Energy-Aware Networked Multimedia Systems: Modeling, Analysis and Optimization

Prof. Radu Marculescu
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
http://www.ece.cmu.edu/~sld/

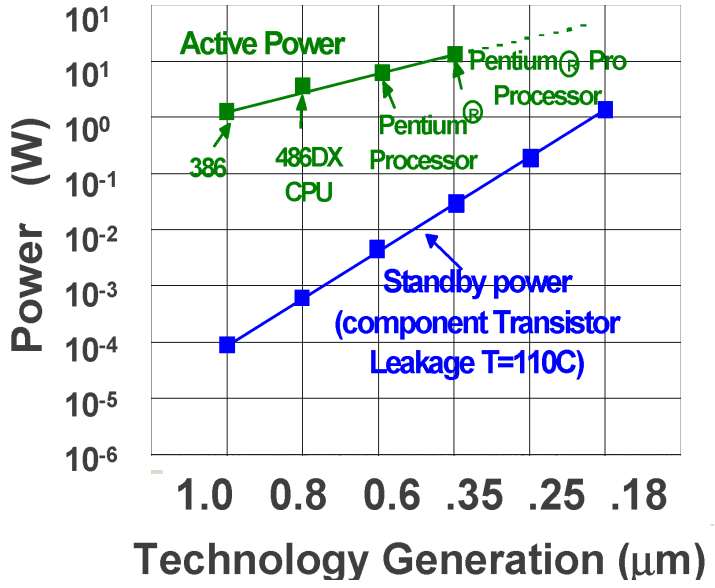
Prof. Massoud Pedram
Dept. of Electrical Engineering
University of Southern California
http://apollo.usc.edu/testbed/

January 21, 2003

Outline

- **Part I**
  - **Sources of Power Dissipation**
  - **Dynamic Power Management**
  - **Runtime Mechanisms for Leakage Control**
- **Part II**
  - **Adaptive Voltage and Frequency Scaling**
  - **Energy-aware MPEG-4 FGS Video Streaming**
  - **Power-aware Source Routing in MANETs**
- **Part III**
- **Part IV**

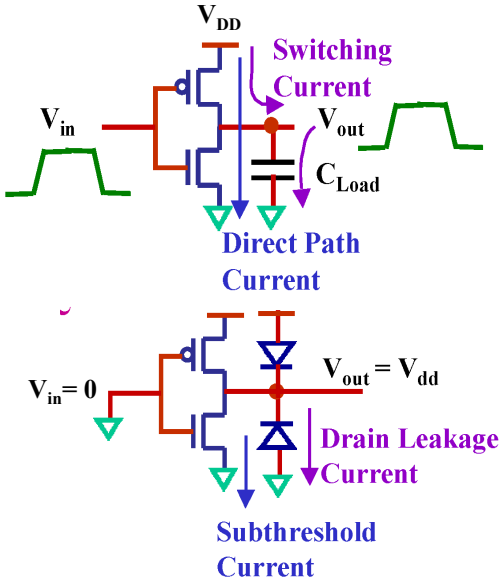
Overview - Power consumption trends



M. Pedram

Power Dissipation Sources 101

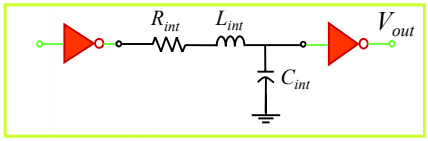
- Dynamic power
  - Switching current
    - Still the dominant source of power consumption in many designs
  - Short-circuit current
    - Less than 15% of dynamic power for properly designed circuits
- Leakage power
  - Sub-threshold leakage current
    - Has become quite important with technology scaling
  - Gate leakage current
    - Is becoming important with shrinking device dimensions
  - PN junction leakage current
    - Remains negligible

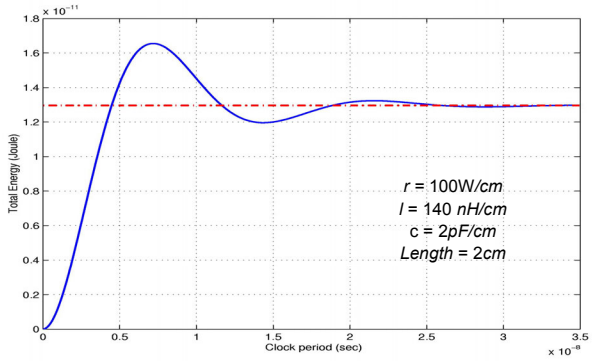


$$P = 0.5CV_{DD}^{2} fN + Q_{SC}V_{DD}fN + I_{leak}V_{DD}$$

M. Pedram

Capacitive Power Dissipation in Interconnect

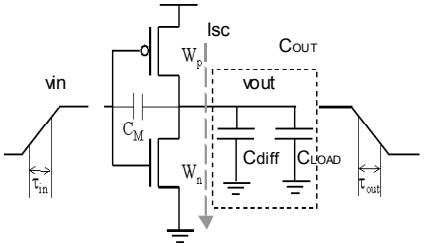


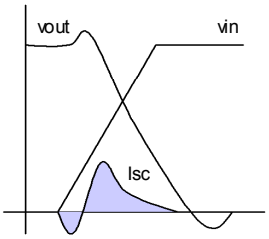


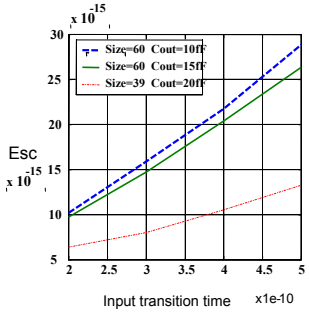
Energy Dissipation of Underdamped RLC Circuits

M. Pedram

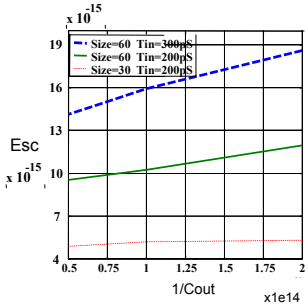
Short Circuit Power



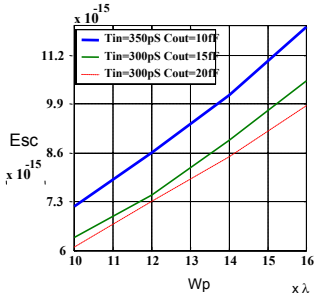




(a)



(b)

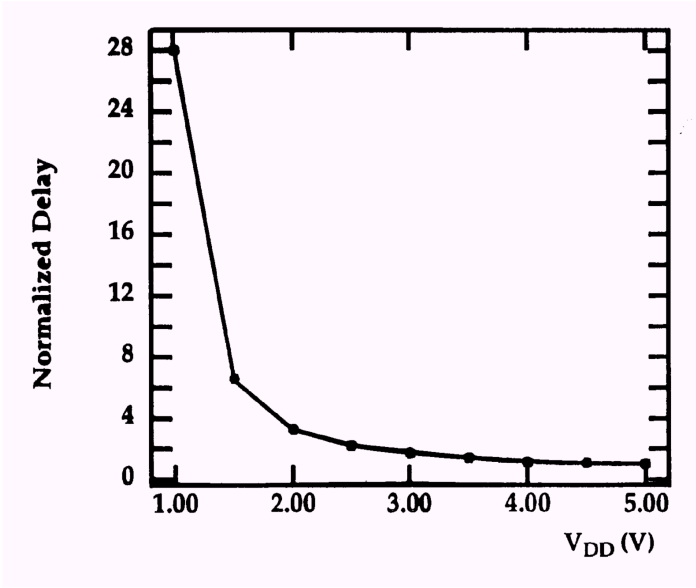


(c)

$$E_{sc}(\tau_{in}, W, C_{out}) = \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} m_{ijk} \frac{W^i \tau_{in}^{\ j}}{C_{out}^{\ k}} V_{DD}$$

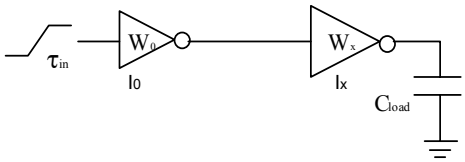
M. Pedram

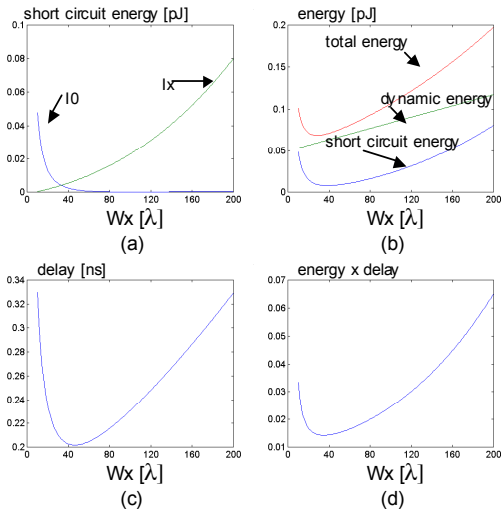
Delay vs. V_{DD} Plot

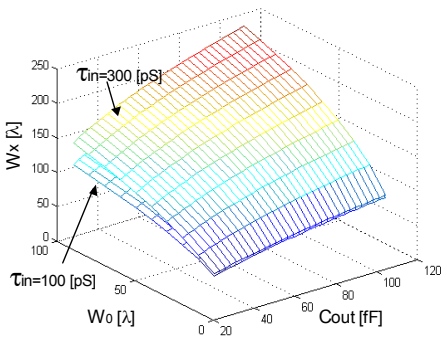


M. Pedram

Minimum Energy-Delay Product







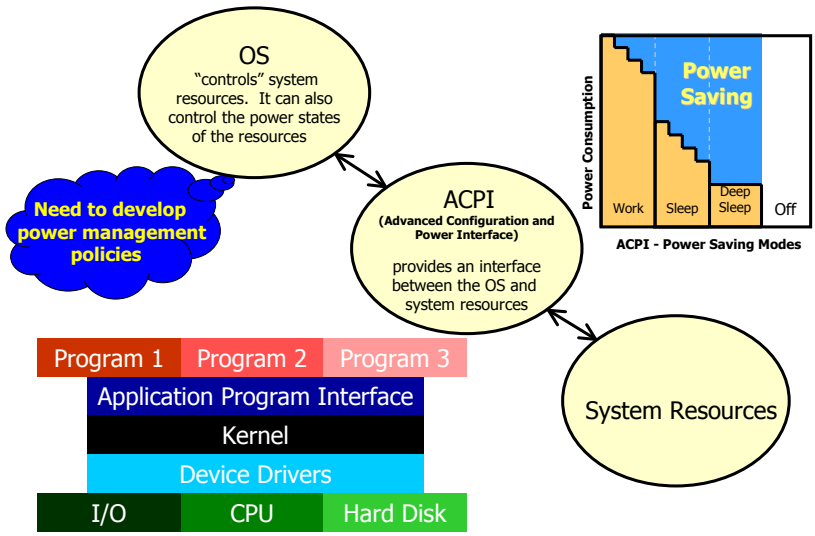
Optimal W_x for minimum energy-delay product with t_{in}, c_{out}, and W_0

M. Pedram

Outline

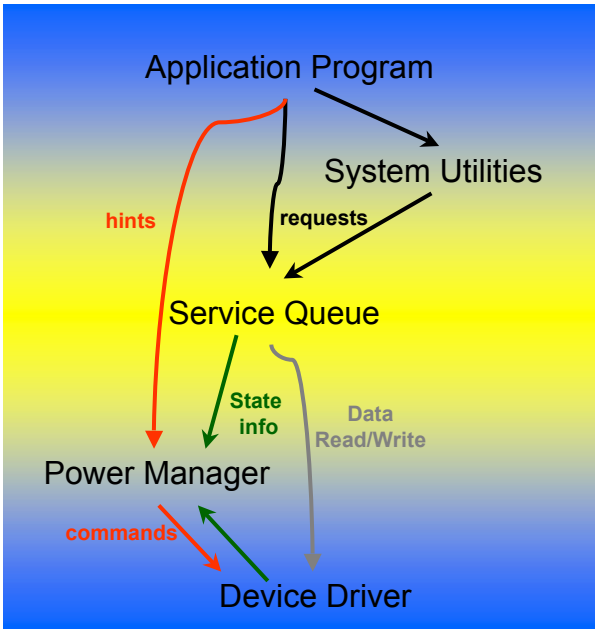
- **Part I**
  - Sources of Power Dissipation
  - Dynamic Power Management
  - Runtime Mechanisms for Leakage Control
- **Part II**
  - Adaptive Voltage and Frequency Scaling
  - Energy-aware MPEG-4 FGS Video Streaming
  - Power-aware Source Routing in MANETs
- **Part III**
- **Part IV**

Dynamic Power Management 101



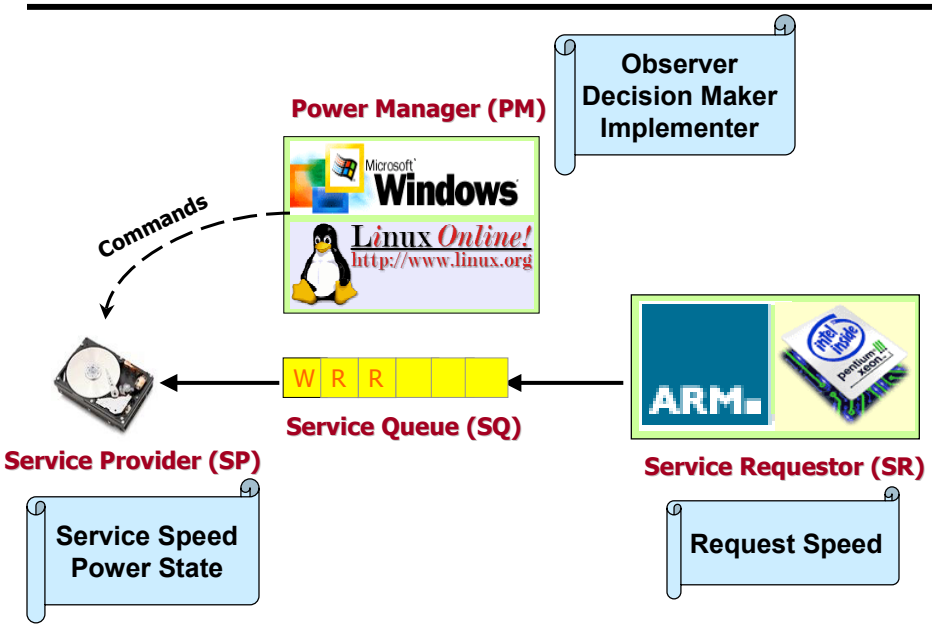
Policy Optimization Flow

- Power management policy optimization flow:
  - Build stochastic models of the service requesters (SRs) and service providers (SPs)
  - Construct a continuous-time Markovian decision process model (CTMDP) of the power-managed system
  - Obtain stochastic parameter values by static profiling and runtime monitoring
  - Solve the resulting power optimization problem under performance and quality-of-service constraints to obtain the optimal policy



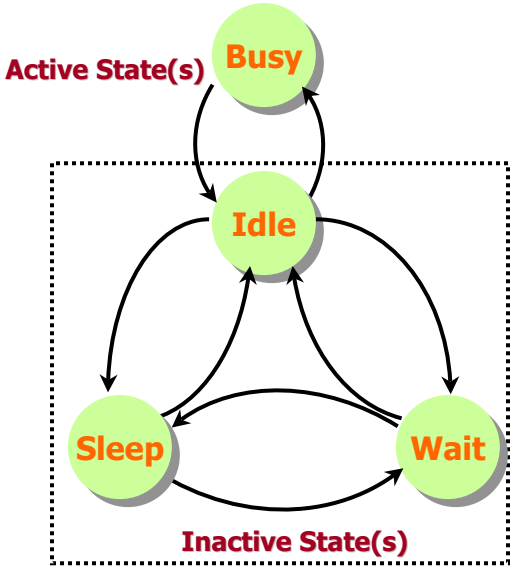
M. Pedram

Stochastic System Modeling



M. Pedram

Service Provider (SP)



- The service provider is modeled as a stationary, controllable, continuous time Markov process

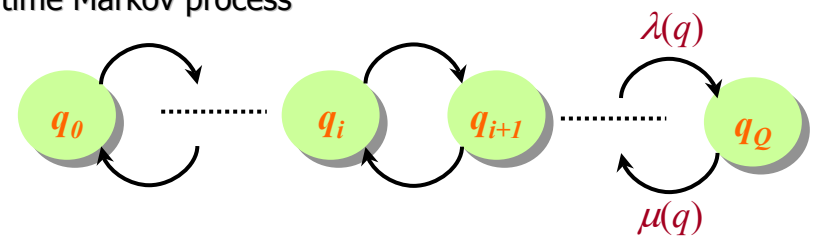
$\tau(s_i, s_j)$	Average transition time
energy(s_i , s_j)	Energy cost
power(s_i)	Power consumption
$\mu(s_i)$	Average service speed

Action set: {go_sleep, go_wait, go_idle, go_busy}

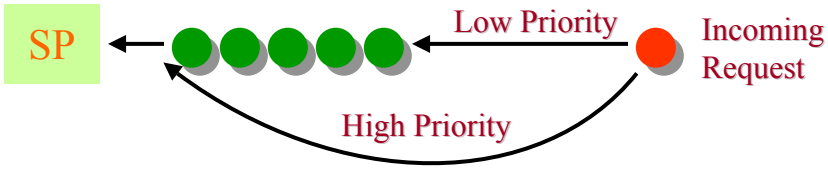
M. Pedram

Service Queue (SQ)

- The service queue is modeled as a stationary, continuous time Markov process



- In practice, requests may have different priorities



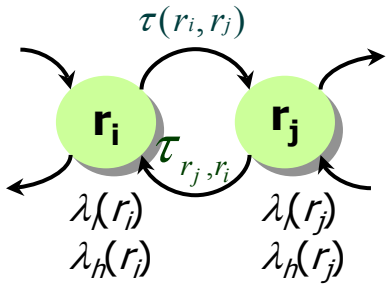
M. Pedram

Service Requester (SR)

- The service requester is modeled as a stationary, continuous-time Markov process

$\tau(r_i, r_j)$ Average transition time

$\lambda_{l,h}(r_i)$ Generation speed of low/high priority service requests in state r_i

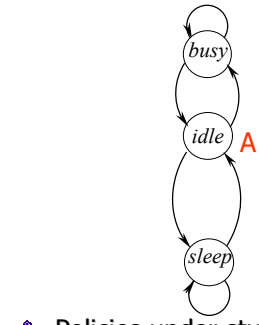


System Model

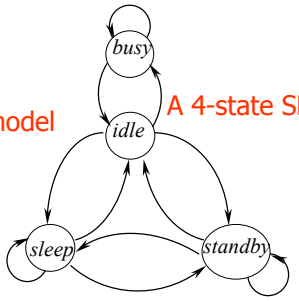
- The system can be constructed from composition of the Markov processes of the SR, SP and SQ
  - State space
    - The Cartesian product of the state space of each component minus invalid states
  - Generator matrix
    - Independent components: tensor-sum method
    - Correlated components: each entry should be computed separately
- Problem statement: Find the set of state-action pairs (I.e., the power management policy) that minimizes the average power consumption of the system subject to performance constraints

