Tutorial: Partially Asynchronous Microprocessors
MICRO-35
Istanbul, Turkey, Nov. 18, 2002

Diana Marculescu (CMU)
Dave Albonesi (Rochester)
Pradip Bose (IBM)
Alper Buyuktosunoglu (Rochester/IBM)

Schedule

- **2:00-3:05 (Diana Marculescu)**
  - Trends and issues in clock distribution
  - Motivation for GALS, synchronization issues, deadlock prevention, possible inter-clocking domain communication schemes
  - GALS processors: power/performance evaluation
- **3:05-3:30 (Dave Albonesi)**
  - GALS processors: power/performance evaluation (cont'd)
- **3:30-4:00 Break**
- **4:00-4:40 (Dave Albonesi)**
  - Workload characterization and impact on the use of fine grain speed/voltage scaling
- **4:40-5:45 (Alper Buyuktosunoglu, Pradip Bose)**
  - Case study - LPX, an IPCMOS based processor
- **5:45-6:00 (Diana Marculescu)**
  - Looking in the crystal ball: Where will partial asynchrony be used?
  - Concluding remarks

But first… What is "partially asynchronous"?

What is partially asynchronous?…

- **Asynchrony in (almost) any shape or form…**
- **… But mostly:**
  - Globally Asynchronous, Locally Synchronous (GALS) designs
  - Self-timed logic within fully synchronous systems (Locally Asynchronous, Globally Synchronous)
  - Not fully asynchronous
- **Thus, "partially asynchronous" encompasses design styles that**
  - Do not have a global timing reference
  - Rely on synchronization mechanisms
  - Are not fully asynchronous

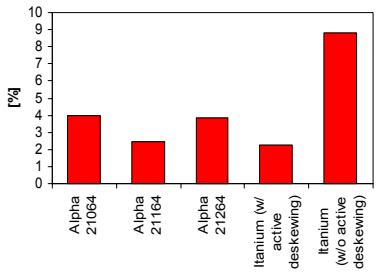
Why partially asynchronous?

- **Current design roadblocks:**
  - Increasing power density
  - Increasing design complexity
- **Shortening time-to-market require:**
  - Design and verification tools and methodology
  - Solutions for design testability and reliability
- **Drivers for minimal clocking:**
  - Design reuse (IPs independently designed)
  - Minimize design effort for global clock distribution
  - Allow fine-grain application-driven adaptability (voltage and speed scaling)

Trends and issues in clock distribution

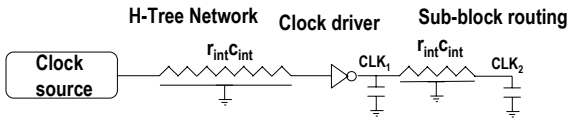
Global clock distribution issues

- **Global clock distribution is getting more difficult**
  - Larger die size and device count
  - Higher clock speed
  - Increasing clock power consumption (30-40% of total power)
- **Clock skew has become a large part of the cycle time**



Clock skew

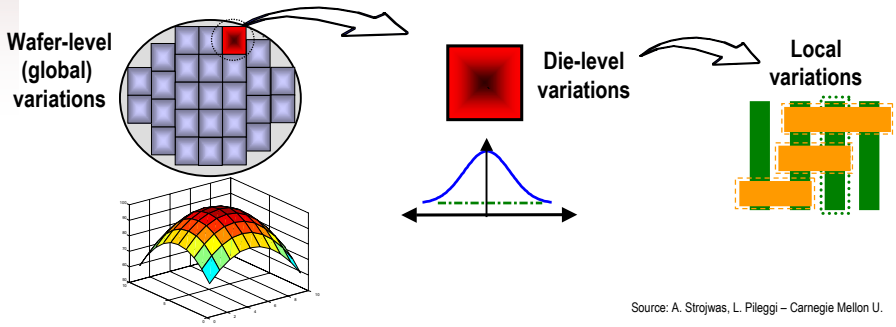
- **Case study: H-tree**
- **Skew components:**
  - Interconnect delay from clock source thru the H-tree to the driver
  - Transistor delay of the sub-block driver
  - Internal wire routing delay within the sub-block from clock driver to registers – also called *internal clock skew*



- **The first 2 components are most affected by physical parameter variation**

Interconnect and Transistor Variations

- Manufacturing and cost factors will force us to design with larger relative) parameter variations than ever before
- New models are required to produce reliable designs
- First-order approximation: Assume only simple linear variations for global variations



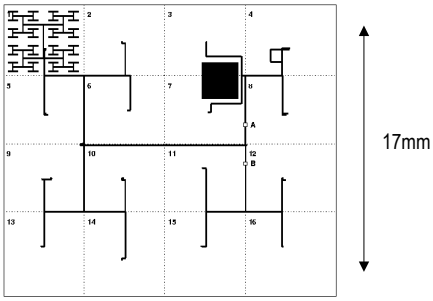
Source: A. Strojwas, L. Pileggi – Carnegie Mellon U.

Case study: IBM 1.0 GHz Clock Tree

- **Description:**
  - Main clock tree in a microprocessor, which has the size of approximately 17x17mm.
  - Based on a 0.25mm CMOS technology with full copper interconnect.
  - Design target of the operating clock frequency 1GHz.
  - Topology:
  - Widths optimized. Shields running in parallel on both sides. Routed solely in top metal layers: LM and MQ.
  - Top level: balanced H-tree over the entire chip area.
  - Second level: 16 sectors. Each sector a balanced H-tree with minor modifications.

