Designing Networks on Chip: Solutions and Challenges

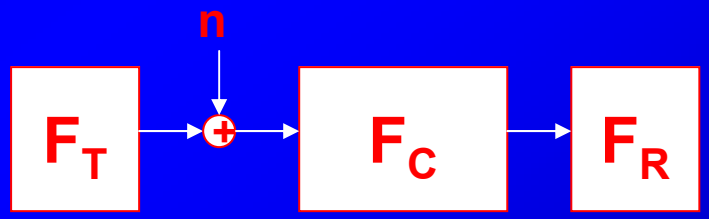
Luca Benini DEIS – Universita` di Bologna

Designing a micro-network

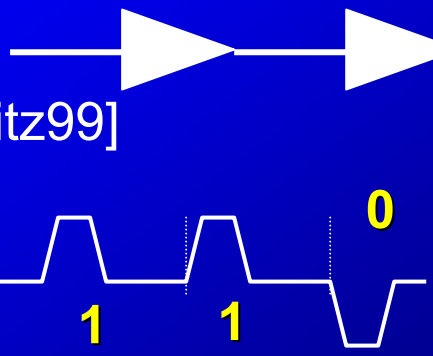
- Physical layer
  - signalling
  - synchronization
- Architecture and control
  - network topology
  - data flow: packetization, encoding
  - control flow: media access, switching, routing
- Software
  - communication API: implicit vs. explicit
  - run-time management

Physical layer: the channel

- Channel characteristics
  - Global wires: lumped → distributed models
    - Time of flight is non-negligible
  - Inductance effects
    - Refelections, matching issues



- Designing around the channel's transfer function
  - Current mode vs. voltage mode [Dally98,Burleson01]
    - Low swing vs. rail-to-rail
  - Repeater insertion [Friedman01,Burleson02]
  - Wire sizing [Cong01,Alpert01]
  - Pre-emphasis / post-filtering [Horowitz99]
  - Modulation [Dally98,Bogliolo01]

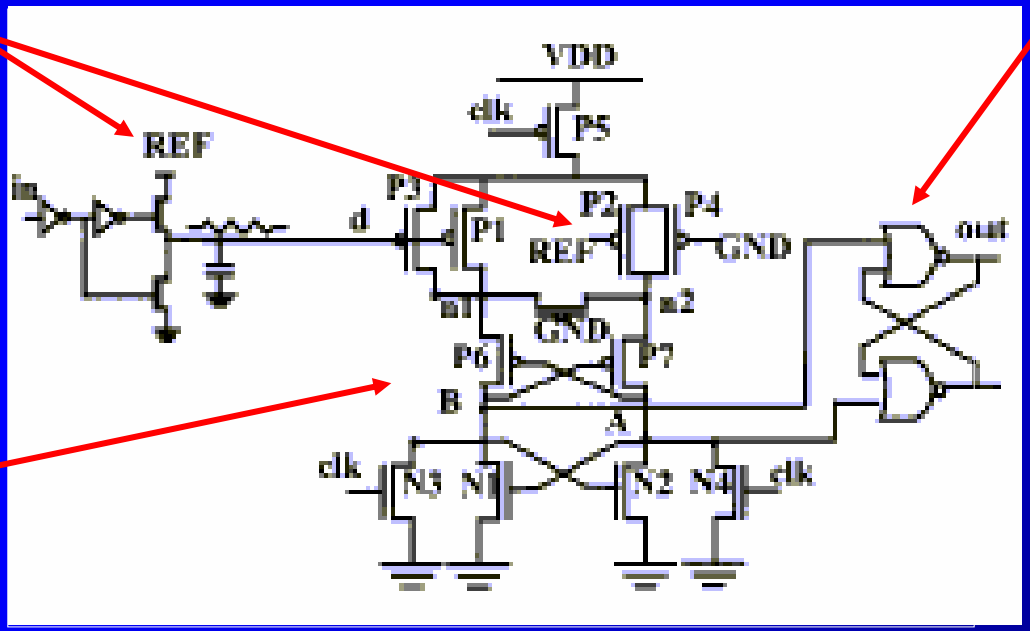


Case study: Low Swing signalling

- Pseudodifferential interconnect [Zhang et al., JSSC00] (x6 energy reduction vs. CMOS Vdd=2V)