Example: A Fujitsu Hard Disk

- **Two abstract models of the Fujitsu Hard Disk**



A 3-state SP model

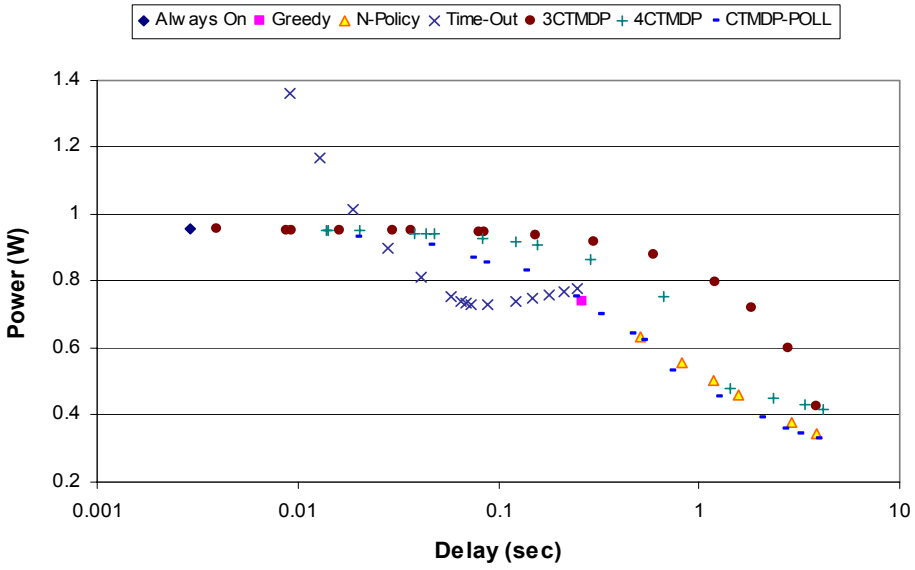


A 4-state SP model

- Policies under study:
  - "Always On" policy
  - "Greedy" reactive policy
  - "Time Out" policies with different time-out values
  - "N" policies with different N values
  - 3CTMDP: CTMDP policies using the 3-state SP model
  - 4CTMDP: CTMDP policies using the 4-state SP model
  - CTMDP-Poll: 3CTMDP with polling to address the long tail of distribution

M. Pedram

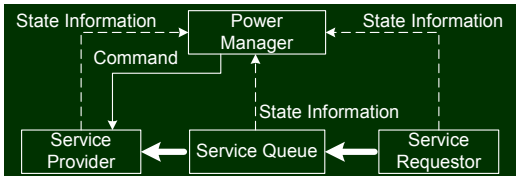
Hard Disk Results

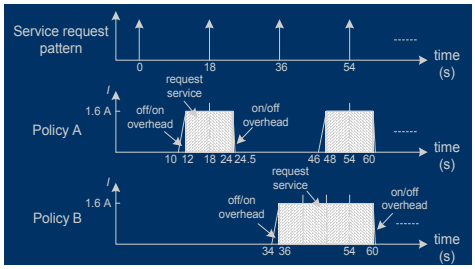


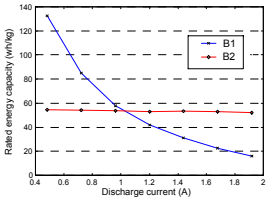
Power-delay trade-off curves for the uniform transition time distribution for the hard disk and a combination of exponential and Pareto distributions for the request inter-arrival times

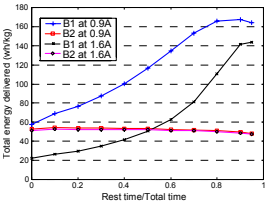
M. Pedram

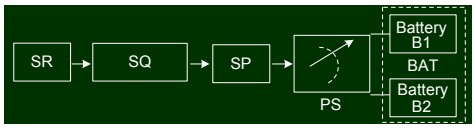
Battery-aware Power Management











		M1	M2	M3	M4	BAPM
P1	graviometric energy delivered (wh/kg)	54.35	53.24	53.32	53.20	61.25
	BAPM Capacity Gain	12.7%	15.0%	14.9%	15.1%	--
P2	graviometric energy delivered (wh/kg)	51.64	52.66	53.05	53.19	60.37
	BAPM Capacity Gain	16.9%	14.6%	13.8%	13.5%	--

M1: use a pre-assigned battery (B1 or B2) when the SP is in a particular state (busy1 or busy2)

M2: switch between the two batteries of type B1 and B2 with a fixed frequency of 0.1 Hz

M3: Similar to M2 except that we use two batteries of type B1, switching between them at a fixed frequency (0.1 Hz)

M4: Similar to M3 except that we use two batteries of type B2

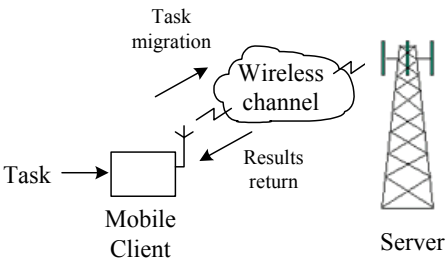
P1: As soon as a battery is completely consumed, it is immediately replaced with a new battery of the same type

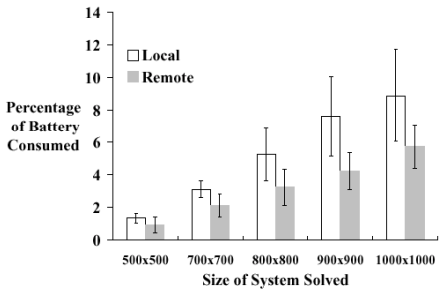
P2: The both batteries are replaced together and only after both of them have been completely used up.

M. Pedram

Extending Network Lifetime by Remote Processing

- **Scenario**
  - A mobile device providing real time services in a client-server wireless network
- **System components**
  - Mobile host (client), wireless channel, base-station (server)
- **Remote Processing**
  - Migrate a task from an energy-constrained slow-speed mobile host to an energy-unconstrained fast-speed base station to save energy of the mobile host
  - Applications: task detection and recognition, voice recognition, large-scale numerical computations, simulation, compilation
  - A limiting factor: power consumptive wireless communication



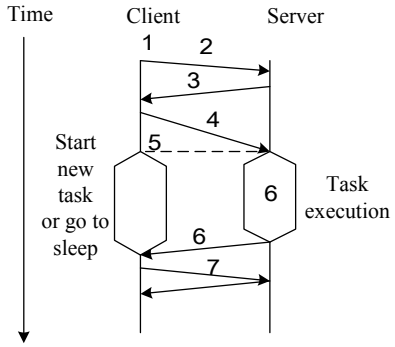


Power savings for remote execution of Gaussian solution of a system of linear algebraic equations

M. Pedram

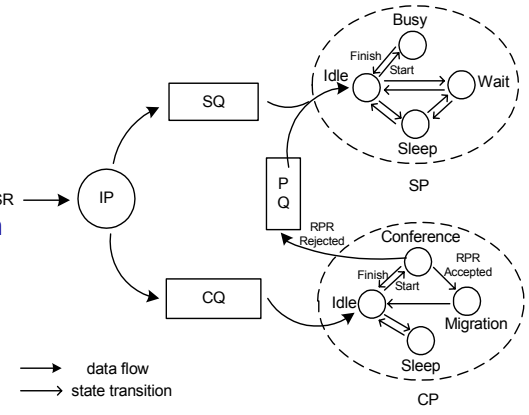
Remote Processing Procedure (RPR)

- **Protocol**
  - Step1: Prepare RPR
  - Step2: Send RPR
  - Step3: Server responds: accept or reject
  - Step4: Migrate remote execution candidate (REC)
  - Step5: Start new task or go to sleep
  - Step6: Server completes REC and informs client
  - Step7: Get back results of computation (RES)



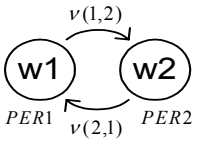
Model of the Mobile Client

- **Assumptions**
  - Continuously execute some real-time service processes for each incoming task
  - Different tasks differ in the task size which is exponentially distributed
  - Relationships between task size and execution and migration time are known in advance (e.g. profiling)
- **Component Description**
  - IP: Issue Processor
  - SP: Service Provider
  - CP: Conference Processor



Model of the Wireless Channel

- **Wireless channel**
  - Noisy, Bandwidth limited
  - Multi-path fading and shadowing effect
- **A Two-State CTMDP Model**
  - State: represents the expected packet error rate
  - State transition: model slow-fading effect
  - Extendable

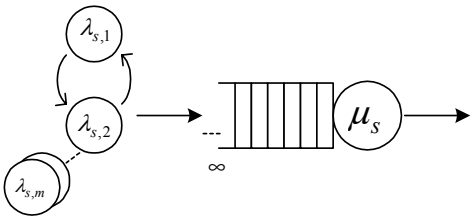


- **Data Migration Time Over Error-Prone Wireless Channel**

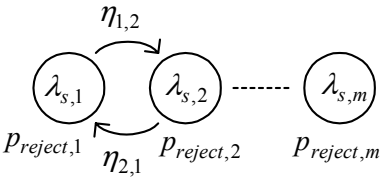
$$t_a = n \cdot \sum_{m=0}^{\infty} t_0 \cdot PER^m = \frac{nt_0}{1-PER} = \frac{t}{1-PER}$$

Model of the Server

- **Queuing Model**
  - An unbounded-length queue with a multi-state task generator



- **Rejection Probability**
  - The mobile client only cares the rejection probability of its RPRs
  - The rejection probability can be related to the parameters of the queuing model as follows:



$$p_{reject} = \sum_{n=1}^{\infty} p_n \cdot e^{\frac{-k \cdot n}{\mu_c \mu_s (c-1)}}, \quad c > 1;$$

$$= 1 - p_0, \quad c = 1$$

Experimental Results

- Offline Optimal Policy (based on solving a linear program)
  - Characteristics of the wireless channel and server dynamically change

Policy	M1	M2	PM-RPR
Average Power (W)	0.2456	0.2350	0.2239
PM-RPR Improvement	8.84%	4.72%	--

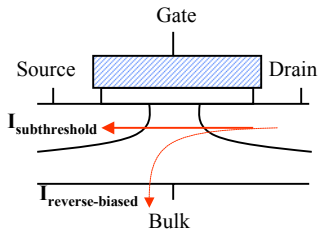
- Online Policy (based on runtime parameter calculation and table lookup)
  - Server: queuing model
  - Wireless channel: slowly and randomly changing

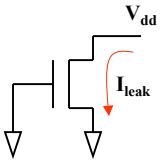
Mode	M1	M2	PM-RPR
Average Power (W)	0.2447	0.2530	0.2300
PM-RPR Improvement	6.01%	9.09%	--

Outline

- Part I
  - Sources of Power Dissipation
  - Dynamic Power Management
  - Runtime Mechanisms for Leakage Control
- Part II
  - Adaptive Voltage and Frequency Scaling
  - Energy-aware MPEG-4 FGS Video Streaming
  - Power-aware Source Routing in MANETs
- Part III
- Part IV

Leakage Current 101





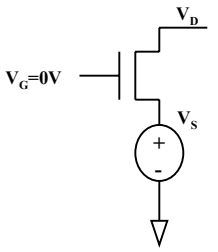
$$I_{leak} \propto K \frac{W_{eff}}{L_{eff}} e^{V_{GS}-V_T}$$

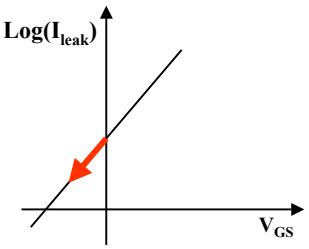
- Reduced channel length requires lowering the supply voltage
- Lowered supply voltage requires scaling the threshold voltage down
- Weak inversion subthreshold current is the dominant contributor to leakage power dissipation
- Leakage current increases exponentially as VT decreases
- High temperature also significantly increases the leakage current
- Multiple-threshold devices, stacked sleep transistors, back bias voltage control, etc. are some of the known methods to control the leakage

M. Pedram

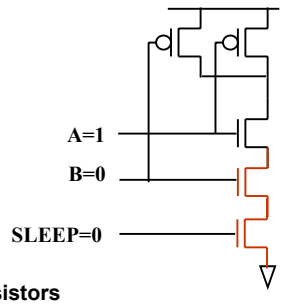
Leakage Control Mechanisms

Increase threshold voltage with self-reverse bias

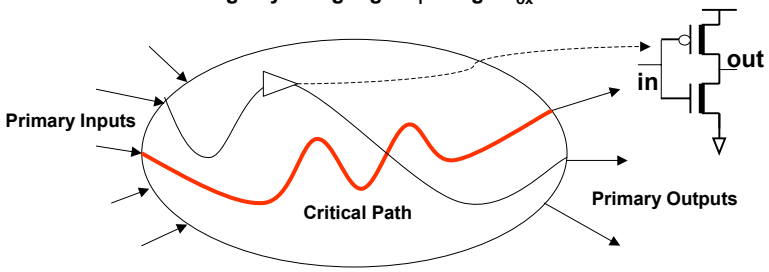




Reduce leakage by using SLEEP transistors

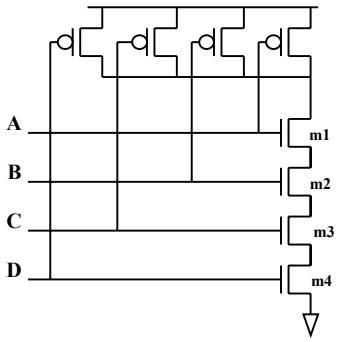


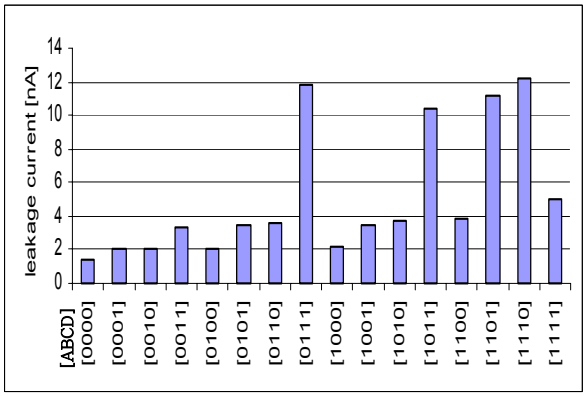
Reduce leakage by using high V_T or high T_{ox} transistors



M. Pedram

Input Pattern Dependence

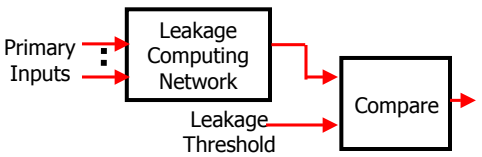




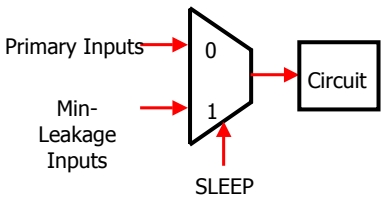
- **Up to a factor of 8 variation in leakage under different input patterns**
- **How to enforce a specific input pattern for a gate in combinational logic**

Leakage Current Minimization Flow

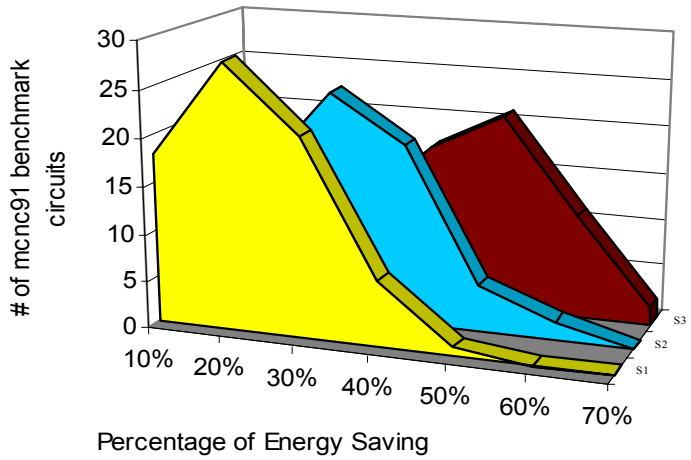
- Solve the problem of identifying the minimum leakage input vector in the circuit



- Shift in the minimum leakage input vector in the standby mode using a SLEEP signal



Simulation Results



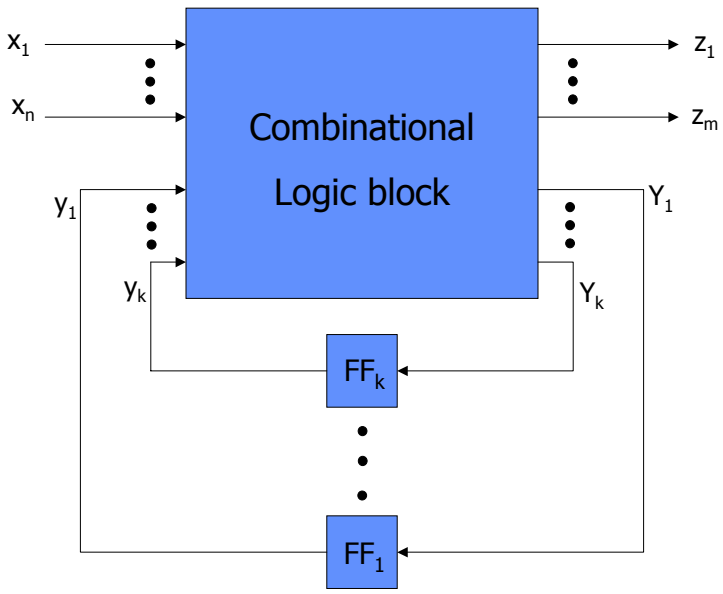
Distribution of Energy Saving with 5% Speed Degradation

Distribution of Energy Saving with 10% Speed Degradation

Distribution of Energy Saving with 15% Speed Degradation

M. Pedram

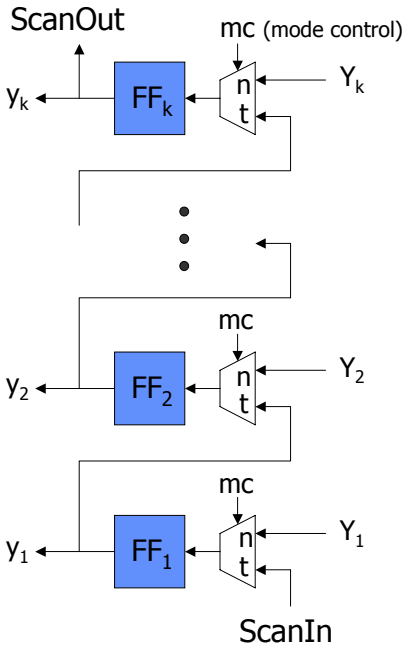
Application to Controller Circuits



M. Pedram

Review: Test Steps Using Scan Chain

1. Set ***mc = test***
2. Shift the test vector into FF's via ScanIn pin by applying k clocks
3. Apply the output of FF's to state inputs y_i
4. Set ***mc = normal*** and apply one clock (Y_i are captured in FF's)
5. Set ***mc = test***
6. Shift out the captured Y_i's in FF's from the ScanOut pin by applying k clocks