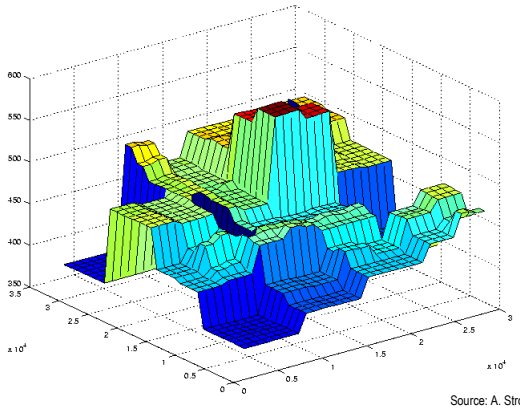
IBM 1.0 GHz Clock Tree

- One gigahertz clock tree
- 0.25 micron technology, Cu interconnect
- Shielded wires on top level metals
- Sixteen sectors



IBM 1.0 GHz Clock Tree

- Model the top level and 16 sector H-trees as lumped RC network.
- Simulated with non-linear drivers to obtain nominal clock skew of 187ps.
- "Hot" sectors: 7 (slowest) and 13 (fastest)



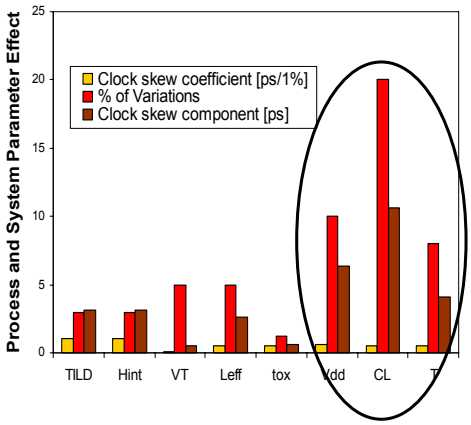
Source: A. Strojwas, L. Pileggi – Carnegie Mellon U.

IBM 1.0 GHz Clock Tree

- Top layers have 15% across-wafer tolerance in width, height and ILD thickness
- Use the slant plane model for the variations and choose 10% as the total across-die variation magnitude for three geometries for each metal layer
- Skew is changed by as much as +/-25%!

➔Clearly impacts the distribution of clock and the use of synchronous design styles for large high-frequency ICs

Clock skew components



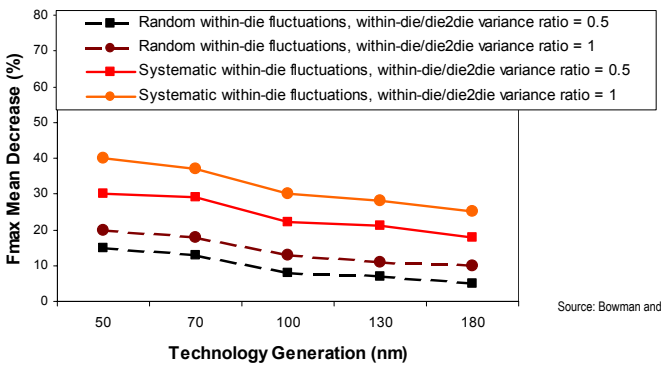
Source: Zarkesh-Ha, Mule and Meindl, 1999

- Non-uniform distribution of clocked registers, Vdd and temperature may have the highest impact on clock skew
  - Assumptions: 20,000 clocked registers and Alpha's temperature map

How bad can the variation be?

- **Clocked pipeline registers**
  - Worst case variations of up to a factor of 3
  - May translate into 20% variation among sub-block loading capacitances
- **Temperature variations**
  - Some structures are hotter than others…
  - …And thus induce further increase in clock skew

Adverse effect of process variation

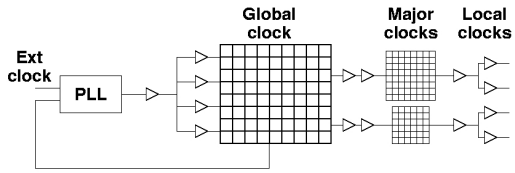


Source: Bowman and Meindl, 2001

- **Significant effect on maximum clock speed**
  - Up to 30-40% reduction in clock speed for 0.05 um technology
- **Need design styles that reduce these variations**

Current solutions for clock distribution

- **Traditionally done with PLL, metal grids and H-trees**
  - Requires lot of die area, power, design effort
- **Nowadays, clock distribution is done hierarchically**
  - Global clock and major clocks
  - Deskewing circuits – may incur a lot of design effort and silicon



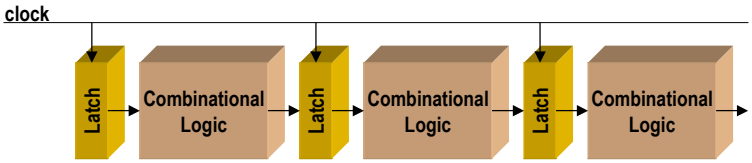
- **ITRS 2001 (SIA Roadmap)**
  - GALS design style for alleviating clock distribution problems

How about fully asynchronous design?

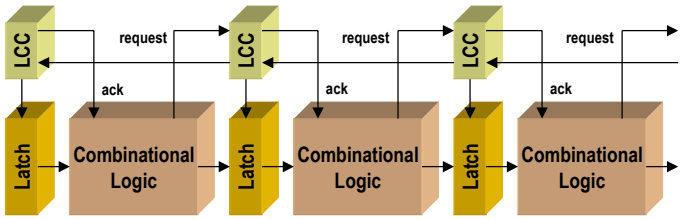
- **Fully asynchronous benefits**
  - No clock, very low power
  - Average case performance may be better
  - No global clock distribution network

Synchronous vs. asynchronous pipelines

- Synchronous

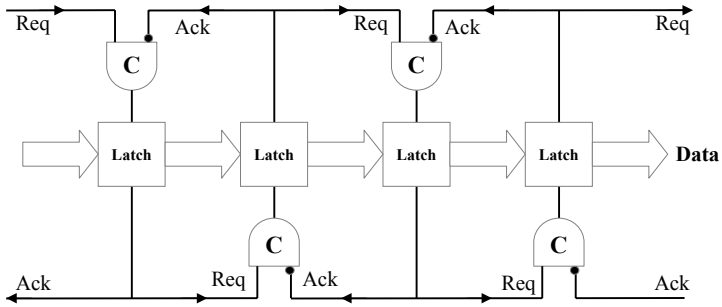


- Asynchronous



Micropipelines

- Introduced in 1989 [Sutherland, CACM 1989, Turing Award Lecture]
- Handshaking based on transition signaling instead of standard logic levels



So… Why not fully asynchronous?