Low Vdd
eference (0.5V)

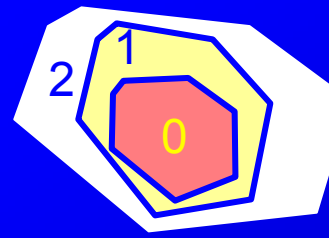
Static FF

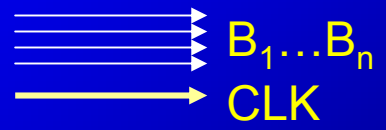


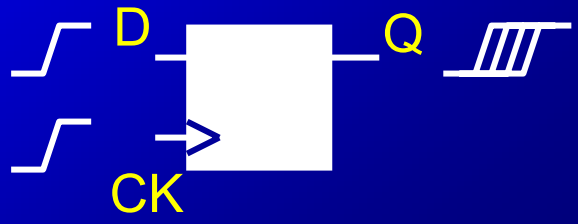
Clocked SA

Physical layer: synchronization

- Single, global timing reference is not realistic
  - Direct consequence of non-negligible *tof*
  - Isochronous regions are shrinking
- *How and when to sample the channel?*
  - Avoid a clock: asyncronous communication
  - The clock travels with the data
  - The clock can be reconstructed from the data
- Synchronization recovery has a cost
  - Cannot be abstracted away
  - Can cause errors (e.g., metastability)