Using the Scan-Chain for Applying MLV

Build a scan-chain (SC) from input and internal flip-flops

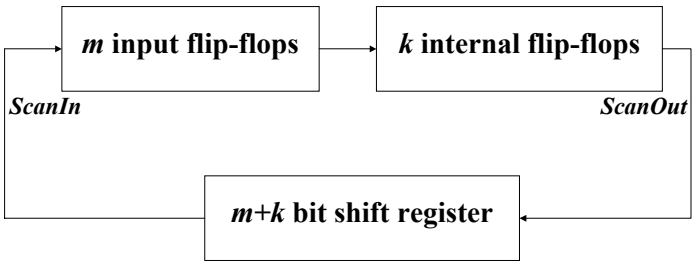
Add a shift register (SR) with the same length of SC

Construct a loop from SC and SR

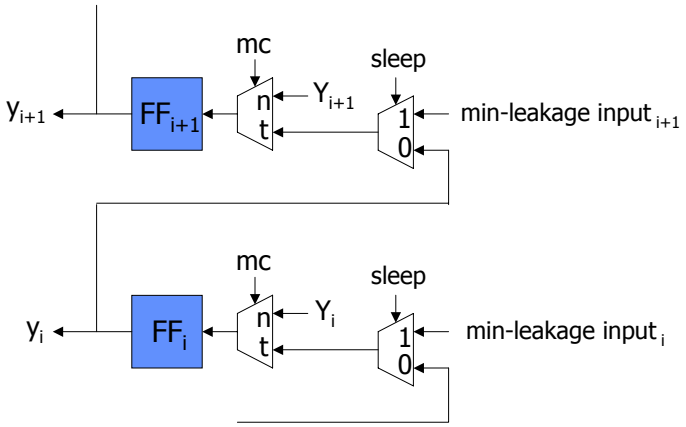
Sleep mode: Shift MLV from SR to SC and the state from SC to SR at the same time

Active mode: Shift the state from SC to SR and at the same time MLV from SR to SC

There is a (m+k) cycle performance penalty

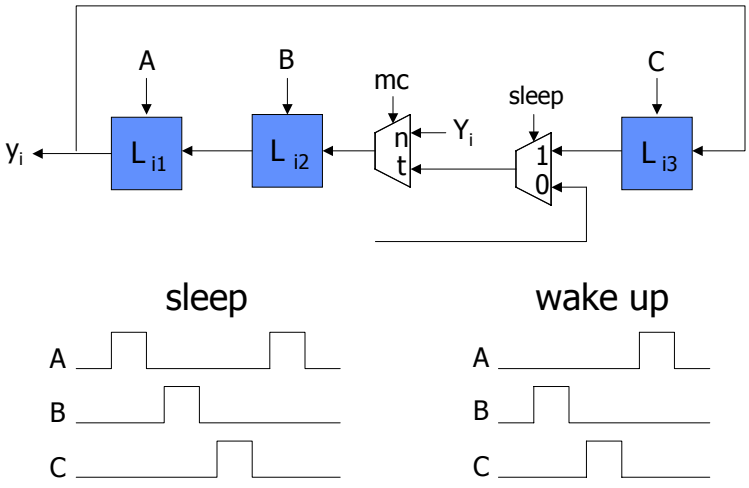


Modifying Scan Chain to Apply MLV in One Cycle



The added multiplexors are not in the critical path, therefore, they do not affect the delay in the active mode (critical path includes the path from Y_i to y_i).

Preserving and Retrieving the State in One Clock



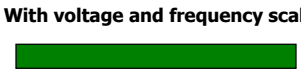
Summary

- Dynamic power consumption and subthreshold leakage are the key contributors to total power consumption in CMOS VLSI circuits
- The degree of power savings due to OS-directed power management is dependent on the workload conditions and the minimum performance and quality of service requirements; Savings of 30% or higher are achievable in most systems
- Remote processing can be used to extend the lifetime of a network of battery-powered devices while delivering an acceptable level of performance
- Leakage power may be reduced by as much as 40% by applying a min-leakage vector when the circuit enters the sleep mode

Outline

- Part I
  - Sources of Power Dissipation
  - Dynamic Power Management
  - Runtime Mechanisms for Leakage Control
- Part II
  - Adaptive Voltage and Frequency Scaling
  - Energy-aware MPEG-4 FGS Video Streaming
  - Power-aware Source Routing in MANETs
- Part III
- Part IV

Adaptive Voltage Scaling 101



	Number of Cycles	Voltage	Energy
No power-down	12.5×10^6	5	125 mJ
With power-down	5×10^6	5	50 mJ
With voltage and frequency scaling	5×10^6	2	8 mJ

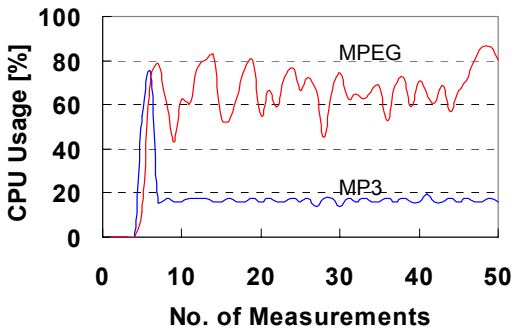
$$E \propto N \cdot V^2$$
$$f \propto V$$

M. Pedram

DVFS for a MPEG Decoder

- Dynamic Voltage and Frequency Scaling (DVFS) is an effective method for reducing the CPU power consumption
- To implement DVFS, accurate workload prediction is essential
- Prediction of a uniform or slowly varying workload is easy and can be done accurately

Workload distributions of MP3 and MPEG



M. Pedram

Workload Calculation Based on a Moving Average

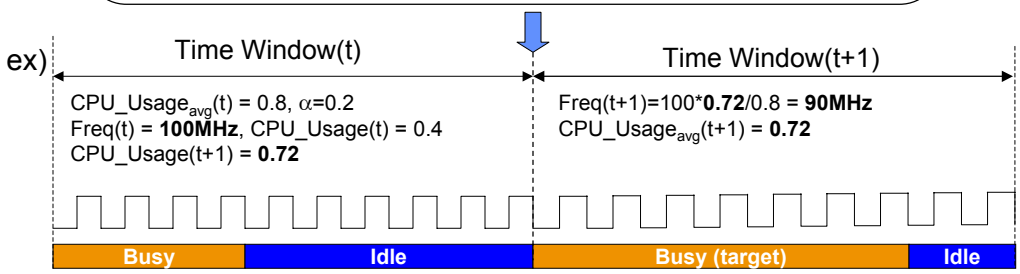
- It is straight-forward to implement an exponentially-weighted moving average of the CPU workload and then set clock freq. on that basis :

$$CPU\_Usage_{avg}(0) \equiv CPU\_Usage(0)$$

$$CPU\_Usage(t+1) = \alpha \cdot CPU\_Usage(t) + (1-\alpha) \cdot CPU\_Usage_{avg}(t)$$

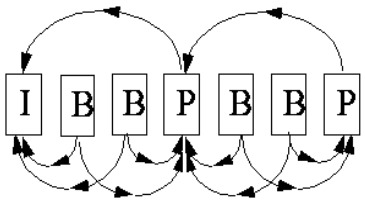
$$= \alpha \sum_{\tau=0}^{t} (1-\alpha)^{\tau} \cdot CPU\_Usage(t-\tau)$$

$$Freq(t+1) = Freq(t) \cdot \frac{CPU\_Usage(t+1)}{CPU\_Usage_{target}}$$



MPEG terminology

- There are three types of frames: I, P, and B frame



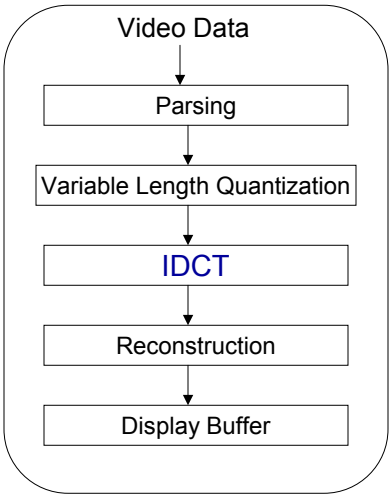
Display order : B B I B B P B B P
Frame number : 0 1 2 3 4 5 6 7 8

Decoding order : I B B P B B P B B
Frame number : 2 0 1 5 3 4 8 6 7

- I (intra) frame has no reference frame
  - All pixels in the frame should be encoded
- P (predictive) frame has a reference to a previous I-frame
  - Only parts that are different from the reference frame are encoded; Needs fewer encoded bits
- B (bi-directional) frame has two references, a previous I-frame and a subsequent I or P frame
  - Uses the least encoded bits

MPEG Decoding Steps

- Inverse Discrete Cosine Transform (IDCT) is the most CPU intensive task
- IDCT time of each frame type is different
  - I-frame > P-frame > B-frame
- The difference in the IDCT times of different frames cause a large variation in the CPU workload due
- A more accurate workload prediction method is needed

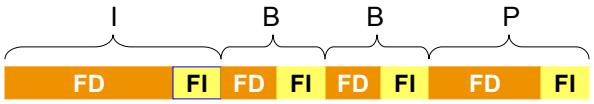


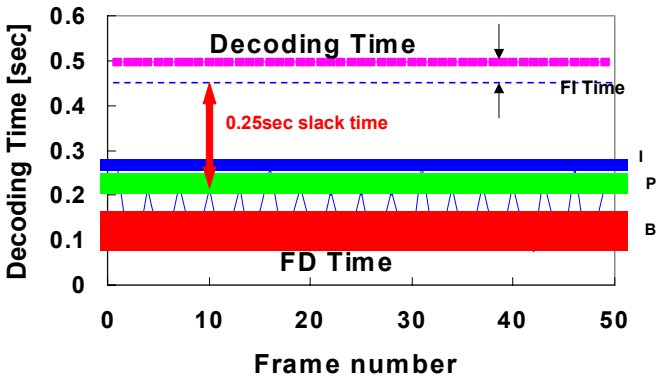
M. Pedram

Workload Prediction

- Divide the whole decoding sequence into two parts
  - Decoding time = FD (Frame-dependent) + FI (Frame-independent)

ex) IBBP





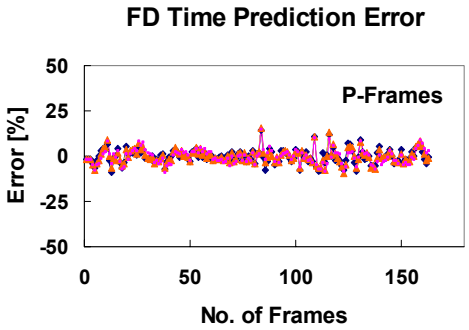
M. Pedram

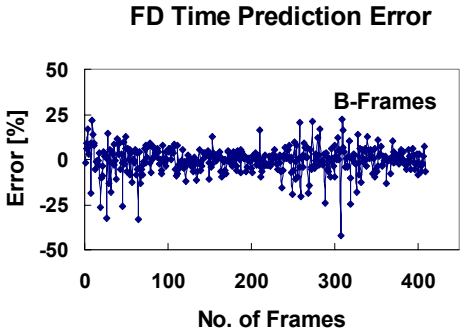
A Frame-based Workload Prediction Scheme

- The workload prediction scheme
  - Iso-type frames produce similar workloads
  - Maintain the workload statistics for each frame type

$$\text{Workload}_{type}(t+1) = \alpha \sum_{\tau=0}^{t} (1-\alpha)^{\tau} \cdot \text{Workload}_{type}(t-\tau)$$

- Prediction error (for 95% of the frames)
  - P-frame prediction error is less than 10%
  - B-frame prediction error is less than 20%

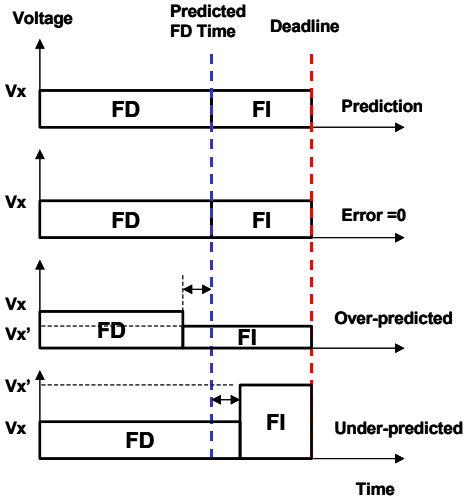




M. Pedram

Prediction Error

- When a prediction error occurs, it can be compensated by changing the CPU frequency during the FI part
  - In effect we use the FI part as a buffer zone



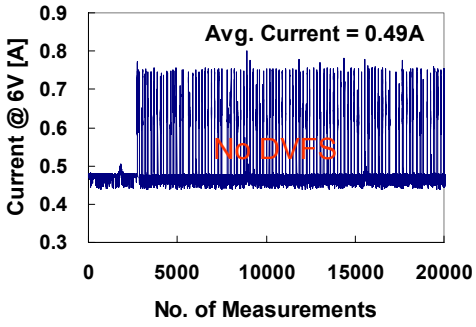
Comparison of the error ratio between P-frame and B-frame

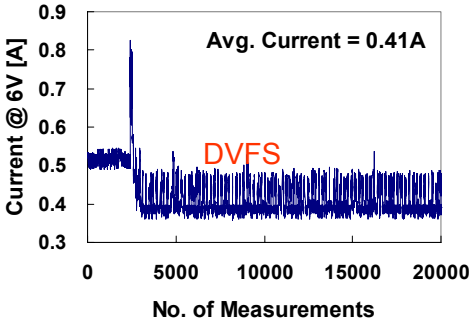
	P-frame	B-frame
FD Time	220ms	120ms
FI Time	50ms	50ms
Error Ratio	10%	20%
Remaining FI Time	28ms	26ms

- The absolute errors for B-frame and P-frame predictions is about the same

M. Pedram

DVFS Results





- Power saving is ~16% of the CPU power

Outline

- **Part I**
  - Sources of Power Dissipation
  - Dynamic Power Management
  - Runtime Mechanisms for Leakage Control
- **Part II**
  - Adaptive Voltage and Frequency Scaling
  - Energy-aware MPEG-4 FGS Video Streaming
  - Power-aware Source Routing in MANETs
- **Part III**
- **Part IV**

Wireless Video Streaming

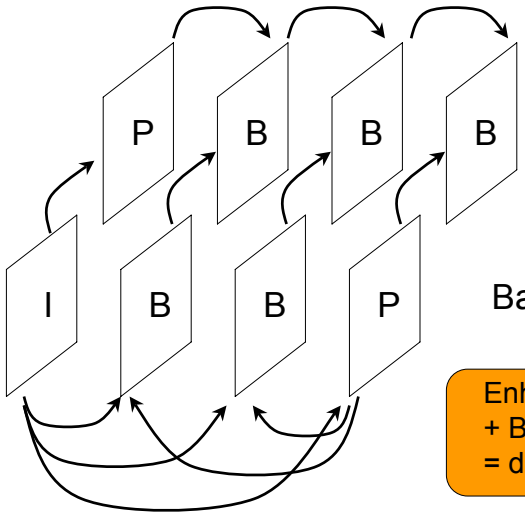
- For mobile wireless video streaming two factors should be considered: high video quality and long service time
- Stable channel for real-time operation
  - Video quality degradation due to channel congestion or error rate
  - Scalable coding technique to be adaptive channel bandwidth variation
- Power-aware operation so as to increase the lifetime of battery-powered mobile system
  - Optimal energy consumption to meet the required video quality

Scalable Video Coding

- Scalable video coding is required to be adaptive to channel variation
  - a base layer (BL) + an enhancement layer (EL)
- Temporal scalability
  - High frame rate (BL + EL)
  - Low frame rate (BL)
- Spatial scalability
  - High resolution (BL + EL)
  - Low resolution (BL)
- Signal-to-noise ratio (SNR) scalability
  - Fine image quality (BL + EL)
  - Coarse image quality (BL)

Temporal Scalability

- **Enhancement layer introduces a layer of P and B frames to increase frame rate**



Enhancement layer

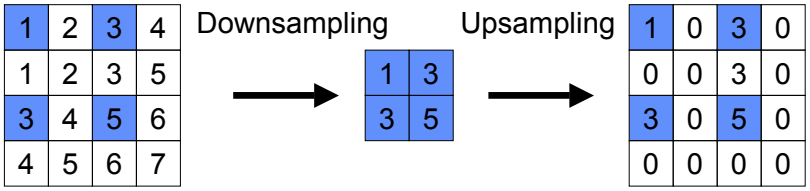
Base layer

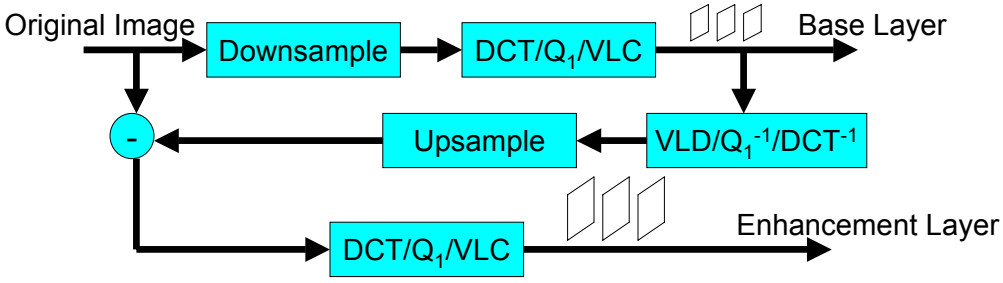
Enhancement layer frames
+ Base layer frames
= doubled frame rate

M. Pedram

Spatial Scalability

- **Code a video into two layers at different spatial resolutions**

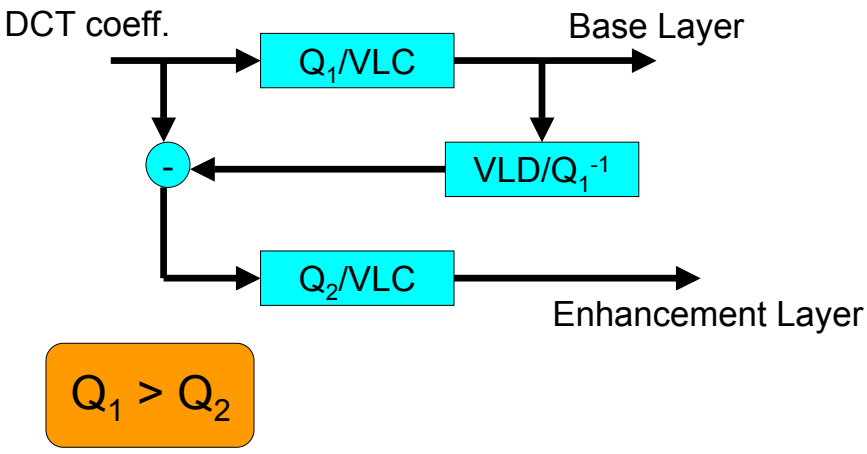




M. Pedram

SNR Scalability

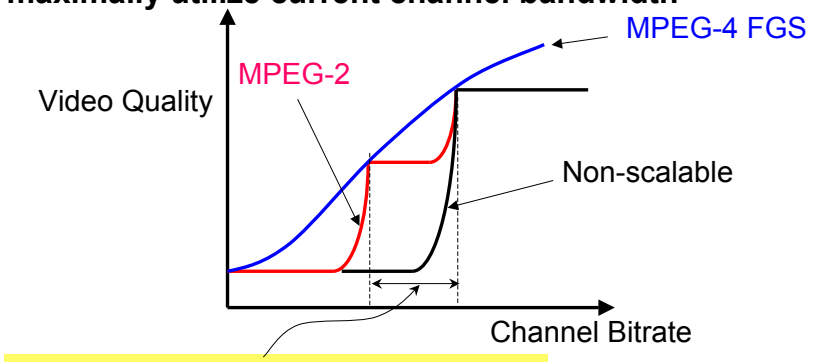
- **Code a video into two layers at different quantization accuracy**