- **Fully asynchronous may not have the clock distribution problems…**
- **However…**
  - Complete migration to asynchronous is not feasible immediately
  - Lack of mature CAD tools for fully asynchronous design…
  - … and inertia
- **Partial asynchrony or GALS provides an intermediate solution**
  - Assumes locally clocked, fully synchronous regions
  - Globally asynchronous communication among regions

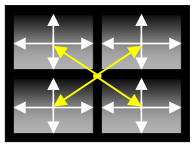
Schedule

- **2:00-3:05 (Diana Marculescu)**
  - Trends and issues in clock distribution
  - Motivation for GALS, synchronization issues, deadlock prevention, possible inter-clocking domain communication schemes
  - GALS processors: power/performance evaluation
- **3:05-3:30 (Dave Albonesi)**
  - GALS processors: power/performance evaluation (cont'd)
- **3:30-4:00 Break**
- **4:00-4:40 (Dave Albonesi)**
  - Workload characterization and impact on the use of fine grain speed/voltage scaling
- **4:40-5:45 (Alper Buyuktosunoglu, Pradip Bose)**
  - Case study - LPX, an IPCMOS based processor
- **5:45-6:00 (Diana Marculescu)**
  - Looking in the crystal ball: Where will partial asynchrony be used?
  - Concluding remarks

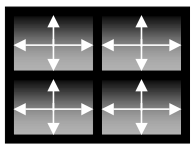
GALS systems: Why and how ?

GALS design benefits

Synchronous



GALS



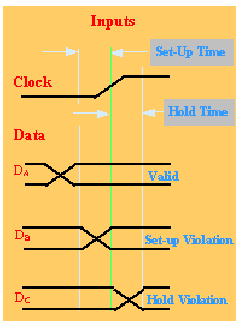
- **Lack of a global clock**
  - Less design effort and cost for dealing with clock skew
  - Less power consumption
- **Potential for fine-grain adaptability**
  - Different speeds among synchronous blocks
  - Different voltages, and hence potential for lower power consumption
  - Can be used with on-the-fly application-driven adaptation

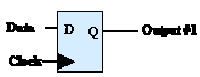
GALS design issues

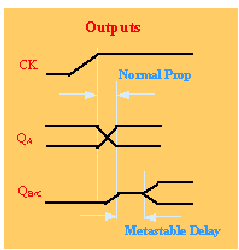
- **Metastability resolution**
  - Always a problem when interfacing clock domains
  - Synchronizers and arbiters
  - Possible solution: mixed-timing FIFOs [Chelcea et al., DAC 2001]
- **Local clock generation**
  - Ring oscillators [Muttersbach et al., ASYNC 2000]
- **Failure modeling**
  - Synchronization failures can happen
  - The goal is to maximize Mean Time Between Failure (MTBF)

Metastability

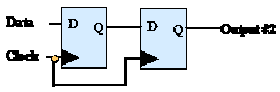
- **Asynchronous signals may violate**
  - Setup time
  - Hold time
  - Both
- **Hence the possibility of a metastable state**

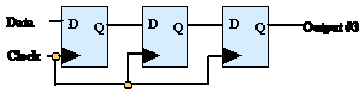




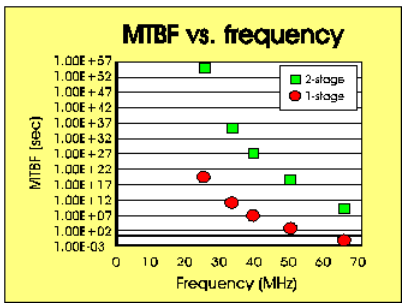


Using synchronizers





- Adding a second FF will reduce the chance of the output going metastable.
- The output from the first flip flop may go valid, before the second flip flop is clocked.
- Adding yet another FF will reduce the probability that its output will be unstable even more.

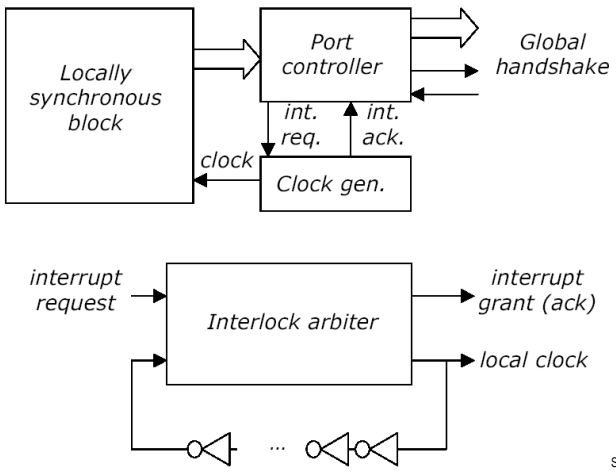


Source: Texas Instruments

Pausible clocks

- Some applications cannot live with a significant failure rate:
  - Multiprocessing
  - Cryptography and security applications
  - Mission critical applications
- Idea for accommodating metastability [Yun et al., ICCD 1996]:
  - Pause the passive phase of a local clock, long enough to allow metastability to resolve. Then the clock can be generated by an interlock arbiter.