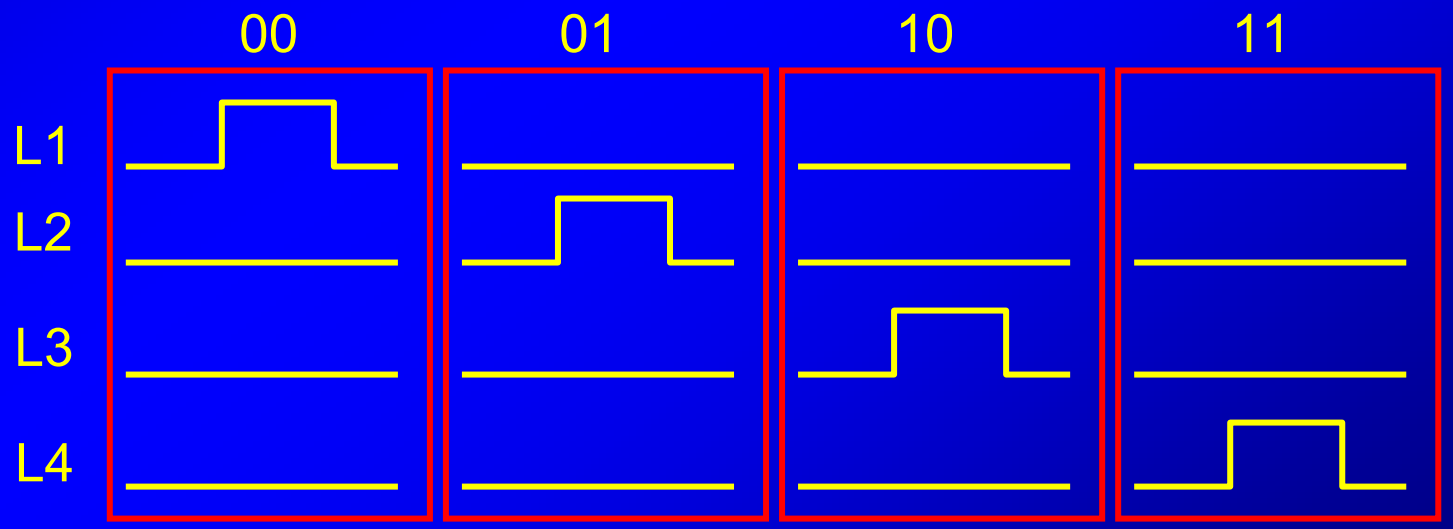






Case study: Asyncronous Bus

- MARBLE SoC Bus [Bainbridge et al. Asynch01)
  - 1-of-4 encoding (4 wires for 2 bits)
  - Delay insensitive - No bus clock - Wire pipelining
  - High crosstalk immunity
  - Four-phase ACK protocol



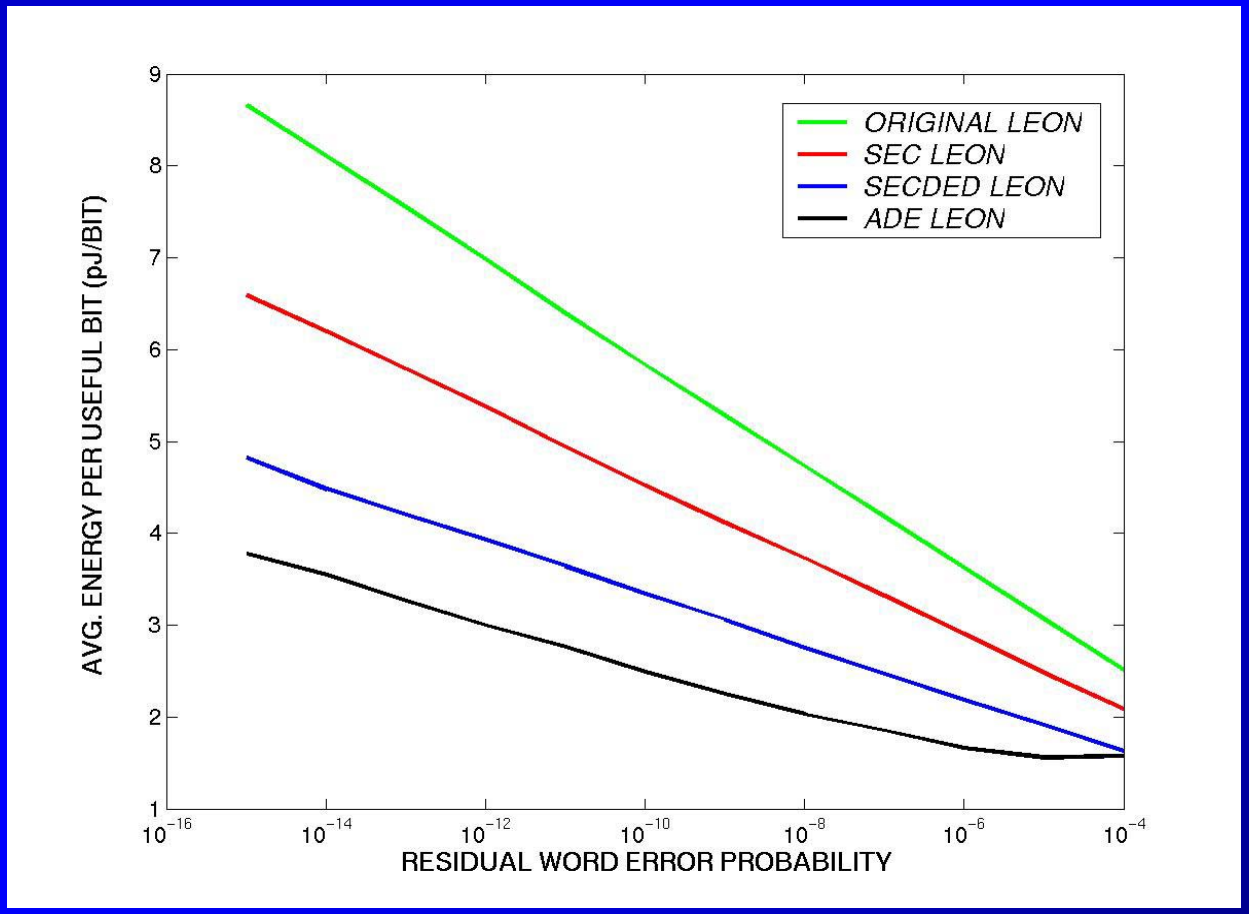
Physical layer: multiobjective optimization