M. Pedram

Deficiency of MPEG-2 Scalable Coding

- **MPEG-2 only provides two layers**
  - **Dramatic change in the video quality as channel bandwidth varies**
- **MPEG-4 provides many more layers**
  - **Continuous video quality improvement is desirable to maximally utilize current channel bandwidth**



M. Pedram

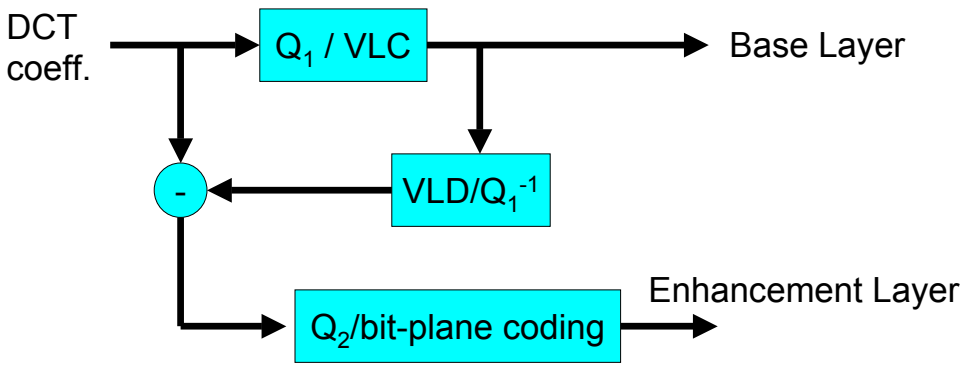
MPEG-4 FGS (Fine-Granular Scalability)

- Graceful degradation of video quality under network losses and bandwidth variation using hierarchical layer structure
  - a base layer (BL) + an enhancement layer (EL)
- BL guarantees the minimum video quality
  EL improves video quality if channel bandwidth allows
- EL bit-stream can be truncated into any number of bits by using bit-plane coding
  - provides continuous scalability as channel bit-rate varies

M. Pedram

MPEG-4 FGS Layer Structure

- Enhancement layer is equal to the original image minus the reconstructed image from the base layer

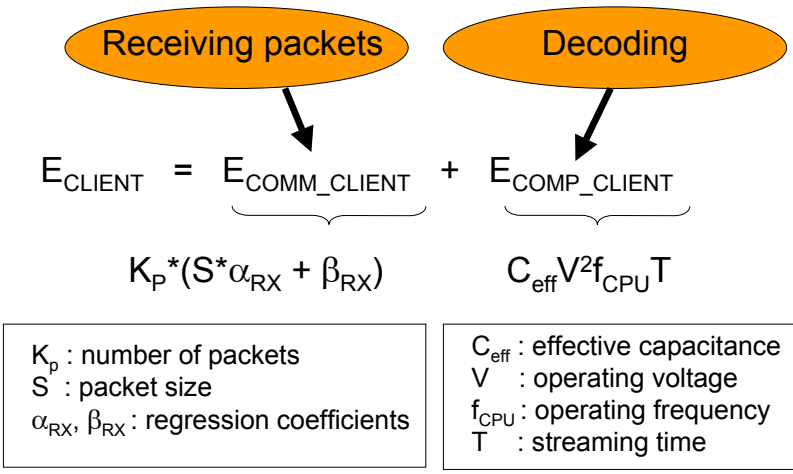


M. Pedram

Energy Consumption in Video Streaming

- There are two sources of energy consumption in wireless video streaming : the communication energy and the computation energy
- Communication energy
  - Transmitting packets (server)
  - Receiving packets (client)
- Computation energy
  - Packetization (server)
  - Decoding bit-streams (client)
- A video streaming system in which a server and a mobile client is considered

Energy Consumption at the Client



Energy Waste at the Client

- **Video streaming is real-time operation**
- **If the client cannot process all the packets from the server in a given deadline, then the communication energy is wasted with no improvement of video quality**

Ex) Arrived packet number : A
Decoded packet number : M

Video quality = min(M, A)

If A > M, then (A-M) packets are useless resulting in energy waste in handling those packets

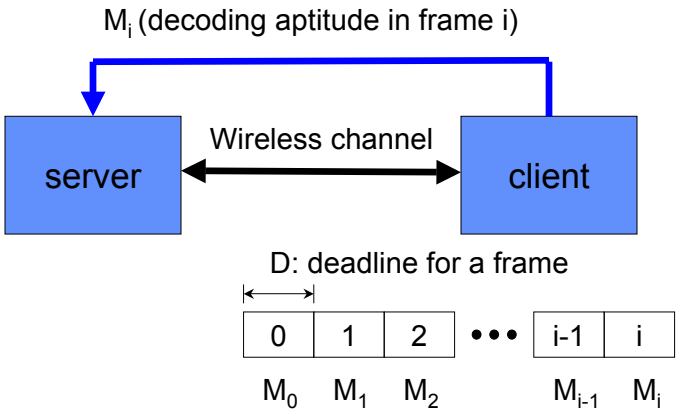
- **For an energy-efficient streaming in which no energy waste occurs, A should be equal to M**

Decoding Aptitude of the Client

- **Decoding aptitude (M) of a mobile client**
  - **The amount of data that can be decoded in a given deadline**
  - **M is proportional to the CPU frequency**
  - **M can be changed by dynamic voltage and frequency scaling (DVFS) or dynamic power management (DPM)**
- **N_i, normalized decoding load at time instance i, is defined as the ratio A_i/M_i**
  - **A_i : the number of correctly arrived packets at the client**
  - **M_i : decoding aptitude**
- **N_i represents the degree of energy waste and should be kept to one for no energy waste**
- **In order to make N_i one, the server should know the M_i.**

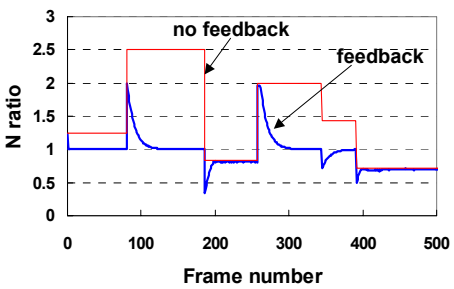
Client-feedback Video Streaming

- A status packet is periodically sent to the server at regular time intervals
- The server sets the amount of data to be transferred based on the client status: making N_i equal to one



Simulation Results

- Variations in N_i with different M_i/B ratio
  - M_i : decoding aptitude at instance i
  - B : maximum number of packets the server can send
- M_i trace : 0.8B, 0.4B, 1.2B, 0.5B, 0.7B, 1.4B
- Channel model : Gilbert-Elliot model with bit error rate (BER) of $1e^{-5}$ and $1e^{-4}$ for good and bad state, respectively

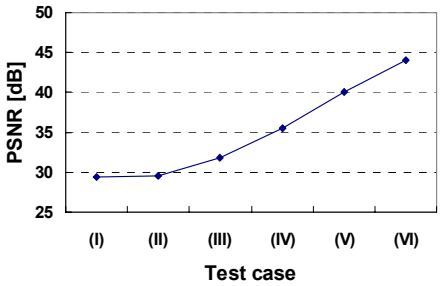


	Energy waste	
	No FB	FB
0.8B	18.74%	0.21%
0.4B	57.35%	2.57%
1.2B	0%	0%
0.5B	48.49%	6.27%
0.7B	28.69%	0%
1.4B	0%	0%

Experimental Setup

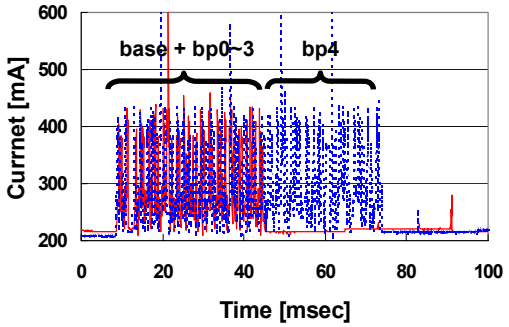
- **Generated MPEG-4 FGS bit-streams using a QCIF test video**
  - **A base + FGS with five bit planes(bp0~4)**
  - **256-byte packet size**
- **Six test cases**
  - **(I) base layer only**
  - **(II) base + bp0**
  - **(III) base + bp0 + bp1**
  - **(IV) base + bp0 + bp1 + bp2**
  - **(V) base + bp0 + bp1 + bp2 + bp3**
  - **(VI) base + bp0 + bp1 + bp2 + bp3 + bp4**
- **Peak signal to noise ratio (PSNR) increases as more bit-planes are decoded**

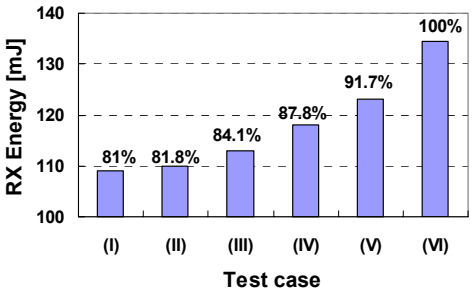
	Size(byte)	Packet number
Base	76	1
FGS Header	9	1
bp0	18	1
bp1	278	2
bp2	1007	4
bp3	2022	8
bp4	3358	14



M. Pedram

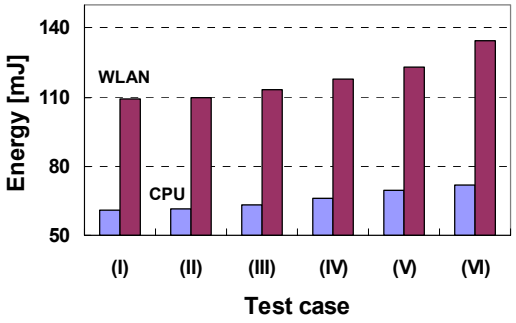
Energy of Wireless LAN Card for Receiving Packets

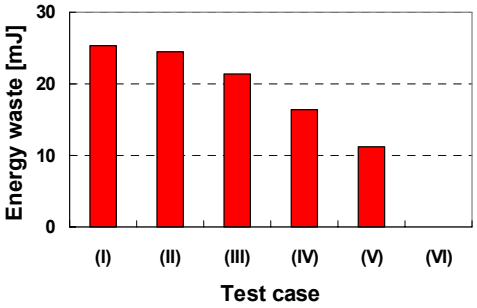




M. Pedram

Energy Consumptions of the CPU and WLAN

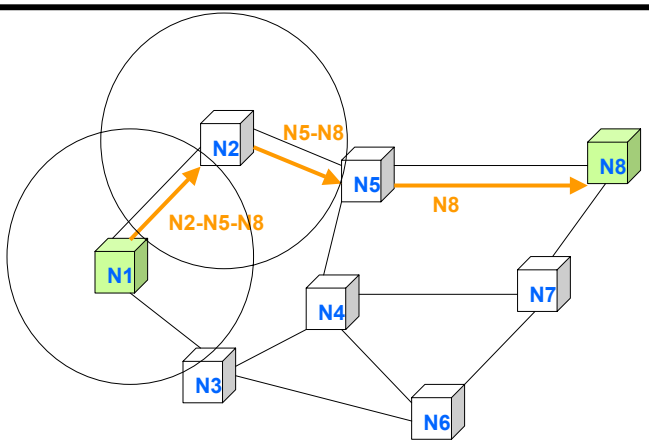




Outline

- Part I
  - Sources of Power Dissipation
  - Dynamic Power Management
  - Runtime Mechanisms for Leakage Control
- Part II
  - Adaptive Voltage and Frequency Scaling
  - Energy-aware MPEG-4 FGS Video Streaming
  - Power-aware Source Routing in MANETs
- Part III
- Part IV

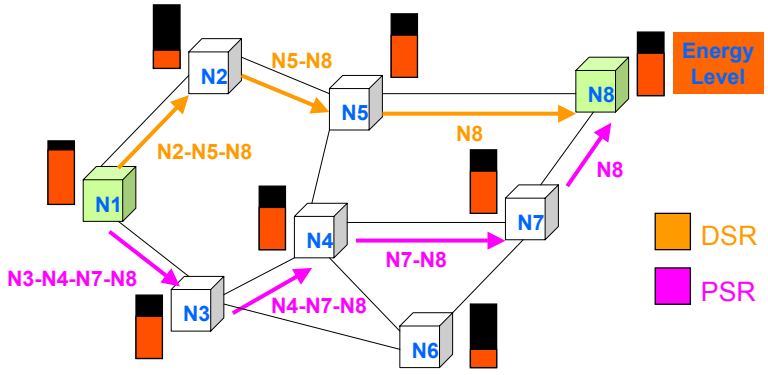
Dynamic Source Routing in MANET 101



- Network can become disconnected if some nodes die out due to lack of energy; A design goal is to maximize the (useful) network lifetime by minimizing the variance of the remaining life of the nodes in the network
- Focus is on on-demand (reactive) routing protocols; Assume fixed transmit power and fixed energy-conserving strategy in each node
- Route discovery is done by flooding the network with Route Requests; Nodes promiscuously listen to control messages flowing through the network; Caching techniques improve performance considerably

M. Pedram

Power-aware Source Routing (PSR)



- Minimize sum of the energy cost of the links along the routing path
- Link cost is proportional to the inverse of the remaining battery capacity (residual energy) of the transmitting node

M. Pedram

PSR: Route Discovery

- Similar to DSR, but with some differences:
  - An intermediate node passes on the first RREQ and all subsequent lower-cost RREQ's until a local timer expires
  - Destination starts a timer after receiving the first RREQ and replies back only after that timer expires
- Control packet overhead for PSR is higher than that of DSR

PSR: Route Maintenance

- Similar to DSR, but with some differences:
  - Path cost changes with time, and hence, cache entries of a node should not remain valid for a "long" period of time (aged cache entries should be purged from the cache tables of all nodes along the path)
  - Each node in the path monitors the decrease in its residual energy from the time of route discovery; When this link cost increase goes beyond a threshold level, the node sends a route error back to the source as if the route was rendered invalid
  - Invalidated routes are added to a victim buffer
- PSR results in a higher degree of energy balancing in the network

PSR: Exact Cost Function

$$\underset{\pi}{Min}\ C(\pi,t)=\sum_{i\in\pi}C_i(t)$$

$$\text{where}\ C_i(t)=\rho_i.\left(\frac{F_i}{E_{r,i}(t)}\right)^{\alpha}$$

ρ_i: transmit power level of node i

F_i : full-charge battery capacity of node i

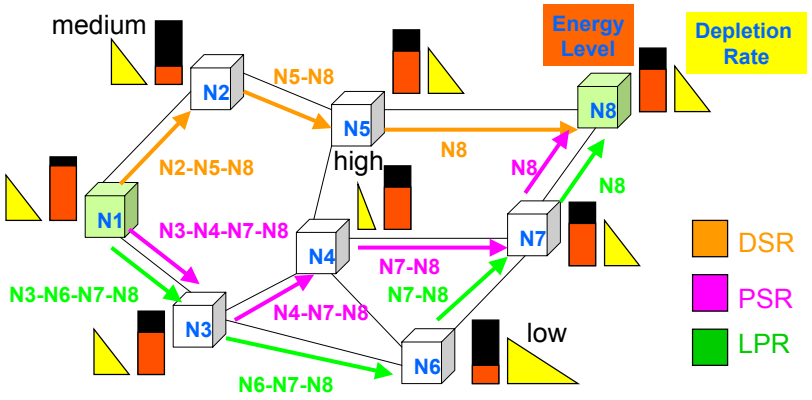
$E_{r,i}(t)$: remaining battery capacity of node i at time t

α: a positive weighting factor

- A cost is associated with every node on the path
- This cost is inversely proportional to the normalized residual energy of the node
- The cost function is graded, i.e., nodes with very low battery capacity dominate the total cost of the path

M. Pedram

Lifetime Prediction Routing (LPR)



- Maximize the minimum link cost along routing path
- Link cost is remaining lifetime of transmitting node
- Node lifetime is equal to the remaining battery capacity divided by the energy depletion rate

M. Pedram

LPR: Exact Cost Function

$$\max_{\pi} T_{\pi}(t) = \min_{i \in \pi}(T_i(t))$$

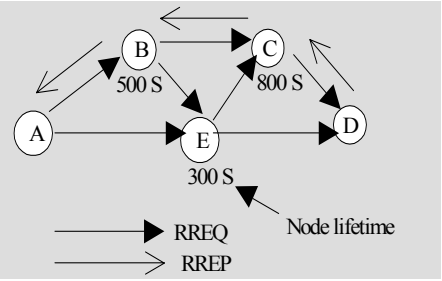
$$T_i(t) = \frac{E_{r,i}(t)}{\frac{1}{N-1}\sum_{k=i-N+1}^{i} R_k(t)}$$

$T_{\pi}(t)$: lifetime of path π
$T_i(t)$: predicted lifetime of node i in path π

- Er,i(t): remaining energy at the ith packet is being sent or relayed through the current node
- Rk(t): rate of energy depletion of current node
- N: length of the history used for calculating the simple moving average

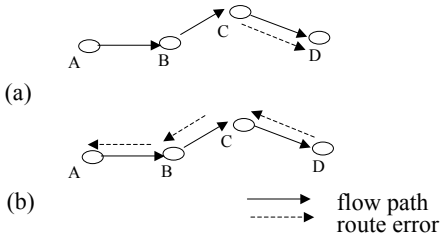
LPR : Flow Setup Process

- Similar to to PSR except that
  - Each node predicts its lifetime when it receives a RREQ
  - Intermediate nodes attach their predicted lifetime to the RREQ packet if it is lower than the current lifetime in the header of the packet



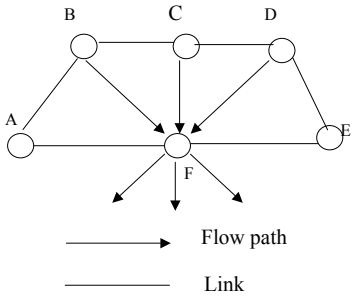
LPR : Route Expiration Process

- Similar to PSR except that:
  - Based on a Timer (RET): Invalidate a cache entry and avoid aging of cache entries and overusing routes
  - Based on a threshold (THR): Change in the predicted lifetime of the critical node in a route



LPR Balances the Traffic Load

- Cost function of LPR is dependent on the residual energy and discharge rate of nodes
- LPR avoids energy depleted and/or traffic congested paths