Pausible clocks

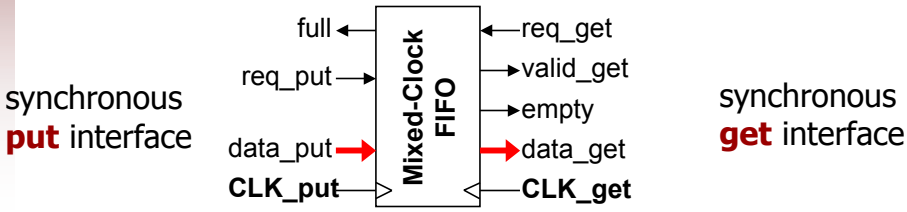


Source: Yun et al., 1996

Mixed timing FIFOs

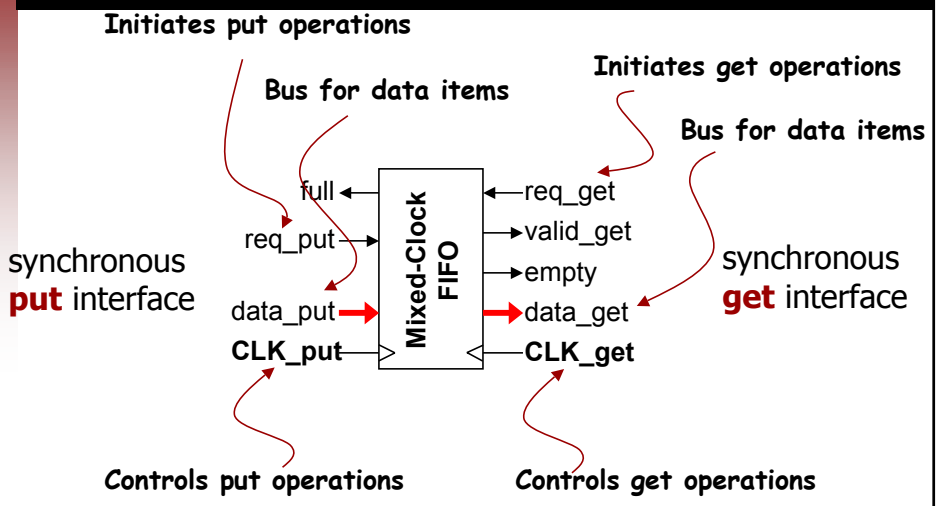
- **Mixed timing or asynchronous FIFOs [Chelcea and Nowick, DAC 2001, WCED 2002]**
  - Low-latency
- **Modular and scalable:**
  - Define interfaces for each combination of Synchronous or Asynchronous domains
  - Combine interfaces to design new async/sync FIFO's
- **High throughput:**
  - In steady state: no synchronization overhead, no failure probability
  - Enqueue/Dequeue data items: one/cycle
- **Low area overheads**
- **Also, solve issue of long interconnect delays between domains**

Mixed-timing FIFOs



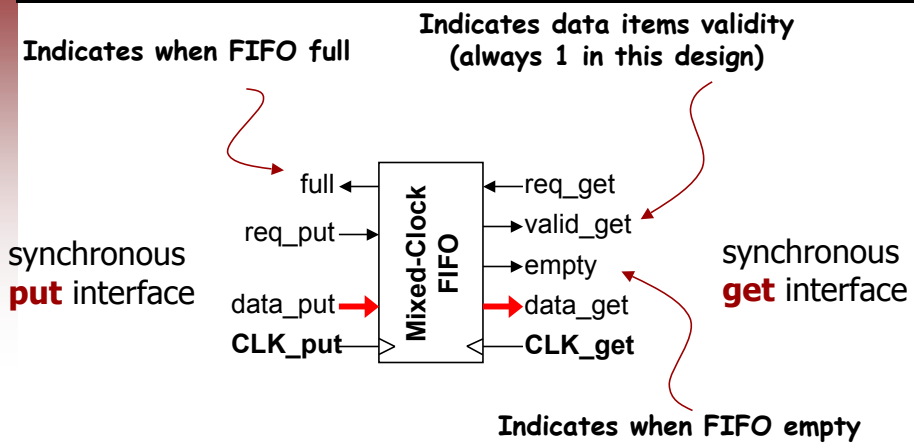
Source: Chelcea and Nowick, 2001, 2002

Mixed-timing FIFOs



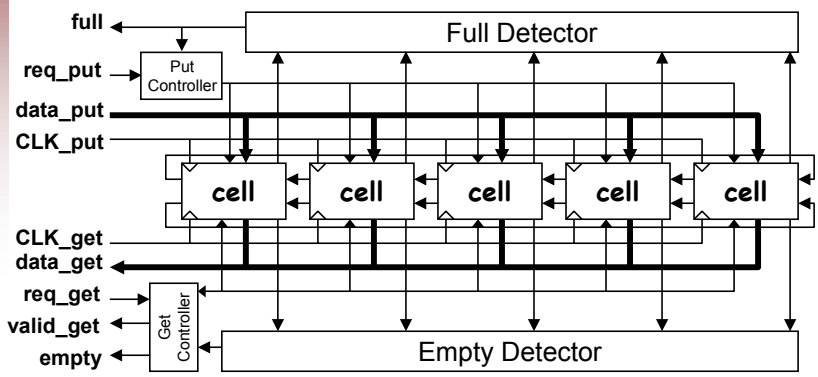
Source: Chelcea and Nowick, 2001, 2002

Mixed-timing FIFOs



Source: Chelcea and Nowick, 2001, 2002

Mixed-timing FIFOs



Source: Chelcea and Nowick, 2001, 2002

Local clock generation

- **Ring oscillators**
  - Pro: Simple
  - Con: Cannot be controlled externally
  - … But they can be coupled with the “local control” of speed and/or voltage
- **Local PLLs**
  - Pro: Externally controlled
  - Con: More costly
  - … but they “scale” [IWLS 2002, Focus Group on GALS]

Schedule

- **2:00-3:05 (Diana Marculescu)**
  - Trends and issues in clock distribution
  - Motivation for GALS, synchronization issues, deadlock prevention, possible inter-clocking domain communication schemes
  - GALS processors: power/performance evaluation
- **3:05-3:30 (Dave Albonesi)**
  - GALS processors: power/performance evaluation (cont’d)
- **3:30-4:00 Break**
- **4:00-4:40 (Dave Albonesi)**
  - Workload characterization and impact on the use of fine grain speed/voltage scaling
- **4:40-5:45 (Alper Buyuktosunoglu, Pradip Bose)**
  - Case study - LPX, an IPCMOS based processor
- **5:45-6:00 (Diana Marculescu)**
  - Looking in the crystal ball: Where will partial asynchrony be used?
  - Concluding remarks