- Communication is *unreliable:*
  - Crosstalk, supply noise, synchronization noise $\Rightarrow P_{bitflip} > 0$
- S/N minimization via S maximization is highly suboptimal (energy-wise)
- High performance decreases reliability
  - Shorter eye opening
- Wire redundancy helps
  - But consumes wiring resources

Multiobjective design space:
energy vs. performance vs. reliability tradeoffs

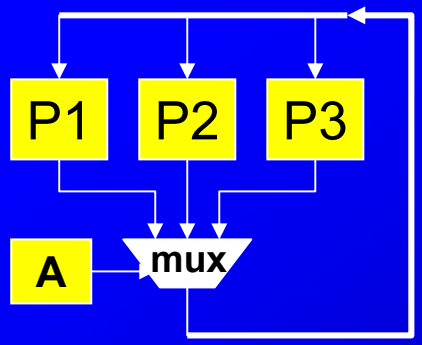
Case study: EC vs. ED codes

- Low swing signalling with redundant codes [Bertozzi et al. DATE02]: exploring energy vs. error rate tradeoff



NoC Architecture: topology

- Point-to-point vs. shared medium
  - Shared mediun: On-chip bus
    - Dominant today (e.g. AMBA, CoreConnect, etc.)
    - Unidirectional (vs. off-chip three state)
    - Bridged (high speed vs. peripherals)
    - Performance/Power bottleneck
  - Point-to-point: dedicated links
    - Ad-hoc width
    - Ad-hoc control
    - Wiring bottleneck
- **Towards multi-stage networks**
  - Hierarchical+eterogeneous, e.g. Maia [Rabaey00]
  - Omogeneous e.g. FPGAs, Network processors, …

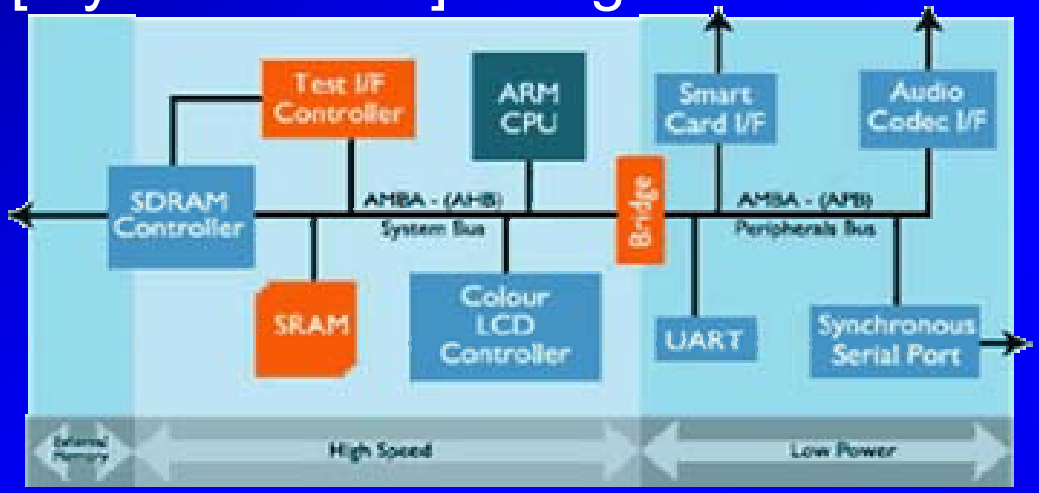


Topology optimization

- One size *does not* fit all...
  - Low-area, low performance systems
    - Shared medium (on-chip bus)
  - General-purpose, high performance
    - Omogenenous multi-stage networks
  - Domain-specific, low power
    - Eterogeneous multi-stage networks
- EDA support
  - Physical design (floorplanning, routing, layer assignment)
    - Eterogeneous solution requires strongest EDA support
  - IP-based approach (VSI: topology-neutral)