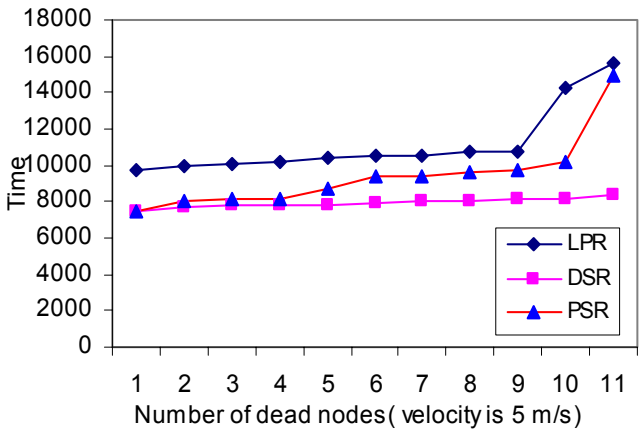


Simulation Setup

- 20 nodes in 1000 x 1000 area with random way-point mobility
- 380 UDP connections which are randomly initiated between nodes at simulation time
- Key parameters of study are the network lifetime and RMS of energy consumption (Erms) in the network
  - Effect of mobility
  - Effect of radio transmission range

M. Pedram

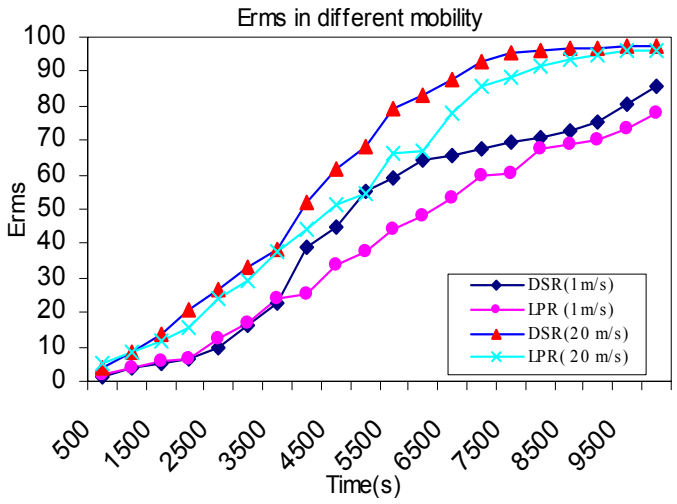
Network Lifetime



- LPR > PSR > DSR in terms of network lifetime
- Lifetime prediction is an effective technique to increase the network lifetime

M. Pedram

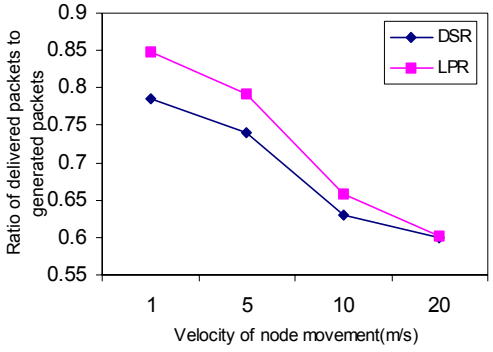
Effect of Mobility on Evaluation of Erms



- LPR is better than DSR in terms of energy variance

M. Pedram

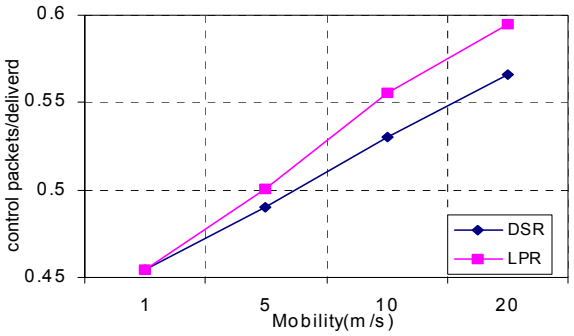
Effect of Mobility on Packet Delivery Ratio



- As velocity of node movement increases, ratio of packets delivered to packets generated for LPR goes down
- At higher velocities of node movement, LPR does not exhibit a significant gain over DSR

M. Pedram

Effect of Mobility on Control to Data Packet Ratio



- Ratio of control packets to delivered packets in the network increases as velocity of node movement goes up for LPR
- Overhead of control packets is less than 6%

Summary

- Energy consumption of an MPEG-2 decoder can be reduced by employing an application –aware DVFS strategy by as much as 20%
- By using a client-feedback method for the MPEG-4 streamer, about 20% reduction in the communication energy is achieved, which is up to 40% of the CPU energy
- Power-aware Routing protocols attempt to improve the network lifetime and balance both the remaining energy and energy depletion rate of the nodes in a MANET with low control overhead
- Performance of these protocols varies based on: Mobility, Radio transmission range, and Accuracy of lifetime prediction

References (Part I and II)

- M. Pedram and J. Rabaey (editors), Power-aware Design Methodologies, Kluwer Academic Publishers, Boston, Jun. 2002.
- P. Heydari, S. Abbaspour and M. Pedram, “A comprehensive study of energy dissipation in lossy transmission lines driven by CMOS inverters,” Proc. of IEEE Custom Integrated Circuits Conference, May 2002.
- P. Rong and M. Pedram, “Extending the Lifetime of a Network of Battery-Powered Mobile Devices by Remote Processing: A Markovian Decision-based Approach,” CENG Tech Report, Univ. of Southern California, Dec. 2002.
- Q. Qiu, Q. Wu, and M. Pedram, “Stochastic modeling of a power-managed system: construction and optimization,” IEEE Transactions on Computer Aided Design, Vol. 20, No. 10, Oct. 2001, pp. 1200-1217.
- P. Rong and M. Pedram, “Battery-aware power management based on Markovian decision processes,” Proc. of Int'l Conference on Computer Aided Design, Nov. 2002, pp. 707-713.
- A. Abdollahi, F. Fallah, and M. Pedram, “Leakage current reduction in sequential circuits by modifying the scan chains,” To appear in Proc. of Int'l Symp. on Quality of Electronic Designs, Mar. 2003.
- A. Abdollahi, F. Fallah, and M. Pedram, “Runtime mechanisms for leakage current reduction in CMOS VLSI circuits,” Proc. of Symp. on Low Power Electronics and Design, Aug. 2002, pp. 213-218.
- K. Choi, K. Dantu, W-C. Cheng, and M. Pedram, “Frame-based dynamic voltage and frequency scaling for a MPEG decoder,” Proc. of Int'l Conference on Computer Aided Design, Nov. 2002, pp. 732-737.
- K. Choi, et al, “Energy-Aware MPEG-4 FGS Streaming ,” CENG Tech Report, Univ. of Southern California, Dec. 2002.
- M. Maleki, K. Dantu, and M. Pedram, “Lifetime Prediction Routing in Mobile Ad Hoc Networks,” To appear in Proc. of IEEE Wireless Communication and Networking Conf., Mar. 2003.
- M. Maleki, K. Dantu, and M. Pedram, “Power-aware source routing protocol for mobile ad hoc networks,” Proc. of Symp. on Low Power Electronics and Design, Aug. 2002, pp. 72-75.

Outline

- **Parts I/II**

- **Part III: Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation
  - **Specification and Modeling**
    - Concurrent processes
  - **Analysis & Simulation**
    - Function/Architecture co-design

- **Part IV: Communication-based design**
  - **Node-Centric Perspective**
    - P2P communication
  - **Network-Centric Perspective**
    - Channel impact on performance

Part III: Specification, Modeling, Analysis

Outline

- **Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation & basic issues
      - Analysis vs. simulation
  - **Specification and Modeling**
    - Concurrency & communication
    - Implementing processes
      - Matlab's Stateflow and other specification languages
  - **Analysis & Simulation**
    - Stochastic models
    - Transitional semantics
      - SAN formalism
    - Application/Architecture modeling
      - Markovian & non-Markovian analysis
    - Simulation

Motivation for Formal Methods

- **Art & *Science* of system-level design**
  - **Time-to-market**
  - **Quality**
  - **Cost (re-use strategy)**

- **System-level analysis based on formal approaches**
  - **Disciplined (systematic) system-level design**
    - Need a platform!
  - **Build first a stochastic model at the highest level of abstraction and then analyze it to derive performance metrics**
    - … but the low-level support is important

A *model* can be constructed to represent some aspects of the dynamic behavior of a system. Once constructed, such a model becomes a *tool* used to investigate the behavior of the system.

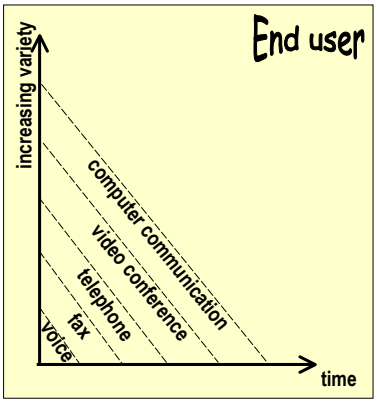
Modeling

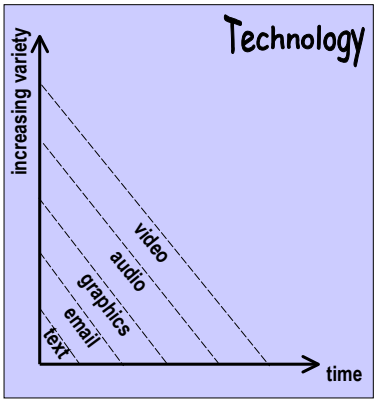
- **Three main issues for any technical system**
  - **Functionality**
  - **Performance**
  - **Economics**
    - *Redesign it's very costly!*
- **Models as tools**
  - **Make predictions about systems' behavior**
    - Should represent an *abstraction* of the system
    - Should be accurate but *tractable*
  - **Experimentation/simulation is expensive**
    - Build models and use parameters to reflect several alternatives
    - Optimization
  - **Critical issue: integrate performance into system design**

Analysis and Simulation

- **Performance evaluation**
  - **Performance measures (response time, throughput, utilization, power, SNR,…)**
  - **Simulation (*guessing* the solution)**
    - Very familiar to everyone: Spice, Cadence, Matlab, VCC,…
    - It's very time consuming: typically, tens of hours for MM systems
    - Does not offer strong guarantees
    - Do people use it? (Yes, a LOT!)
  - **Analysis (*approximating* the solution)**
    - Less familiar: Petri nets, queuing networks, process algebras (PAs), etc.
    - Mostly based on *stochastic models*
    - Much faster than simulation: typically, from a few to a few tens of minutes
    - Do people use analysis? (Yes, whenever possible. State space explosion is a big issue!)
  - **What's between simulation and verification?**
    - Answer: ANALYSIS!

Emergence of MultiMedia





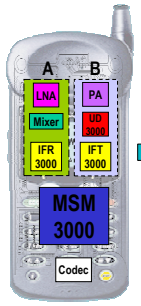
Media Types

- Discrete media
  - "Plain" data (e.g. binary files)
  - Encoded text
  - Bitmap images
  - …
- Continuous media
  - Audio data
  - Video data
  - Formatted video data (e.g. compressed video)
  - …

- Text: storage for one page of text ~12Kbytes
- Bitmap images: storage for one bit mapped page ~384Kbytes
- Digitized audio: storage for digitized audio: ~88Kbytes/sec
- Digitized video: storage for digitized video: ~30Mbytes/sec

➢These figures are calculated according to the provision of a particular QoS. The notion of QoS is fundamental to MM computing!

Evolving to Smaller More Feature-Rich Devices



1998

50% Size Reduction

50% Greater Standby Time

1999

Support for future standards

Significantly higher integration

Support for Internet/data applications

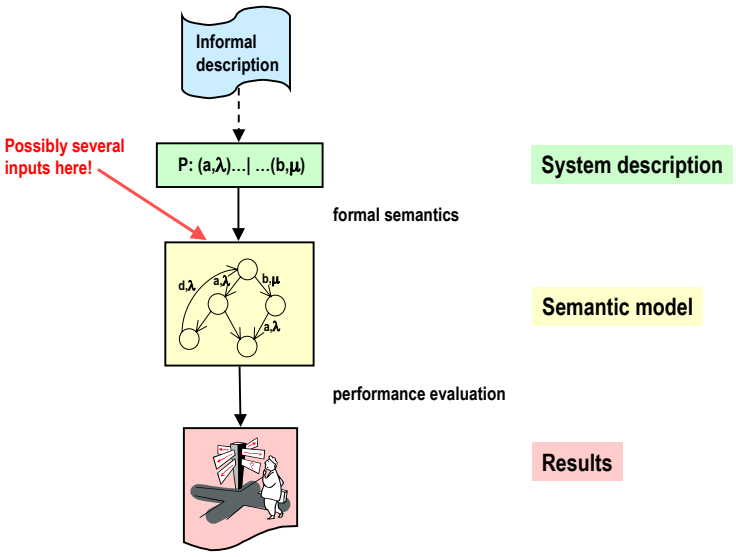
Support for multiple channels/high data rates

Tomorrow's Wireless Devices

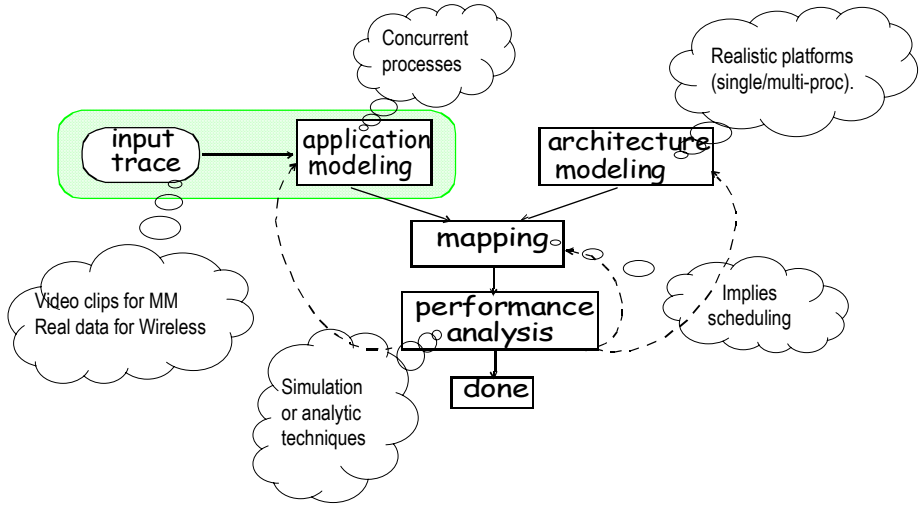
2000

Andrew J. Viterbi, Tutorial Hot Chips XII, Aug. 2000

An Abstract Perspective



… and A Design Perspective



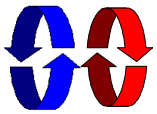
Outline

- **Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation & basic issues
      - Analysis vs. simulation
  - ➔ **Specification and Modeling**
    - Concurrency & communication
    - Implementing processes
      - Matlab's Stateflow and other specification languages
  - **Analysis & Simulation**
    - Stochastic models
    - Transitional semantics
      - SAN formalism
    - Application/Architecture modeling
      - Markovian & non-Markovian analysis
    - Simulation

Why using concurrency?

- A sequential program has a single thread of control.

- A concurrent program has ***multiple threads*** of control allowing it perform multiple computations in parallel and to control multiple external activities which occur at the same time.



- Advantages
  - Performance gain from multiprocessing hardware
    - parallelism.
  - Increased application throughput
    - an I/O call need only block one thread.
  - Increased application responsiveness
    - high priority thread for user requests.
- … but there are also problems
  - Complexity of concurrent programs
    - Designing and testing are very difficult

Processes and Threads

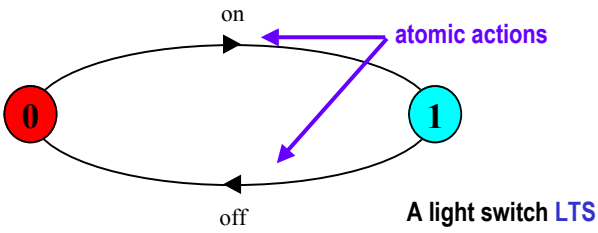
Concepts: processes - units of sequential execution.

Models: Finite State Processes (FSP) – algebraic form to model processes as sequences of actions.
Labelled Transition Systems (LTS) – graphical form to analyse and display their behaviour.

Practice: Statecharts, Java threads, etc.

Modeling Processes

A process is the execution of a sequential program. It is modelled as a *finite state machine* which transits from state to state by executing a sequence of atomic actions (*Labelled Transition Systems - LTS*).



A light switch LTS

on→off→on→off→on→off→ ……. a sequence of actions or *trace*

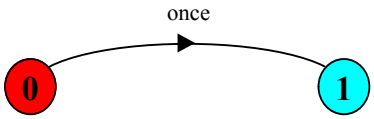
Representation is finite while the *behavior* may be infinite!
Need an *algebraic* notation (FSP) to manage complexity.

FSP - action prefix

If x is an action and P a process then (x-> P) describes a process that initially engages in the action x and then behaves exactly as described by P.

This is only a notation! We'll see soon another one (Milner's)…

```
ONESHOT = (once -> STOP).
```



ONESHOT state machine (terminating process)

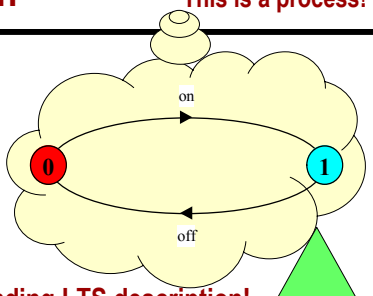
Convention: *actions* begin with lowercase letters
PROCESSES begin with uppercase letters

FSP - action prefix & recursion

This is a process!

Repetitive behaviour uses recursion:

```
SWITCH = OFF,
OFF    = (on -> ON),
ON     = (off-> OFF).
```



Every algebraic description has a corresponding LTS description!

Substituting to get a more succinct definition:

```
SWITCH = OFF,
OFF    = (on ->(off->OFF)).
```

And again:

This is a process!

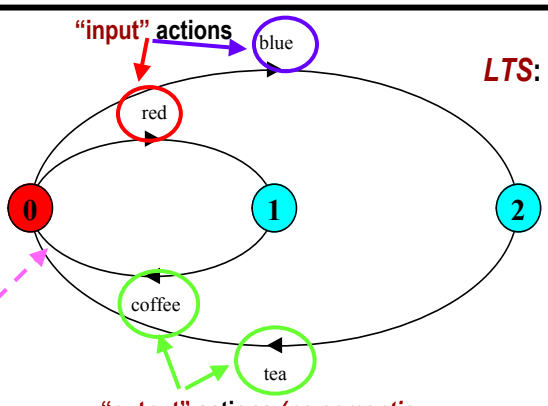
```
SWITCH = (on->off->SWITCH).
```

Same LTS description!

FSP - choice

choice

"input" actions



LTS:

"output" actions *(no semantic difference from input actions)*

Possible traces?