Power-performance evaluation of GALS processors

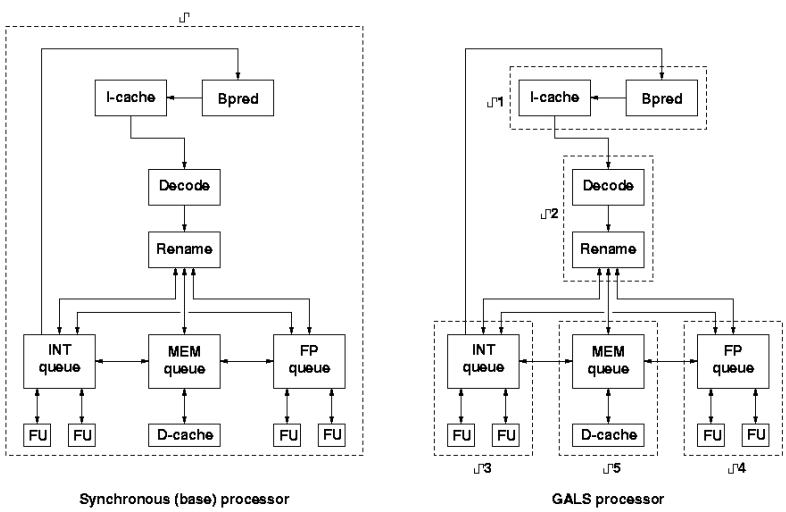
Architecture definition

- **Automatic partitioning into clock domains [Hemani et al., DAC 1999]**
  - Applied only to random logic, not helpful for high-end processors
- **A typical superscalar core exhibits "natural decoupling":**
  - Front-end (I-cache and BP hardware)
  - Decode, Register renaming
  - Integer, FP and memory "domains"
- **Local clock grids usually follow the same type of partitioning**

Architecture definition

- **Starting point:**
  - A hypothetical synchronous machine resembling the Alpha 21264
- **GALS architecture: satisfies physical, as well as architectural constraints**
- **Five clock domains**
  - 1. I-cache, B-pred (FETCH)
  - 2. Decode, rename (DECODE)
  - 3. Integer issue queue, integer units (INT)
  - 4. Memory issue queue, d-cache (MEM)
  - 5. Floating point issue, floating point units (FP)
- **Clock domains are interfaced using asynchronous FIFOs**

Architecture definition

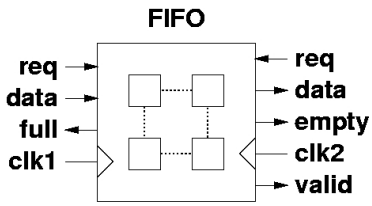


Synchronous (base) processor

GALS processor

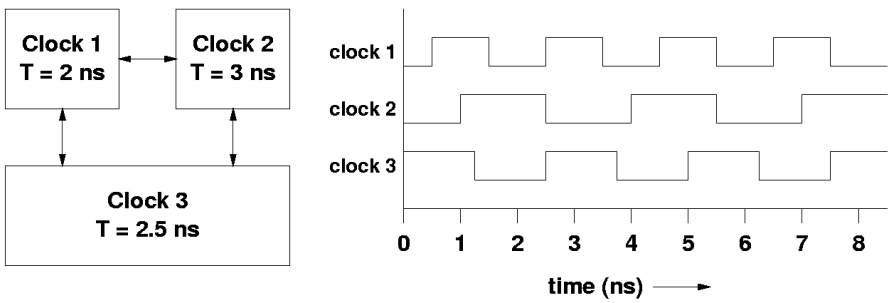
Mixed timing communication

- Used the token-ring based FIFO design
  - [Chelcea et al., DAC 2001]
  - Interfaces two clocks by synchronizing control signals
  - Provides full throughput and low latency in steady state
  - Included cycle-accurate model in our simulation environment



Simulation framework

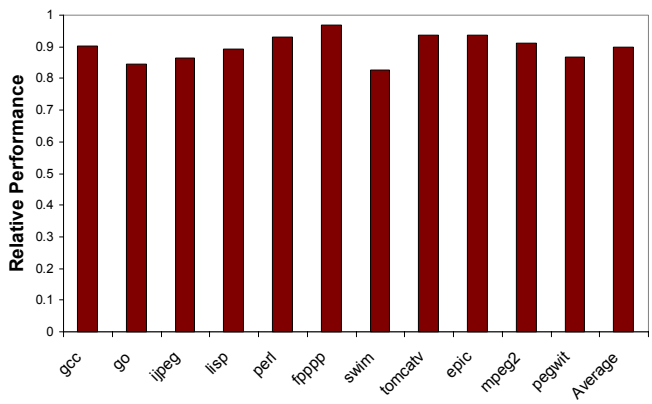
- Event-driven simulation engine: sim-GALS
  - Each clock domain is represented by one set of periodic events
- Reused some of the primitives in SimpleScalar and Wattch



Microarchitecture settings

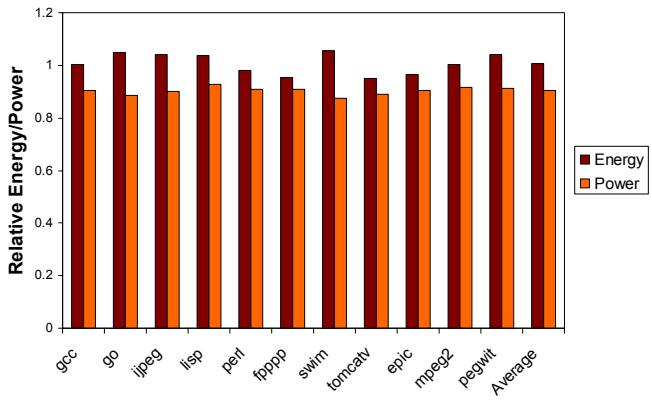
Fetch, decode, commit rate	4 inst/cycle
Integer issue queue size	20
FP issue queue size	16
Memory issue queue size	16
Integer registers	72
FP registers	72
L1 data cache	16KB 4-way 1 cycle latency
L1 instruction cache	16KB direct-mapped 1 cycle latency
L2 unified cache	256KB 4-way 6 cycles latency
ALUs	4 Int, 4 FPs
Branch predictor	Bimodal (SimpleScalar default)
Clock gating	1-2 cycles for restarting the clock