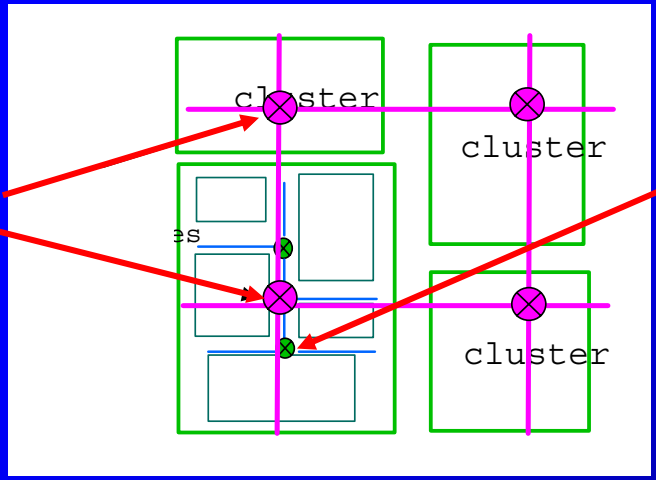
Case study: hierarchical networks

- AMBA [Flynn Micro97]: bridged bus architecture



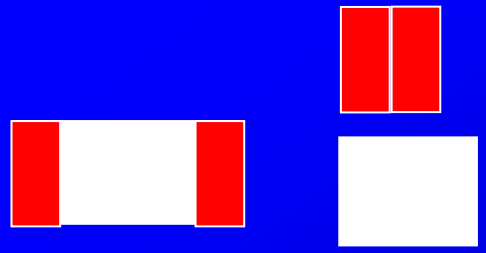
- Hierarchical Mesh [Zhang et al. JSSC00]

Hierarchial switch-boxes



Universal (intra-cluster) switch-boxes

NoC control: data flow



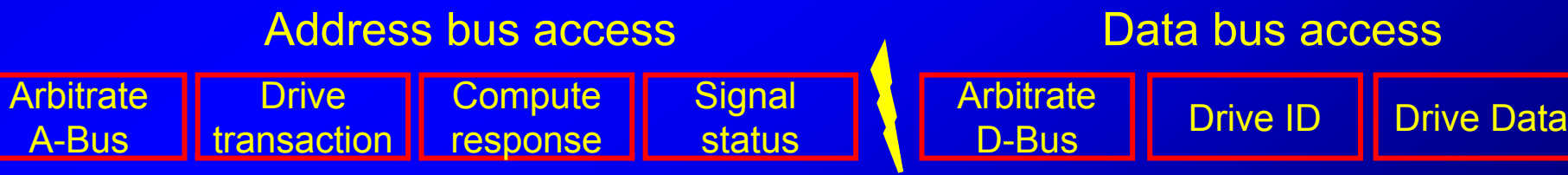
- Packetization
  - Payload: single-word vs. multi-word
    - E.g. burst transactions in AMBA
  - Header-tail: in packet vs. dedicated channels
    - E.g. SPIN (in-packet) [Guerrier00] vs. AMBA (control signals)
  - Acknowledgement: blocking vs. non blocking
    - E.g. Split transaction bus in Daytona [Ackland00]
- Data representation/encoding
  - Fast hardware-based compression [Benini01]
  - Encoding for low energy/error resiliency […]