FSP model of a drinks machine :

```
DRINKS = (red->coffee->DRINKS
         |blue->tea->DRINKS).
```

Milner's calculus…

- **What is it?**
  - **A *theory* of communicating systems**
    - Strong mathematical basis (just a few few basic ideas)
    - Huge impact on research
      - One of the top cited authors in Computer Science (citeseer)
  - **CCS = 'A Calculus of Communicating Systems' (Springer 1980)**
    - Milner calls it "Process Calculus"
  - **CSP = 'Communicating Sequential Processes' (Hoare, CACM 1978)**
    - It's more a programming language

- **Closely related: Process Algebras (PAs)**
  - Many PAs exist today; they are all essentially *behavioral* approaches
  - Besides these, logics (mainly temporal) have strongly contributed to concurrent systems

Modeling Communication

- ***Communication* and *concurrency* are complementary notions**
  - **Diversity/unity in complex systems**
  - **Parts of a system have identity as *agents***
    - Level of decomposition depends upon our interest, NOT upon the entities
    - No distinction between systems and their components

- ***Agent*: any system whose behavior consists of discrete actions**
  - **Each agent action**
    - is either an *interaction* with the neighboring agents (environment) $\rightarrow$ communication
    - … or it occurs *independently* and then it may occur concurrently w/ their actions
      - All independent actions are *internal* communications
  - **Everything is ultimately *communication***
    - *The behavior of a system is its entire capability of communication*
    - *The behavior of a system is exactly what is observable*

FAQs

- **Are PAs a specification language?**
  - **No! PAs provide a *conceptual* and *logical* foundation for writing formal specifications**
    - PAs determine what a specification must say; a language determine in *detail* how it is said
- **What is a *formal specification*?**
  - **The "specification" is a 'contract' between the user of a system and its designer**
    - This contract should tell everything the user needs to know to use the system, and it should tell the designer everything he must know to implement it
    - Similar to ISA discussed in the computer architecture context
  - **The question of whether or not an implementation satisfies the specification must be reducible to to the question of whether or not some properties of the system are provable in some mathematical system**
    - The existence of a formal basis is the only way to guarantee that a specification is unambiguous and the system meets some performance criteria

FAQs

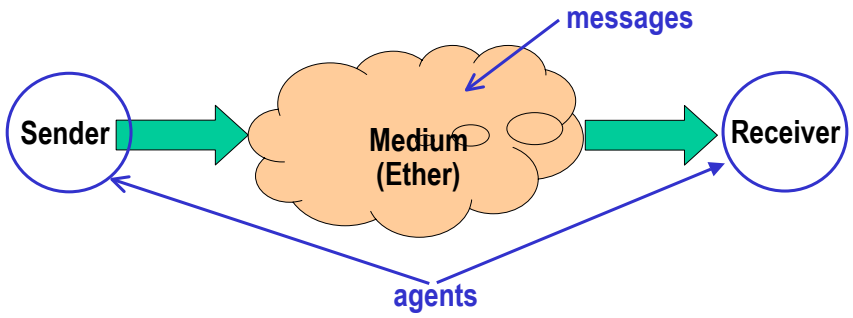
- **What is a "system"?**
  - **Anything that interacts w/ the environment (for us, perhaps in a digital manner) across a well-defined boundary**
    - Example: an airline reservation system. (How about the Solar system?)
- **How can a formal system specify *all* of the properties of a "real" system ?**
  - **Some properties are easier to specify (e.g. behavioral properties) others are much harder (e.g. average response time, power)**
- **Are people using today these theories?**
  - **A lot! Mostly in formal verification and performance analysis**
  - **We will discuss *performance*-related issues of portable MM systems and see how formal methods can help**

Typical Issues in Concurrent Specifications

- **Separate (but intimately correlated) issues:**
  - **Communication**
    - How do two blocks communicate?
  - **Synchronization**
    - How can we enforce dependencies/constraints?
  - **Execution semantic**
    - When does a block execute?

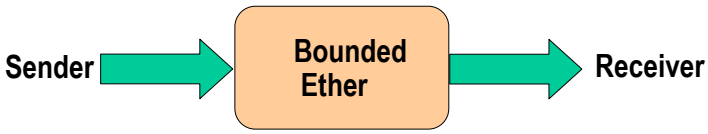
Communication Media

Main issue: information transmission between agents



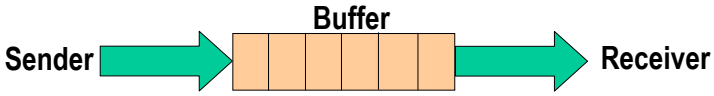
- **The ETHER discipline**
  - **The Sender may always send a message**
  - **The Receiver may always receive a message, provided the medium is not empty**
  - **The order of receiving messages may differ from the order of sending messages**

Communication Media



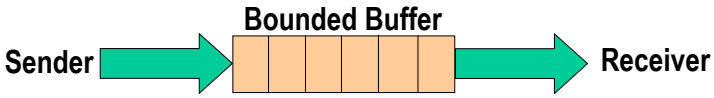
- The BOUNDED ETHER discipline
  - The Sender may always send a message, provided the medium is not full
  - The Receiver may always receive a message, provided the medium is not empty
  - The order of receiving messages may differ from the order of sending messages

Communication Media



- The BUFFER discipline
  - The Sender may always send a message
  - The Receiver may always receive a message, provided the medium is not empty
  - The order of receiving messages is the same as the order of sending messages

Communication Media



- The BOUNDED BUFFER discipline
  - The Sender may always send a message, provided the medium is not full
  - The Receiver may always receive a message, provided the medium is not empty
  - The order of receiving messages is the same as the order of sending messages

A Few Lessons…

- What is common to all these forms of information transmission?
  - An arrow is NOT just a (passive) channel. Rather, it represents an *adjacency* of two agents, allowing them to interact or ***handshake***
  - Agents (like *sender* and *receiver*) *and* media participate in the single indivisible act of communication
    - Sender, Receiver and Media are *all* active agents
- Communications in which a system participates
  - Between system and its environment
  - Between two components of a system
  - … and within a single component

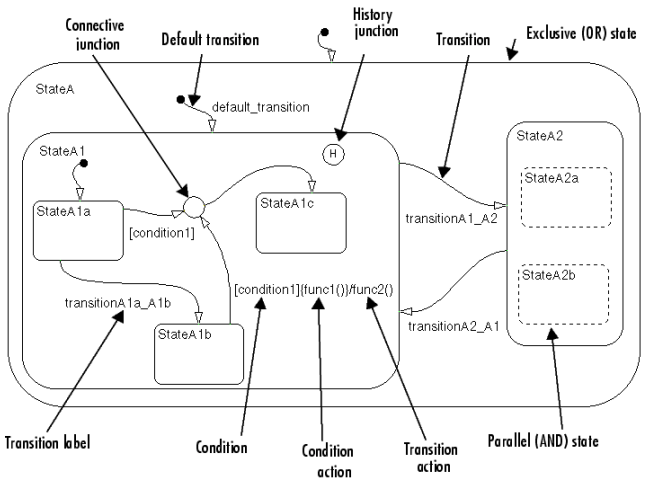
Outline

- **Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation & basic issues
      - Analysis vs. simulation
  - **Specification and Modeling**
    - Concurrency & communication
    - ➔ Implementing processes
      - Matlab's Stateflow and other specification languages
  - **Analysis & Simulation**
    - Stochastic models
    - Transitional semantics
      - SAN formalism
    - Application/Architecture modeling
      - Markovian & non-Markovian analysis
    - Simulation

Implementing Processes: StateCharts

- **Basics**
  - **Specification and design of complex discrete-event, reactive systems (ex. real-time systems, communication and control) – Harel (1987)**
    - Reactive systems: event-driven, continuously reacting to external and internal stimuli
    - Discrete event systems: State changes take place in response to events that occur discretely, asynchronously and often non-deterministically
  - **Extension to state machines formalism**
    - Visual formalism of states and state transitions
    - Extended with hierarchy, concurrency and communication
  - Statecharts = State-Diagrams + Depth + Orthogonality + Broadcast-Communication
  - **Broad acceptance**
    - In academia
      - Numerous extensions were proposed (Hyper/MultiCharts, HierSM, *charts, etc)
      - Parts of UML are based on Statecharts
    - In Industry
      - Boeing (help simulate landing unmanned space craft), Motorola (uses to run speed tests on its products), TI (uses to bridge the gap between R&D and product development)

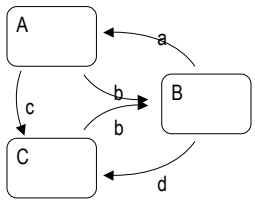
StateCharts Terminology



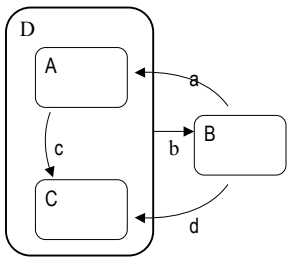
- **State** – a mode of the event-driven system
  - AND States – states that are concurrently active
  - OR States – exactly one state is active within a group of OR states
  - History – a state that remembers the current state of a system
- **Transitions** – indicate how the system changes states
  - Guard – used to set a condition
- **Events** – external change on system inputs
  - Action – system's output, external changes caused by the system

Hierarchy

- **Why?**
  - FSMs lack modularity for designs of large and complex systems
  - Statecharts exhibit substantial descriptive economy (exponential compared to state machines)



Typical FSM



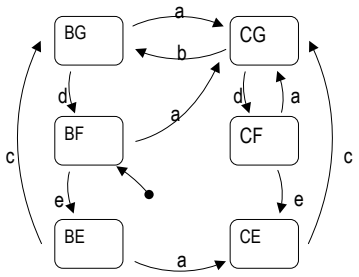
Statechart

Plain English

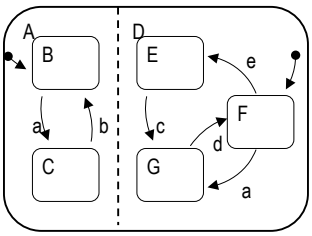
- To be in state D the system must be precisely in one of A or C
- A and C are OR states

Concurrency

- **Why?**
  - **Avoid state explosion of conventional FSMs describing complex concurrent systems**
- **Example**
  - **dashed line splitting a state into concurrent (or *orthogonal*) substates**



Typical FSM



Statechart

Plain English

- Both states A *and* D are active
- But notice… A's and D's substates are OR-states, thus the actual possibilities are (B,E), (B,F), (B,G), (C,E), (C,F), and (C,G)

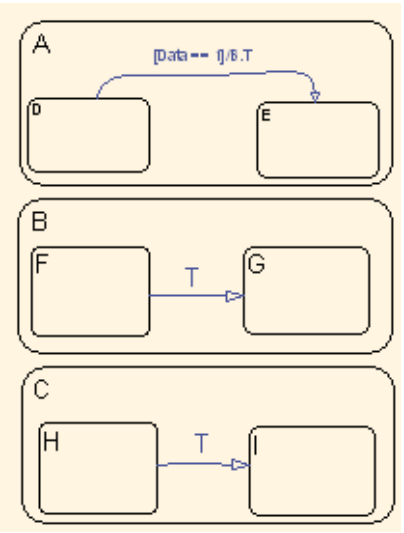
Concurrency

Details

- **Synchronization**
  - Achieved with Broadcast Events
  - Single event causing two simultaneous transitions
  - Modeled with same event label in multiple concurrent states
- **Timing**
  - Synchronous time model – system executes a *single step* every time unit (e.g. digital systems)
  - Asynchronous time model – external changes can occur at any time
- **Race conditions and deadlocks**
  - Read-write racing is resolved by executing writes before reads
  - Write-write races are detected and reported to the designer
- **Non-determinism**
  - Resolved arbitrarily or requires user's input
  - Higher-level transitions take priority over internal transitions

Communication

- **Matlab-style communication**
  - **Direct event broadcasting and implicit communication**
  - **One state generates an event and it's sensed by all other states**
  - **Actions can be directed toward specific states**
    - Initial state of system: {D,F,H}
    - When [Data == 1] evaluates to true, the B.T action in state A becomes active
    - Subsequently, the T event is triggered in state B, but *not in C*
    - Final state of system: {E,G,H}



Overview of Specification Languages

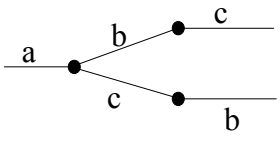
- **Spectrum**
  - **Specification logics**
    - Propositional logic (built up from propositions; $\cap$, $\cup$, $\neg$, $\rightarrow$, …)
    - Temporal logic (Pnueli 1977)
      - *eventually* P, i.e. P holds in at least one of the future states of the system
  - **Process algebra**
    - CCS (Milner 1980, 1989)
    - CSP (Hoare 1985)
    - LOTOS (1988)
  - **FSM**
    - SDL (1988)
  - **Petri nets (60's)**
  - **Synchronous languages (synchrony hypothesis)**
    - Statecharts (Harel 1987)
    - Esterel (Berry & Gonthier 1988)
- **Fundamental issue: *time***

Why is Real-Time so Difficult?

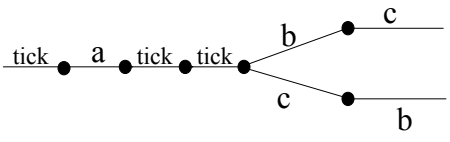
- Proving correctness may be tedious
  - This is essentially a formal verification problem
  - In some cases, other formalisms (e.g. temporal logic) may be more appropriate
- Performance is essentially an implementation-related issue
  - … but, on the other hand, we want formal specifications be abstract!
    - Formal specifications are -usually- bounded to the highest level of abstraction
- Bringing (real-) time into the picture requires the highest level of abstraction to be reconciled against the lowest level!
  - This is very difficult!
  - Solution? Separation of concerns!?
    - Specification of behavior vs. specification of constraints (separation between behavior and performance aspects)
    - Behavioral time vs. real-time

The Space of Real-Time

- Qualitative time
  - Pure event ordering
  - No explicit reference to any quantitative measure of time
  - Are temporally closed (no external constraints to be met)
  - Programming languages
- Quantitative time
  - Timed event ordering
  - Provides a quantitative measure of time, not necessarily external view of time (e.g. “tick” may be one processor cycle
  - (Still) temporally closed



a; (b || c)



σ; a; σ^2; (b || c)

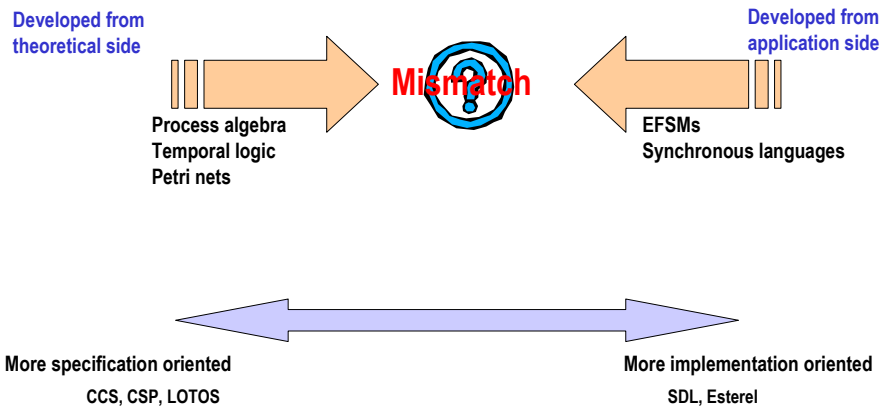
The Space of Real-Time

- Real-time
  - Non-system, human time (e.g. 5 sec)
  - Correctness depends on both the logical results of computation and the (real)-time when they're produced
  - Are temporally open (performance and correctness are tightly related)
  - Example: aircraft control systems
- Hard and soft real-time
  - Hard: exhibit strong failure conditions
    - "the system must always do this activity before time t"
  - Soft: certain degree of flexibility in their timing behavior (e.g. distributed MM systems)
    - "the system should perform this activity before time t in 90% of the cases"
    - Delay variance (jitter)
    - QoS (e.g. frames/sec, end-to-end latency, BER, etc…)

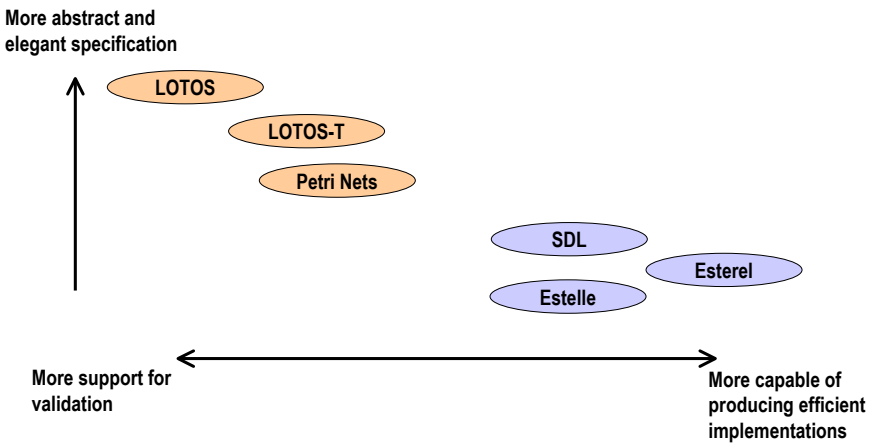
Requirements Imposed on Specification

- Expressiveness (usability and naturalness)
  - Support abstract specification
  - Enable modularity and compositionality in specification
  - Avoid over specification
  - Have viable validation techniques and synthesize efficient implementations from specifications
- Concurrency
  - Usually achieved by mapping concurrent behavior to sequential behavior using interleaving semantics
- Interaction
  - Synchronous, asynchronous, or broadcast communication
  - Unpredictable behavior!
- Etc.