Performance



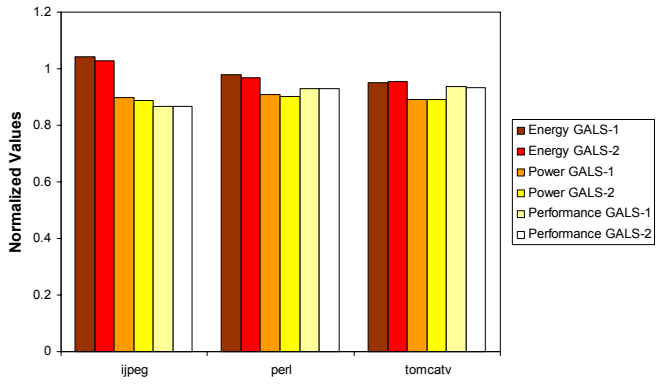
- 10% average performance penalty (3-18%)

Energy and power



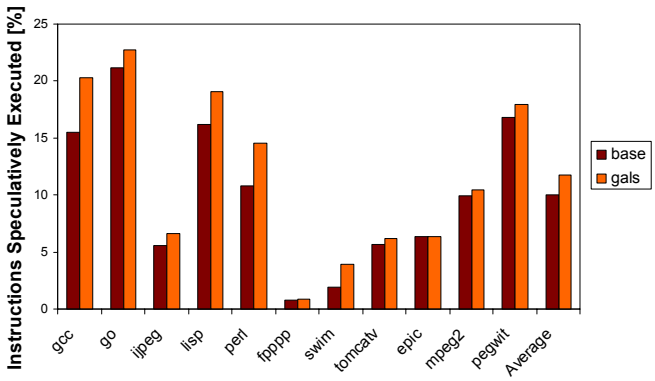
- Elimination of global clock does not necessarily reduce total energy consumption
- Average power consumption is reduced (10% on average)

Impact of clock stopping overhead



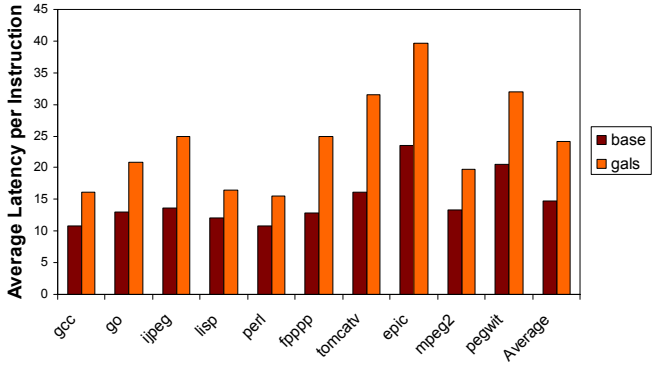
- Shutdown a clock domain after 4 idle cycles
- Restart it when decode stage detects an instruction which “needs” that domain
- Minimal impact of clock stopping/restarting overhead

Speculative execution



- On average, 2% more instructions are speculatively executed
- In the worst case, 5% more instructions are speculatively executed

Average latency per instruction

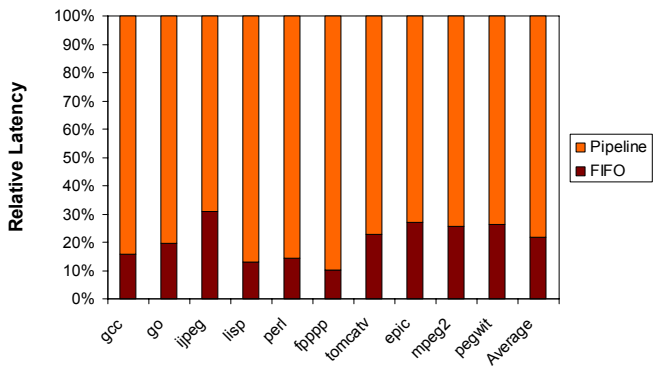


- End-to-end latency increases from 15 to 24 cycles, on average

Energy and performance trends

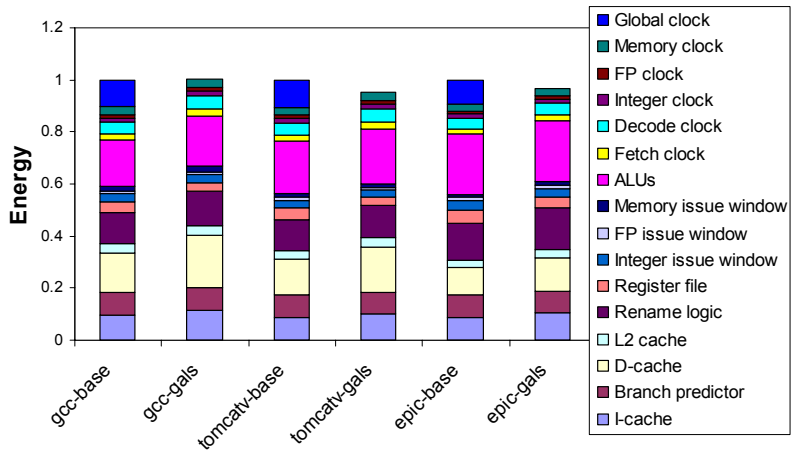
- **Average power decreases, but overall energy may increase**
- **Reasons:**
  - Performance drops due to asynchronous communication
    - Hence longer execution time and more overhead for unused modules
  - Longer branch recovery pipeline
    - Consequently, more speculative execution
  - Higher fetch to commit time for each instruction
  - Higher occupancies of rename tables and issue queues

Where does performance go?



- **Asynchronous FIFOs introduce 22% latency overhead on average**
- **The rest of latency is spent in other stages of the pipeline**

Where does energy go?