NoC data-flow optimization

- Packet size/format optimization
  - Payload vs. control
    - Lager payload⇒reduce control overhead
    - Smaller payload⇒improved error recovery
  - Dedicated control channels vs. in-packet
    - Control wires overhead (long and slow)
    - Smaller payload (reduced effective bandwidth)
    - Forward (data) and backward (ack) traffic
- Data representation
  - Payload/address compression, low power encodings
    - Compression-decompression cost (performance/power)

Case study: STBus

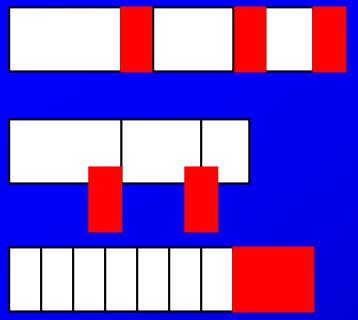
- Daytona split transaction bus [Ackland JSSC00]
  - Pipelined 128b Data, 32b Address
  - Multiple outstanding transactions (8b transaction ID)
- Variable packet size (1 - 128 B)
- Multiple types of transactions
  - Explicit data transfers (e.g., IO): RD, WR
  - Cache coherency (modified MESI write-invalidate, snoopy)
- Four priority levels with RR: Instr, Data, Touch, DMA



NoC control flow

- Shared medium accessn⇒TDMA
  - Bus arbitration (e.g., AMBA)
  - Slot reservation (e.g., SiliconBackplane)
- Switching & Routing (multi-stage NoCs)
  - Access
  - Switching
  - Routing

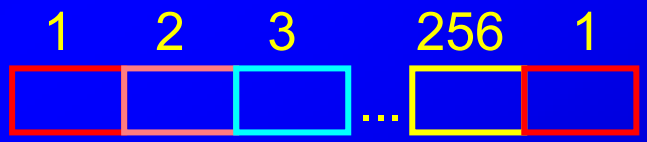
NoC control flow optimization



- Shared-medium protocol optimization
  - Define bus priorities [Lahiri01]
  - Decentralized/pipelined arbitration [Sonics]
  - Slotted access window assignment [Lahiri01]
- Multistage networks
  - Static routing, circuit swiching ⇒ FPGAs
  - Dynamic routing, circuit switching ⇒DPGAs
  - Static routing, packet switching
    - Burst-level switching (virtual circuit)
    - Single packet switching ⇒ STM Octagon [Dey01]
    - Cut-through switching ⇒ SPIN
  - Dynamic routing, packet switching (not yet)

Case study: Slot reservation

- Sonic µNetwork [Wingard DAC01]
  - Two-level arbitration mechanism
- First level: TDMA
  - Time wheel of 256 frames
  - Each frame can be pre-allocated to one initiator
- Second level: Round Robin
  - Only in idle reserved frames or unreserved frames
  - Token passing mechanism (distributed protocol)
- Use first level for regular, heavy traffic sources
- Use second level for sporadic, light traffic sources

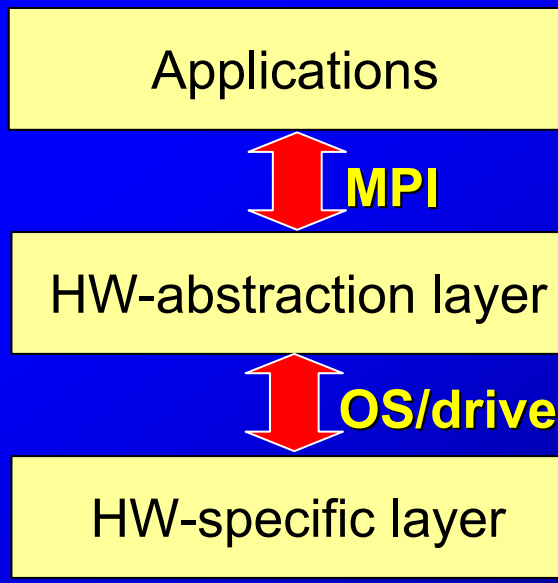


Programming for NoCs

- The programmer's model
  - *Implicit communication*: a single-thread application, communication is to-from memory
  - *Explicit communication*: multiple threads/tasks, communication and synchronization are either fully explicit (message passing) or partially explicit (shared variables)
- Parallelism extraction vs. parallelism support