Overall Assessment



Comparison According to Usability



Outline

- **Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation & basic issues
      - Analysis vs. simulation
  - **Specification and Modeling**
    - Concurrency & communication
    - Implementing processes
      - Matlab's Stateflow and other specification languages
  - ➔ **Analysis & Simulation**
    - Stochastic models
    - Transitional semantics
      - SAN formalism
    - Application/Architecture modeling
      - Markovian & non-Markovian analysis
    - Simulation

Stochastic Processes

- **A family of r.v. {X(t), t ∈ T}**
  - **T is the *index set (time)***
    - T is continuous: continuous (time) stochastic process
  - ***State space*: the set of all possible values that X(t) can assume**
    - X is continuous (real-valued): continuous (state) space

- **Markov process = Stochastic process + Markov property**
  - **X(t) satisfies the Markov or memoryless property**
    - The time spent in a state of a MC (sojourn time) must satisfy the memoryless property: at any time t, the remaining time that the chain will spend in its current state is independent of the time already spent in that state
    - If T is continuous, then the *sojourn time must be exponentially distributed*; if T is a discrete, then the sojourn time must be geometrically distributed
  - $p\{X(t) \leq x \mid X(t_0) = x_0, X(t_1) = x_1, \ldots, X(t_n) = x_n\} = p\{X(t) \leq x \mid X(t_n) = x_n\}$

Example

- **Stochastic models: behavior is random**
  - Unpredictable when viewed individually, but predictable when viewed in large numbers
  - ***Average behavior***

Lag-One MC

- **{X(t)} is a MC**

$$p(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_1) = x_1) =$$
$$p(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n)$$

- **{X(t)} is time homogeneous**
  - **Transition rates between states are independent of the time at which the transitions occur**
- **{X(t)} is irreducible**
  - **All states can be reached from all other states by following the transitions of the process**
- **{X(t)} is stationary**

Probability Distributions

- **Examples**
  - **A fair die (uniform distribution)**
    - {1/6, 1/6, …, 1/6} then the avg. value is (1/6) + (2/6) + … + (6/6) = 21/6 = 3.5
  - **Buffer length (non-uniform distribution)**
    - $L_{avg} = 0 \cdot \pi_0 + 1 \cdot \pi_1 + 2 \cdot \pi_2 + \ldots + (n-1) \cdot \pi_{n-1}$

- **An important distribution: *Negative Exponential* (continuous)**
  - $F(x) = 1 - e^{-\mu x}; \quad x \geq 0$
  - **Expectation of an exponential r.v. w/ parameter μ is $1/\mu$**
  - **Note: this is often used to denote *time* which elapses until some event occurs (e.g. arrival of a request at a computer system, etc.)**
  - **Nice mathematical properties**

Weather Example

- **Matricial representation**
  - **state 1: rainy (r)**
  - **state 2: cloudy (c)**
  - **state 3: sunny (s)**

$$Q = \begin{array}{c} \\ r \\ c \\ s \end{array} \begin{bmatrix} \overset{r}{0.8} & \overset{c}{0.15} & \overset{s}{0.05} \\ 0.7 & 0.2 & 0.1 \\ 0.5 & 0.3 & 0.2 \end{bmatrix}$$

- **Note: elements in Q represent conditional probabilities**
  - **For instance, the entry p_{32} tells us that the probability that tomorrow is cloudy, given that today is sunny, is 0.3**

Steady-State Distribution

- **Chapman-Kolmogorov equations**
  - **n-step transition probability matrix**

$$p_{ij}^{(n)} = p\{X_{m+n} = j \mid X_m = i\}$$

$$p_{ij}^{(n)} = \sum_{all\ k} p_{ik}^{(l)}\, p_{kj}^{(n-l)}$$

- ***Theorem*: a steady-state probability distribution p, exists for every time homogeneous, finite, irreducible MC. Moreover, this is the same as the limiting distribution of the chain.**
  - $\pi^{(n)} = \pi^{(0)}\, Q^{(n)}$
  - $\pi = \lim_{n\to\infty} \pi^{(n)}$

Return to Our Weather Example

$$Q^2 = \begin{matrix} r \\ c \\ s \end{matrix}\begin{bmatrix} 0.770 & 0.165 & 0.065 \\ 0.750 & 0.175 & 0.075 \\ 0.710 & 0.195 & 0.095 \end{bmatrix}$$

$$Q^{\infty} = \begin{matrix} r \\ c \\ s \end{matrix}\begin{bmatrix} 0.7625 & 0.16875 & 0.06875 \\ 0.7625 & 0.16875 & 0.06875 \\ 0.7625 & 0.16875 & 0.06875 \end{bmatrix}$$

Outline

- **Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation & basic issues
      - Analysis vs. simulation
  - **Specification and Modeling**
    - Concurrency & communication
    - Implementing processes
      - Matlab's Stateflow and other specification languages
  - **Analysis & Simulation**
    - Stochastic models
    - ➔ Transitional semantics
      - SAN formalism
    - Application/Architecture modeling
      - Markovian & non-Markovian analysis
    - Simulation

Transitional Semantics

- **Labeled Transition System (LTS)**

$$(S, T, \{\xrightarrow{t} : t \in T\})$$

where S = set of states, T = set of transition labels, and $\rightarrow$ is a transition relation $\rightarrow \subseteq S \times S$ for each $t \in T$

E (agent expressions) **Act (actions)**

An LTS can be thought as an automaton w/o a start state or accepting states!

- **Operational semantics of PA for performance analysis**
  - **associate a r.v. which represents a duration, to every action type**

$$(C, Act, \{\xrightarrow{(\alpha, r)} : (\alpha, r) \in T\})$$

action **rate $\in R^+$**

The Basic Language

- Idea: associate a r.v. which represents a duration, to every action type. When enabled, a = (α,r) will delay for a period determined by its associated distribution function $F_a(t) = 1 - e^{-rt}$
  - Setting a timer whenever the activity becomes enabled

This is a simplification!

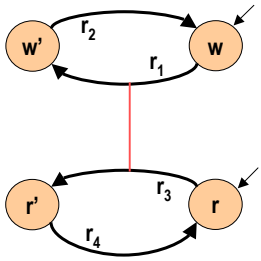
- Example

synchronization

worker = (use, r_1).(idle, r_2).worker

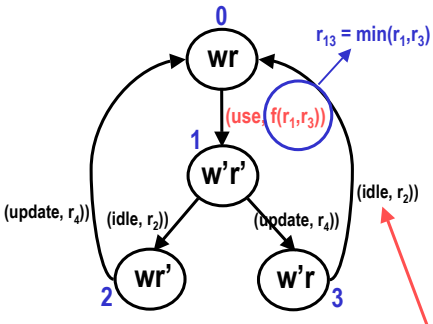
resource = (use, r_3).(update, r_4).resource

system = worker $|_{use}$ resource



R. Marculescu

Example



$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} -r_{13} & r_{13} & 0 & 0 \\ 0 & -(r_2+r_4) & r_2 & r_4 \\ r_4 & 0 & -r_4 & 0 \\ r_2 & 0 & 0 & -r_2 \end{pmatrix} \end{array}$$

$$\pi \cdot Q = 0 \qquad \sum_i \pi_i = 1$$

Assume:
(use, r_1) = 2
(use, r_3) = 6
(idle, r_2) = 2
(update,r_4) = 8

Then:
π_0 = 20/41
π_1 = 4/41
π_2 = 1/41
π_3 = 16/41

Also:
(ρ = utilization)
ρ_0 = 1
ρ_1 = 1
ρ_2 = 1
ρ_3 = 0

Then:
$U_{res} = \rho_0 \pi_0 + \ldots +$
$\rho_2 \pi_2 = 60\%$

R. Marculescu

The SAN Modeling Paradigm

- **Highest level of abstraction**
  - **representations are simple, intuitive, not confined to any hw/sw implementation**
  - **System: a set of concurrent communicating processes**
  - **Communication: *event* and *wait* statements**

- **Idea**
  - **Performance modeling based on SANs**
    - Application is a process graph w/ processes active concurrently
    - Process graph translates into a network of automata
  - **Model evaluation for steady-state regime**
    - Use tensor products (representations remain compact!)
    - Used to compute latency, utilization, response time, …
  - **Big advantage: SAN analysis is much faster than simulation!**

Continuous-Time SANs

$$Q = \begin{bmatrix} -\sigma_{0,0} & \sigma_{0,1} & \sigma_{0,2} & \cdots \\ \sigma_{1,0} & -\sigma_{1,1} & \sigma_{1,2} & \cdots \\ \sigma_{2,0} & \sigma_{2,1} & -\sigma_{2,2} & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \qquad \sum_{j \neq i} \sigma_{i,j} = \sigma_{i,i}$$

$$\sigma_{i,i} = \lim_{t \to 0} \frac{1 - p_{i \to i}}{t} = -p'_{i \to i}$$

$$\sigma_{i,j} = \lim_{t \to 0} \frac{p_{i \to j}}{t} = p'_{i \to j}$$

How Do Automata Interact?

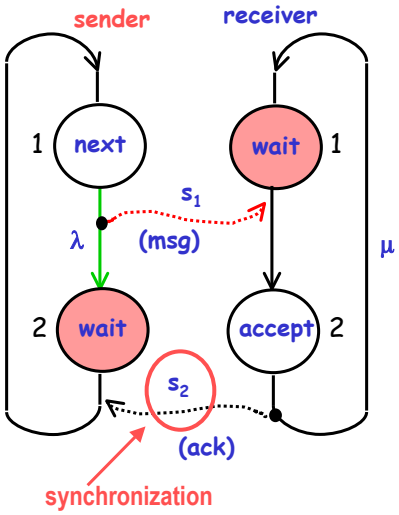
- Synchronizing transitions
  - May alter the state of possible many automata
  - Transitions that are not synchronized are local
- Functional transitions
  - Affect the state of a single automaton

Steady-State Regime

- Global descriptor: $Q = \sum_{j=1}^{2E+N} \bigotimes_{i=1}^{N} Q_j^{(i)}$
  - $\pi \cdot Q = 0$
    $\sum_i \pi_i = 1$
  - Note: we want to avoid the explicit construction of Q!
- Use iterative methods
  - Complexity reduces to

$$\prod_{i=1}^{N} n_i \times \sum_{i=1}^{N} n_i$$

The Ping-Pong Protocol (a.k.a. *Stop and Wait*)



Very important: we talk about MCs and steady-state analysis because we assume exponentially distributed RVs (that is, $F_a(t) = 1 - e^{-rt}$)!

$$(C, Act, \{\xrightarrow{(\alpha,r)} : (\alpha,r) \in T\})$$

action — rate $\in R^+$

The Ping-Pong Protocol (cont'd)

1st automaton — 2nd automaton

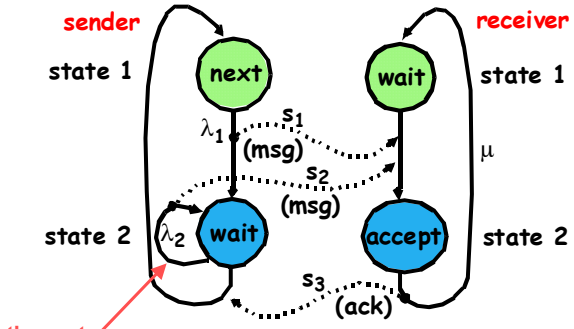
$$Q_{s1} = \begin{bmatrix} 0 & \lambda \\ 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -\lambda & 0 \\ 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Q_{s2} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 \\ \mu & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 \\ 0 & -\mu \end{bmatrix}$$

$$Q = Q_I + Q_{s1} + Q_{s2} \Rightarrow \begin{pmatrix} -\lambda & 0 & 0 & \lambda \\ \mu & -(\lambda+\mu) & 0 & \lambda \\ 0 & 0 & 0 & 0 \\ \mu & 0 & 0 & -\mu \end{pmatrix}$$

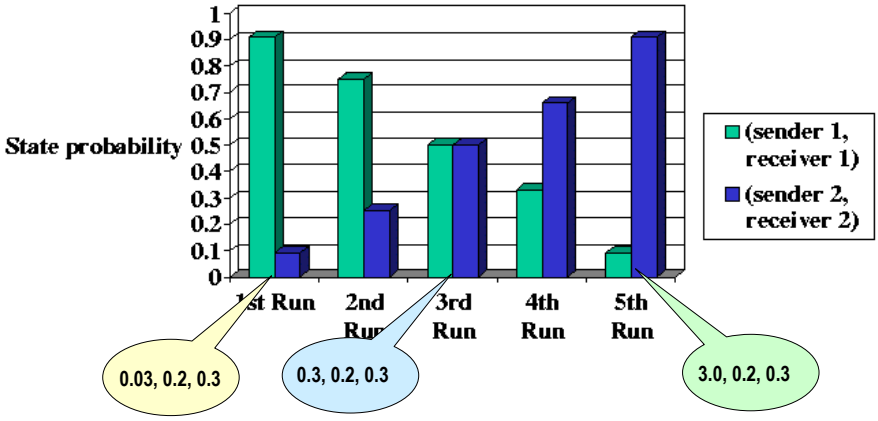
Solve this w/ Matlab

Ping-Pong w/ Timeout



$$Q = \begin{bmatrix} -\lambda_1 & 0 & 0 & \lambda_1 \\ \mu & -(\lambda_1 + \mu) & 0 & \lambda_1 \\ 0 & 0 & -\lambda_2 & \lambda_2 \\ \mu & 0 & 0 & -\mu \end{bmatrix}$$

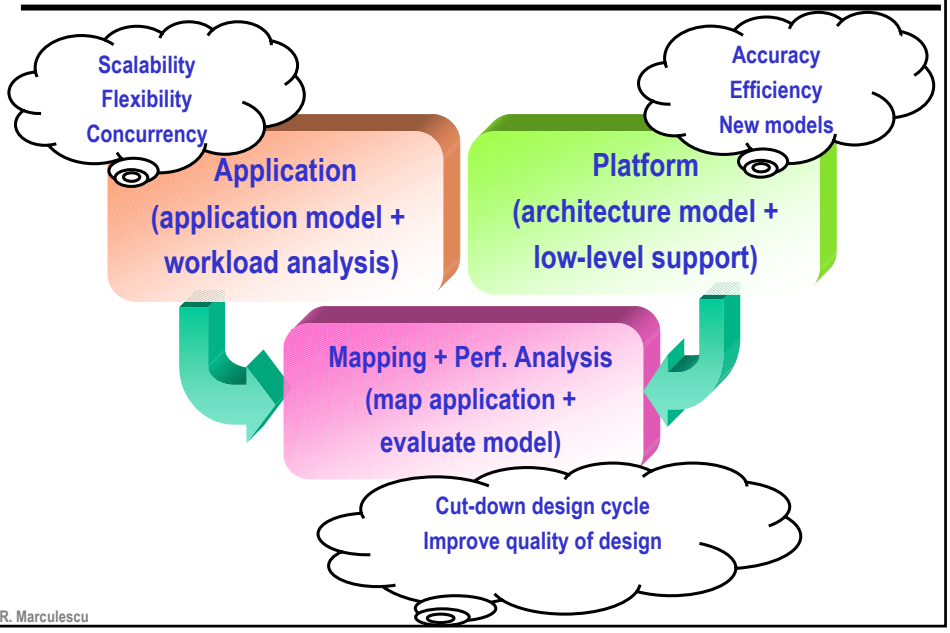
Steady-State Behavior (λ_1, λ_2, μ)



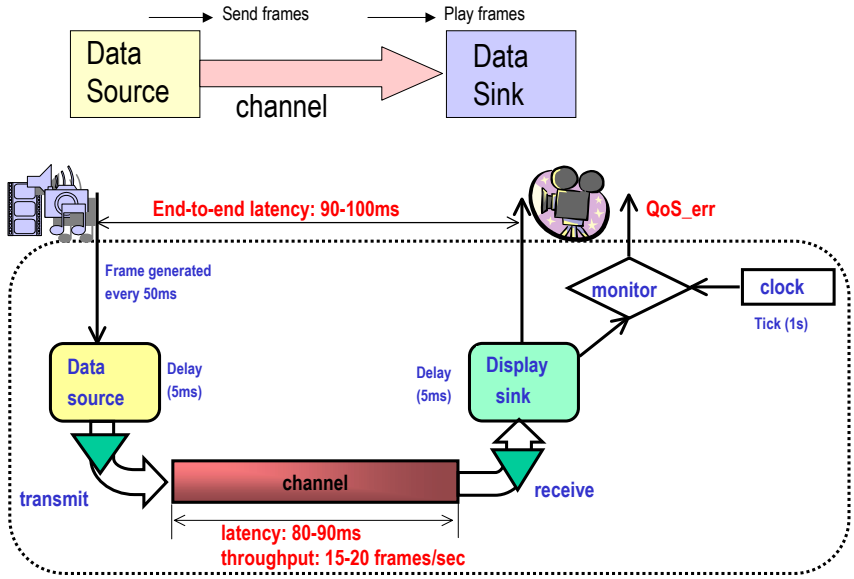
Outline

- **Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation & basic issues
      - Analysis vs. simulation
  - **Specification and Modeling**
    - Concurrency & communication
    - Implementing processes
      - Matlab's Stateflow and other specification languages
  - **Analysis & Simulation**
    - Stochastic models
    - Transitional semantics
      - SAN formalism
    - ➔ Application/Architecture modeling
      - Markovian & non-Markovian analysis
    - Simulation

The Big Picture: Separation of Concerns

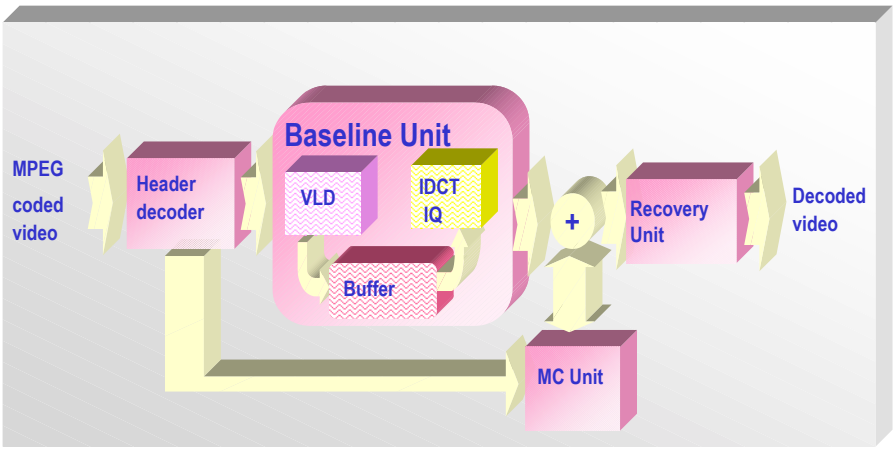


Multimedia Stream Abstraction



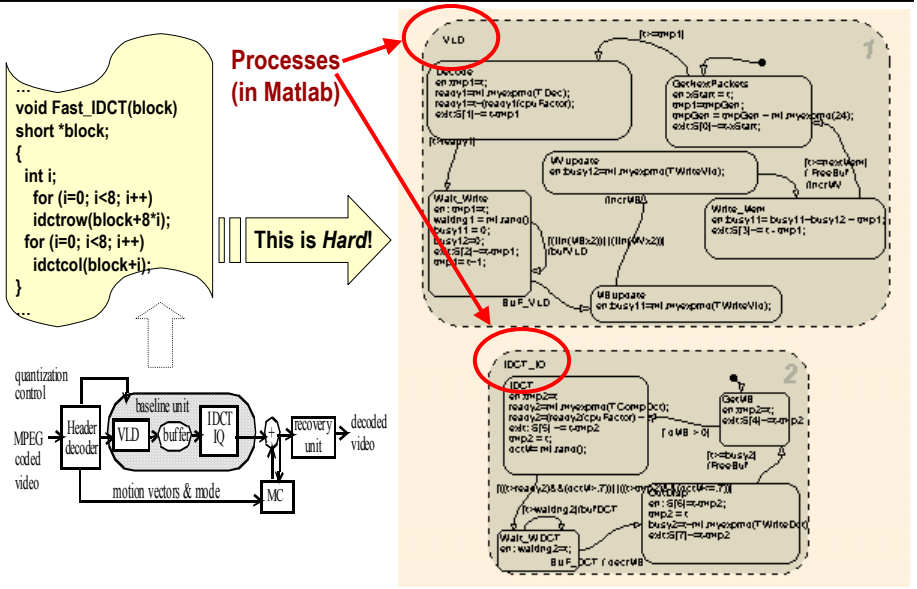
R. Marculescu

MPEG-2 Video Decoder

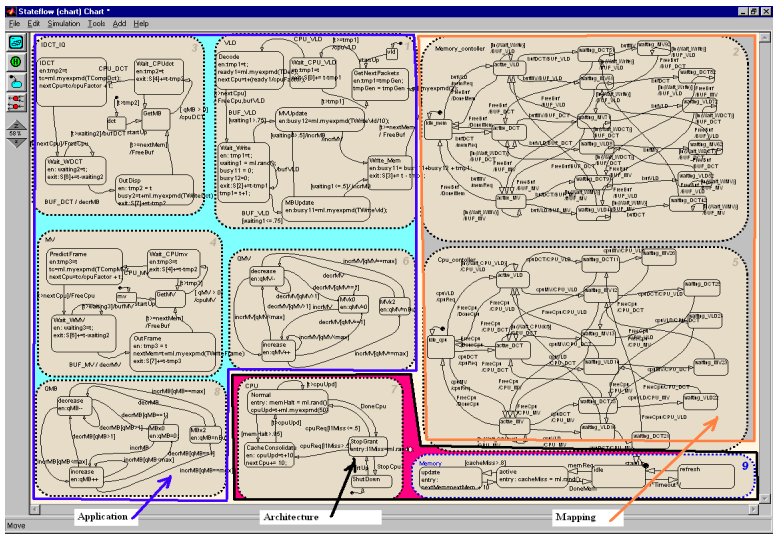


R. Marculescu

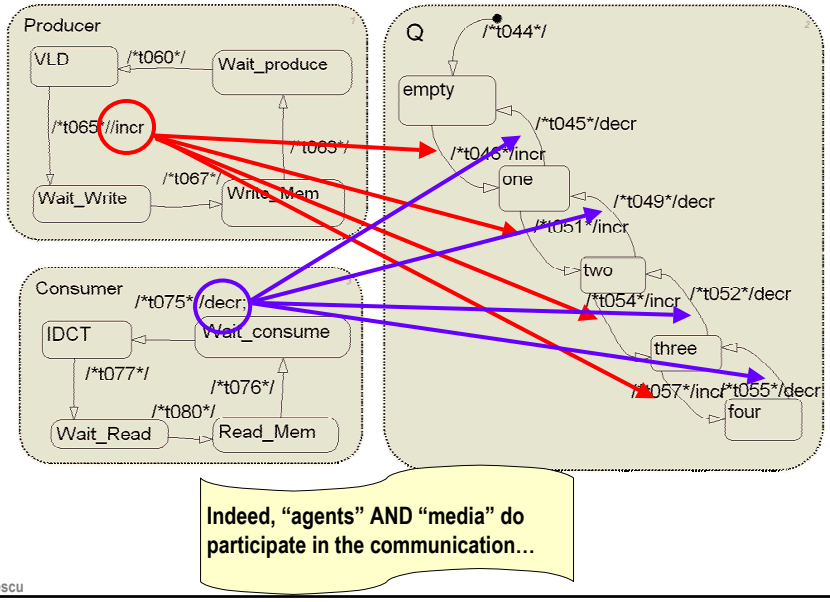
How Do Actually Processes Look Like?