- **Longer execution times translate into larger energy costs**

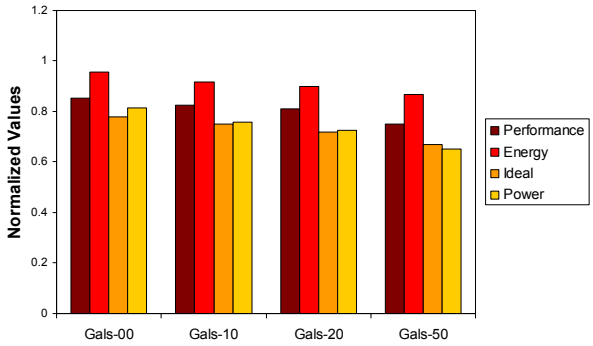
Delay - voltage dependency

- **Fine-grain voltage reduction can be beneficial in GALS systems**
  - Each clock domain can be run at a different speed

$$D \propto \frac{V_{dd}}{(V_{dd} - V_t)^{\alpha}}$$

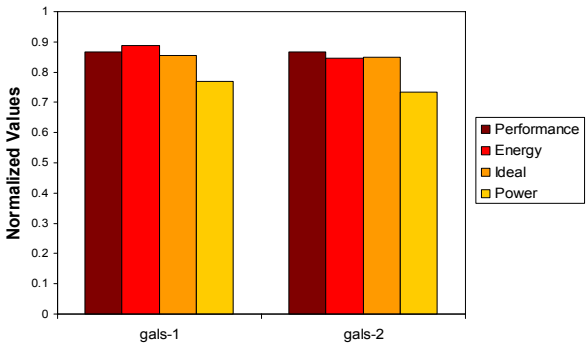
  - V_{dd} is the supply voltage
  - V_t is the threshold voltage
  - α is a technology-defined constant between 1 and 2;
  - α is 1.2-1.6 for present generation [Chen et al, JSSC 1998]

Selective voltage reduction for ijpeg



- **GALS version gives some energy savings**
  - Fetch clock slowed down 10%
  - FP clock slowed down 20%
  - Memory clock slowed down 0-10-20-50%
  - Ideal: base, fully synchronous version with reduced voltage

Selective voltage reduction for gcc



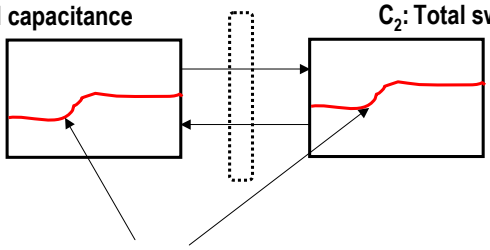
- **1: FP clock: 1.5; 2: FP clock: 3.0**
- **GALS version is as good as the fully synchronous baseline**

Interesting theoretical result

- **For a linear pipeline without feedback:**
  - Under the same performance constraints, the multiple clock, multiple voltage core is better than the single clock, single voltage core **only if** the lower voltage is applied to the clock domain with:
    - $C_{domain}/L_{domain} > C_{total}/L_{total}$
    - C_d is the total switched capacitance of domain d, L_d is the total load on the critical path of domain d.

C_1: Total switched capacitance for module 1

C_2: Total switched capacitance for module 2

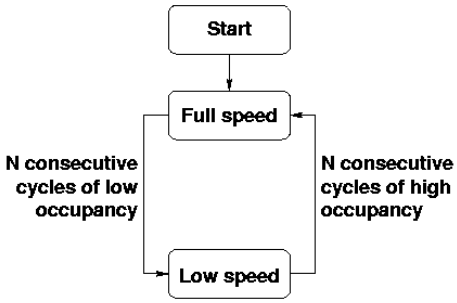


Critical paths, with total loads L_1, L_2 respectively

Implications?

- **Intuitively:**
  - To achieve power savings, the multiple clock, multiple voltage core must use the **lower** voltage on the clock domain that has the **highest** switched capacitance and contributes the **least** to the overall end-to-end latency.
  - …In other words: make the power consuming, non-critical clock domains run slower

A possible solution: Dynamic control strategy

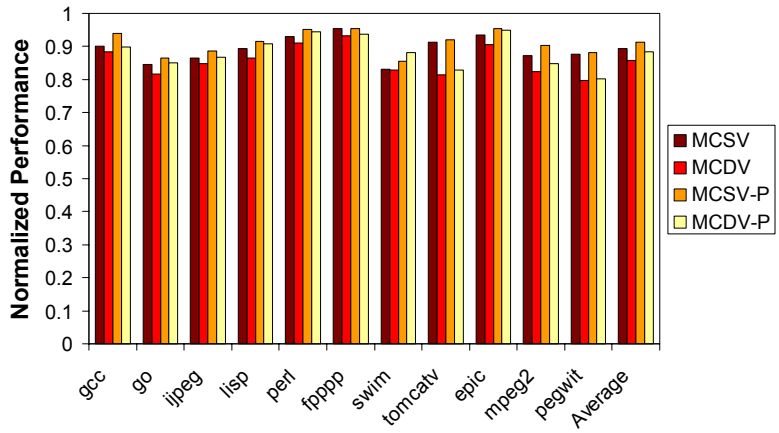


- **Applied for the Int, FP, and Mem clock domains**
- **Occupancy of issue queues monitored**
  - If lower than a given threshold, switch to lower voltage
  - If higher than a given threshold, switch to higher voltage

