suitable applications
  - Embedded, Networking, HPC, …
- More Tools, Evaluations, and Collaborations
  - Compiler, Benchmark, New types of NanoProcessors, …