Explicit communication

- Explicitly parallel programming styles
  - Implicit communication (memory traffic) still relevant
  - Explicit communication (inter-process)
- APIs for explicit communication
  - From multiprocessors (e.g. MPI, pthreads)
  - Support for eterogeneous network fabrics
- Parallel programming API as HW abstraction layers
  - How much abstraction do we need?

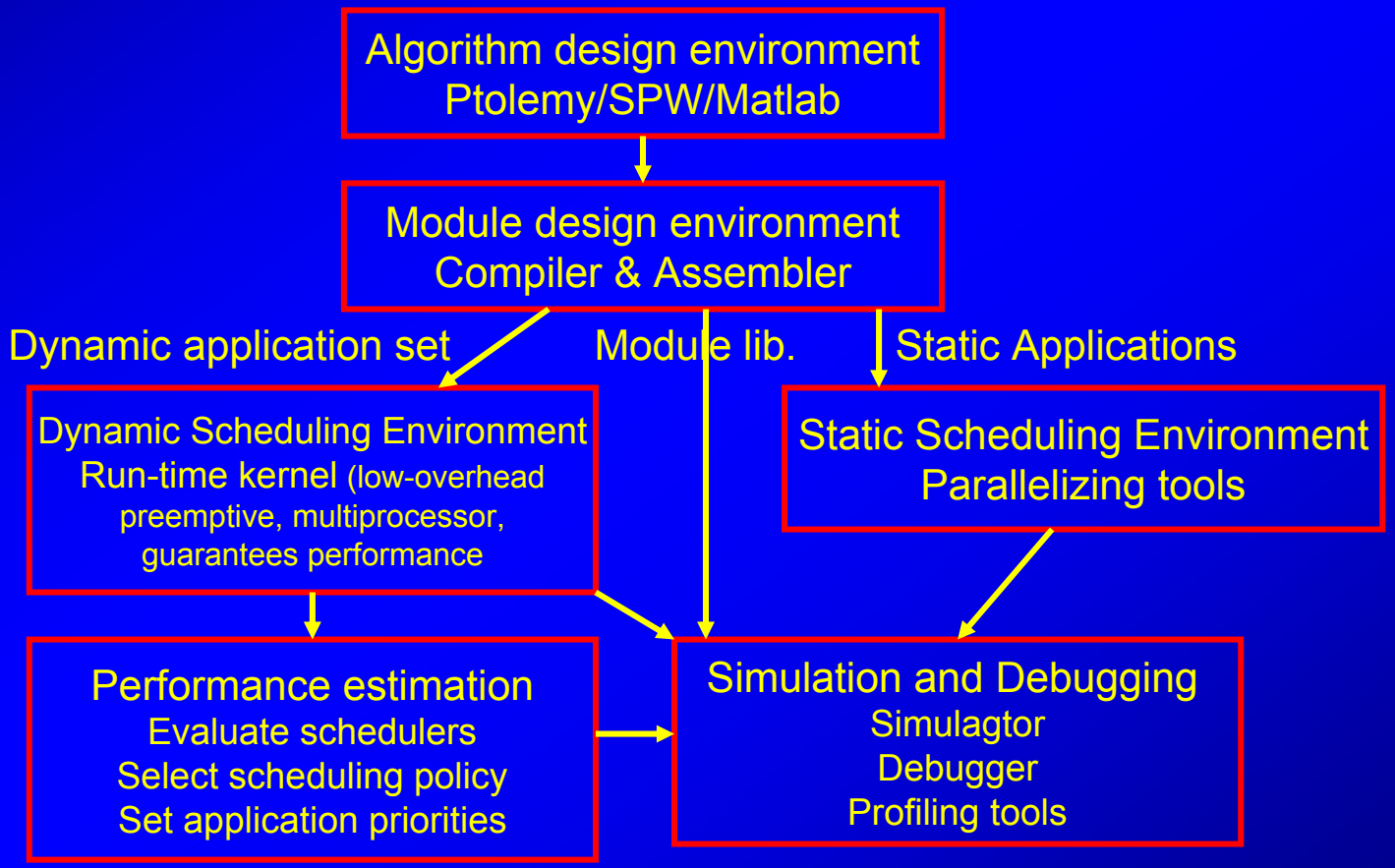


Run-time infrastructure

- Traditional RTOSes
  - Single-processor master
  - Limited support for complex memory hierarchies
  - Focused on performance
- The NoC OS
  - Natively parallel
  - Supports eterogeneous memory, computation, communication
  - Energy/power aware

Case study: MPDSP SDE

- Daytona SDE [Kalawade DAC99]
  - Software design methodology and tools