Microarchitecture settings

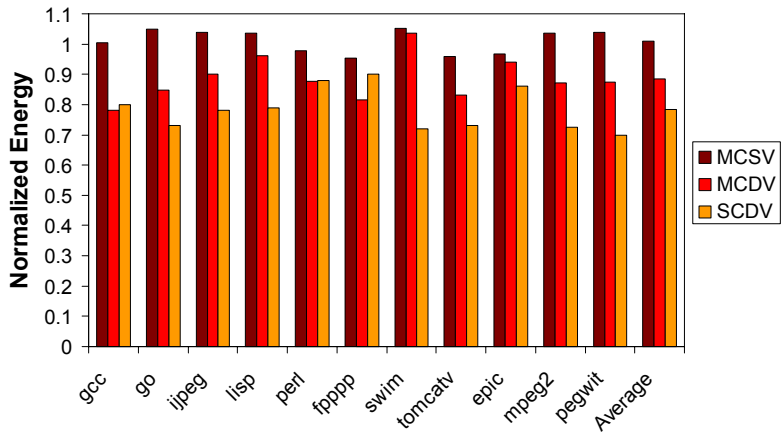
- **Same as before, plus…**
  - Occupancy monitoring — 2K cyc.(before deciding to switch)
  - Integer issue queue thresholds — 7/9
  - FP issue queue thresholds — 3/5
  - Memory issue queue thresholds — 5/8
  - Threshold for “stretching” clocks — 30% of the smallest clk period
  - Aggressive mode — FP 2.4X, Mem 1.2X slowdown
  - Moderate mode slowdown — Int 1.8X, FP 1.2X, Mem 1.2X
- **Stretchable clock-based FIFOs have been introduced in [Semeraro et al. HPCA 2002]**
  - If rising edges are within a given threshold (30% above), then stall the producer domain for another cycle

Performance



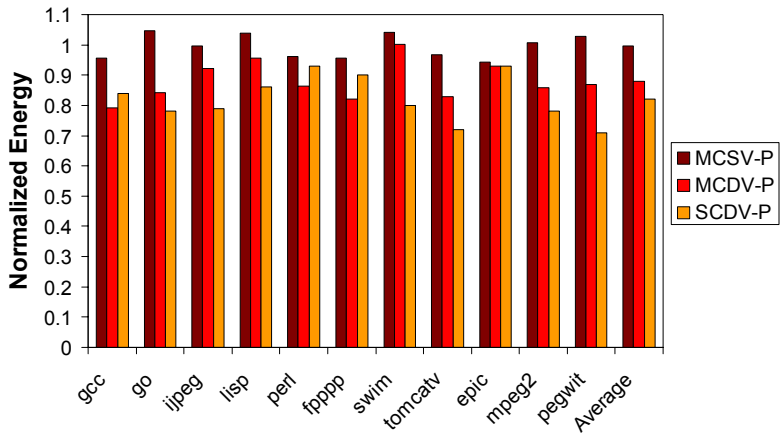
- MCSV (simple GALS – Multiple Clock, Single Voltage) and MCDV (GALS with fine grain DVS – Multiple Clock, Dynamic Voltage) compared to the fully synchronous baseline
- -P is the case of stoppable clocks – less performance penalty overall

Energy (FIFOs with synchronizers)



- *Moderate* mode was used for *swim, fpppp, epic* (*aggressive* mode for the rest)
- SCDV (single clock, DVS case) fares 5% better on average, but worse in some cases (*gcc, perl, fpppp*)

Energy (FIFOs with stretchable clocks)



- SCDV (single clock, DVS case) fares 3% better on average, but worse for *gcc, perl, fpppp, epic*

More information…

- CMU's Energy Aware Computing Group
  - http://www.ece.cmu.edu/~enyac

Energy Aware Computing Research Group

Related reading list

- D. M. Chapiro, Globally Asynchronous Locally Synchronous Systems. PhD thesis, Stanford University, 1984.
- Ivan E. Sutherland. Micropipelines. (Turing award lecture) Communications of the ACM, 32(6):720-738, June 1989.
- R.F. Sproull, I.E. Sutherland, and C.E. Molnar. The counterflow pipeline processor architecture. IEEE Design & Test of Computers, 11(3):48-59, Fall 1994.
- P. Zarkesh-Ha, T. Mule, J. Meindl. Characterization and Modeling of Clock Skew with Process Variations. In Proc. Custom Integrated Circuits Conference (CICC), 1999.
- Y. Liu, S.R. Nassif, L.T. Pileggi, A.J. Strojwas. Impact of interconnect variations on the clock skew of a gigahertz microprocessor. In Proc. Design Automation Conference (DAC), June 2000.
- K.Y. Yun, R.P. Donohue. Pausible Clocking: A First Step Toward Heterogeneous Systems. Computer Design: In Proc. Intl. Conference on VLSI in Computers and Processors (ICCD), 1996, pages: 118-123.
- A. Hemani, T. Meincke, S. Kumar, A. Postula, T. Olsson, P. Nilsson, J. Oberg, P. Ellervee, and D. Lundqvist. Lower Power Consumption in Clock By Using Globally Asynchronous Locally Synchronous Design Style. In Proc. Design Automation Conference (DAC), 1999.
- J. Muttersbach, T. Villiger, and W. Fitchner. Practical Design of Globally Asynchronous Locally Synchronous Systems. In Proc. Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), 2000.
- I.E. Sutherland, S. Fairbanks. GasP - A Minimal FIFO Control. In Proc. Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), 2001.
- T. Chelcea and S. M. Nowick. Robust Interfaces for Mixed-Timing Systems with Application to Latency-Insensitive Protocols. In Proc. Design Automation Conference (DAC), 2001.
- A. Iyer and D. Marculescu. Power and Performance Evaluation of Globally Asynchronous, Locally Synchronous Processors. In Proc. Intl. Symposium on Computer Architecture (ISCA), 2002, pages: 158-168.
- A. Iyer and D. Marculescu. Power Efficiency of Voltage Scaling in Multiple Clock,Multiple Voltage Cores. In Proc. Intl. Conference on Computer-Aided Design (ICCAD), 2002, to appear.

Schedule

- **2:00-3:05 (Diana Marculescu)**
  - Trends and issues in clock distribution
  - Motivation for GALS, synchronization issues, deadlock prevention, possible inter-clocking domain communication schemes
  - GALS processors: power/performance evaluation
- **3:05-3:30 (Dave Albonesi)**
  - GALS processors: power/performance evaluation (cont'd)
- **3:30-4:00 Break**
- **4:00-4:40 (Dave Albonesi)**
  - Workload characterization and impact on the use of fine grain speed/voltage scaling
- **4:40-5:45 (Alper Buyuktosunoglu, Pradip Bose)**
  - Case study - LPX, an IPCMOS based processor
- **5:45-6:00 (Diana Marculescu)**
  - Looking in the crystal ball: Where will partial asynchrony be used?
  - Concluding remarks