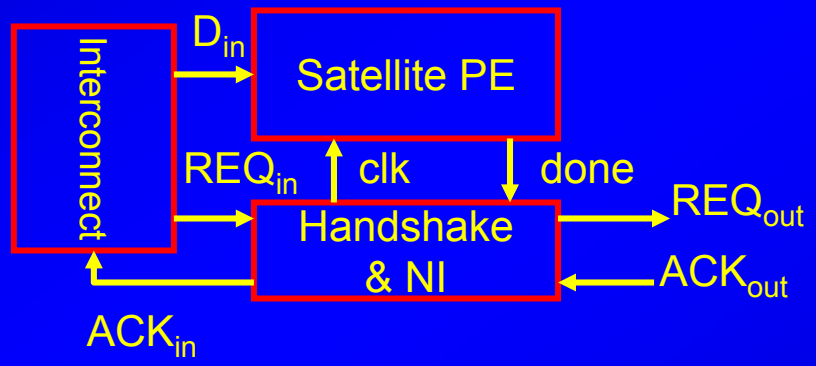


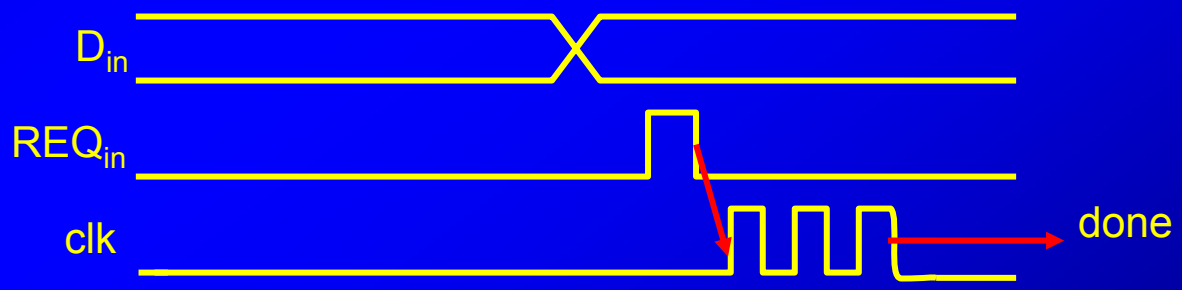
Managing system energy

- Power is a primary constraint
- Hardware support for energy efficiency
  - Multiple shutdown states (idle, sleep, etc.)
  - Variable/multiple clock speed
  - Variable voltage
- The OS should manage the degrees of freedom
  - Dynamic power management policies
- In NoCs⇒distributed control issue
  - Multi-server systems
  - Interaction with application layer

Case study: node DPM

- Maia processor [Zhang JSSC00]
  - On-demand node activation (GALS)





Summary

- Trend toward NoCs
  - Physics/technology drives us there
- A methodology to design/use NoCs
  - A layered approach
- Some solutions are already out there
  - EDA support is essential
  - Software infrastructure is key