MPEG-2 Stateflow Model (screen shot)



Application Modeling



From Automata to SANs

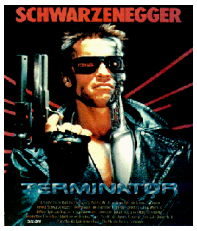
Local Transition Matrices

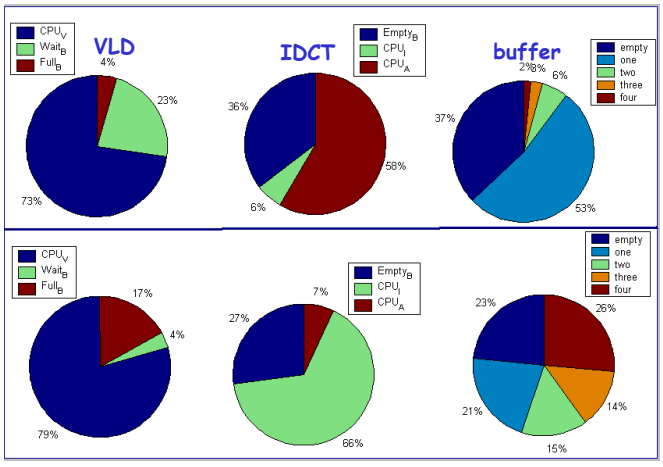
	VLD	Wait_Produce	Write_Mem	Wait_Write
VLD	0	0	0	0
Wait_Produce	λ_1	$-\lambda_1$	0	0
Write_Mem	0	λ_2	$-\lambda_2$	0
Wait_Write	0	0	λ_3	$-\lambda_3$

Synchronous Transition Matrices

	VLD	Wait_Produce	Write_Mem	Wait_Write
VLD	0	0	0	λ_4
Wait_Produce	0	0	0	0
Write_Mem	0	0	0	0
Wait_Write	0	0	0	0

MPEG-2 Application Analysis





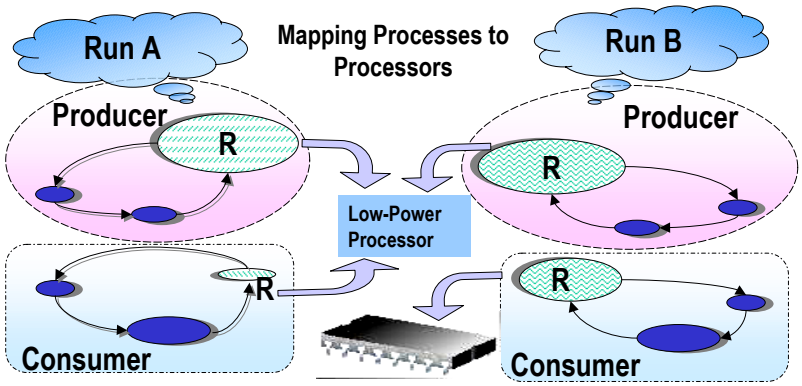
Global Probabilities Distribution

Probabilities		Producer	Consumer	Buffer
Run A	Run B			
0.26	0.60	R[a]	R	0-1
0.62	0.22	R	I	0-1
0.11	0.12	I[b]	I	0-1
0.01	0.01		All others§	

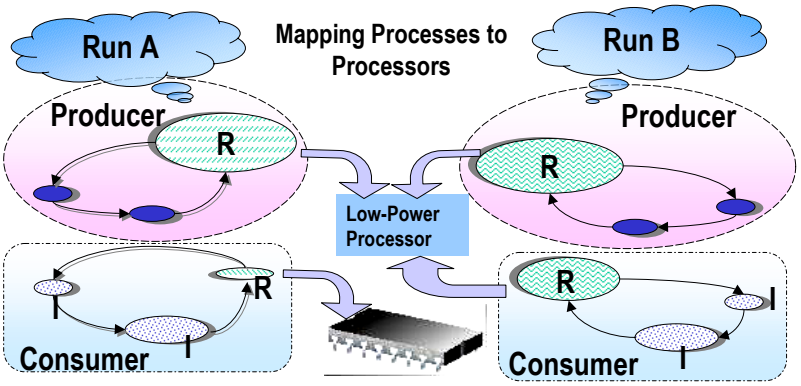
Global Probability Distribution

a.〉 R means the process is actively computing
b.〉 I means the process is idle and is not computing

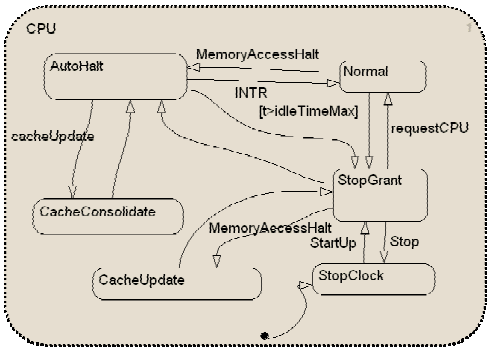
Mapping 1



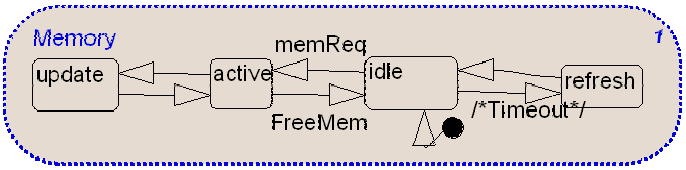
… and Mapping 2



How About Architecture?

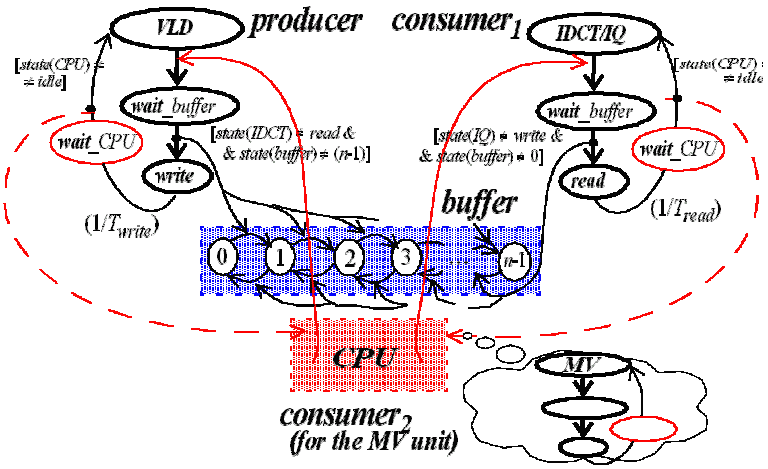


Model of a 486 CPU

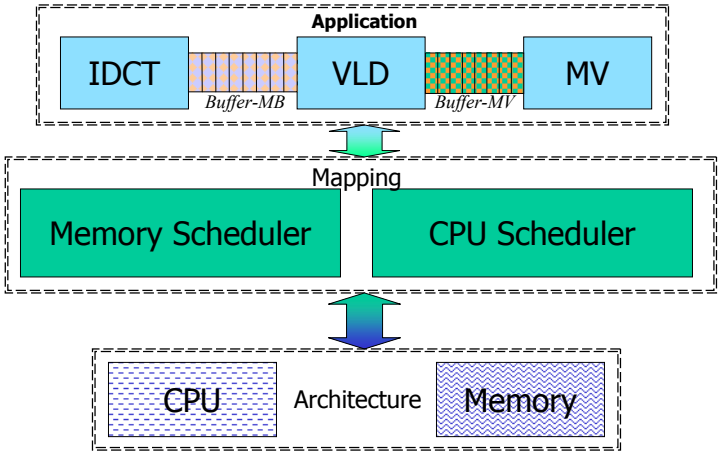


Model of the memory

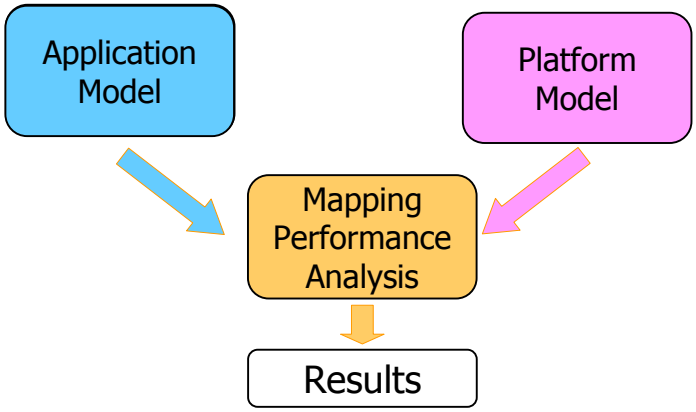
Putting Everything Together



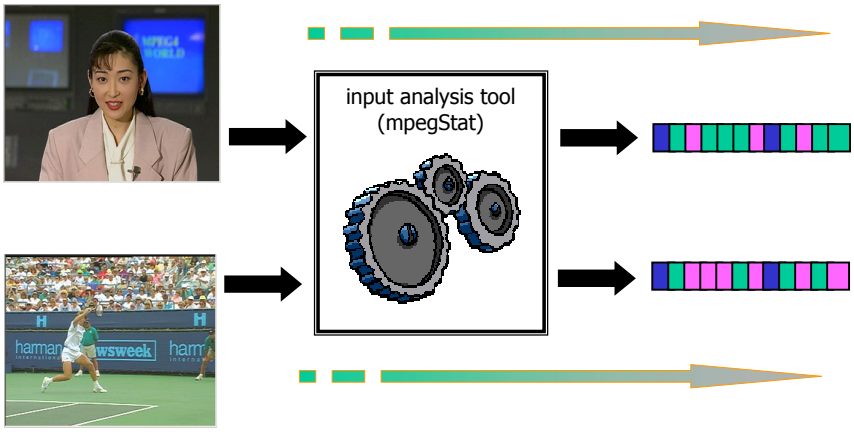
Putting Everything Together



Experimental Setup: The Y-chart

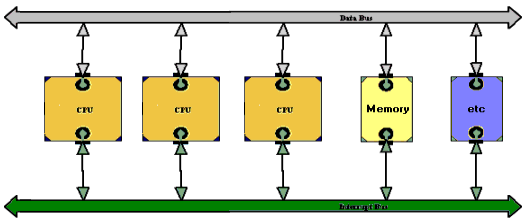


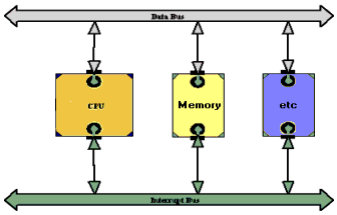
Video Stream Modeling



Platform Models

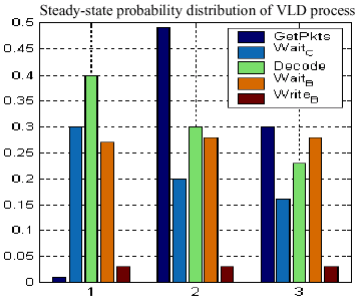
infinite resources

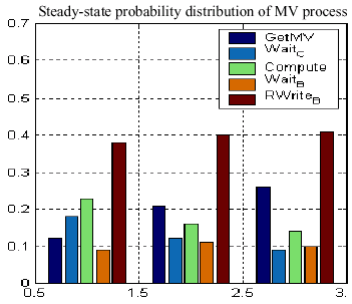


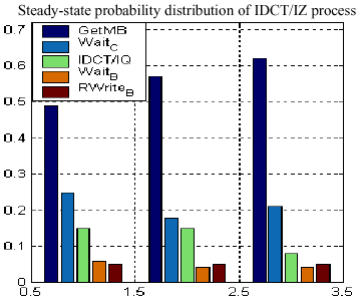


finite resources
CPU-Speed:
$\{ f_0 , 2f_0 , 3f_0 \}$

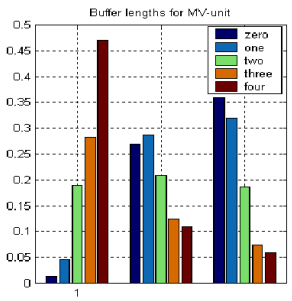
Results

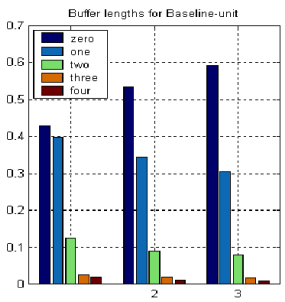


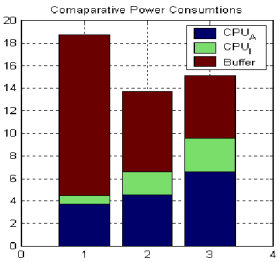




Results



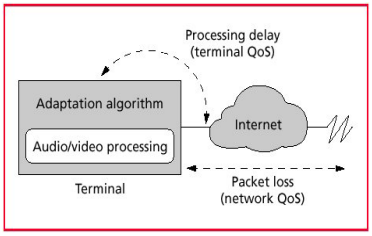




$$P^{(k)} = \sum_{all\ \ i} \pi_i \cdot P_i + \sum_{all\ \ i,j} \lambda_{ij} \cdot P_{ij}$$

Beyond Markovian Analysis: Terminal QoS Approach

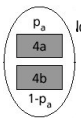
- **Adaptive application:**
  - Variable amount of computation
  - Variable impact on the network performance
  - Variable effect on other applications



- **Analytical framework**
  - **Model the *adaptation process***
    - Select the adaptation levels and feedback protocol
  - **Model the *adaptive application***
    - Create a task graph with several alternate execution path
  - **Characterize the state of processing**
    - We need the task-level resource requirements
    - We need to know the mapping of the tasks and the execution time for each of them

Application Specification

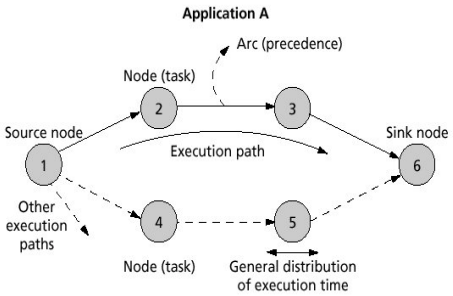
- **Node = task**



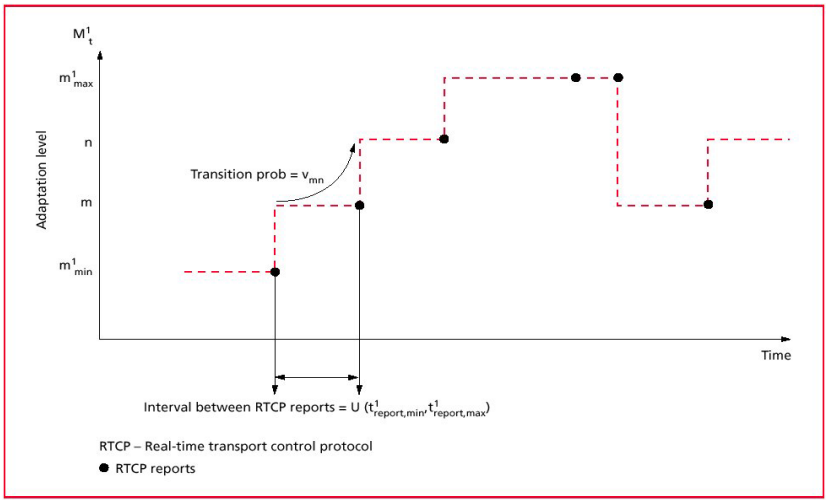
lode with a variable execution time

- **Arc = precedence relation**

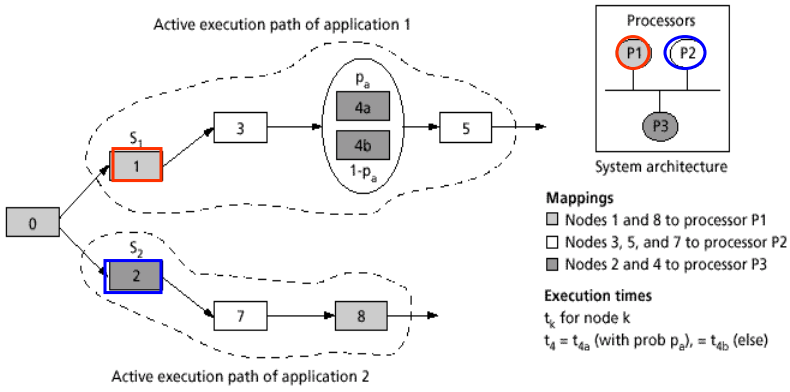
Arcs do not have a time associated. For modeling communication dummy nodes are used.



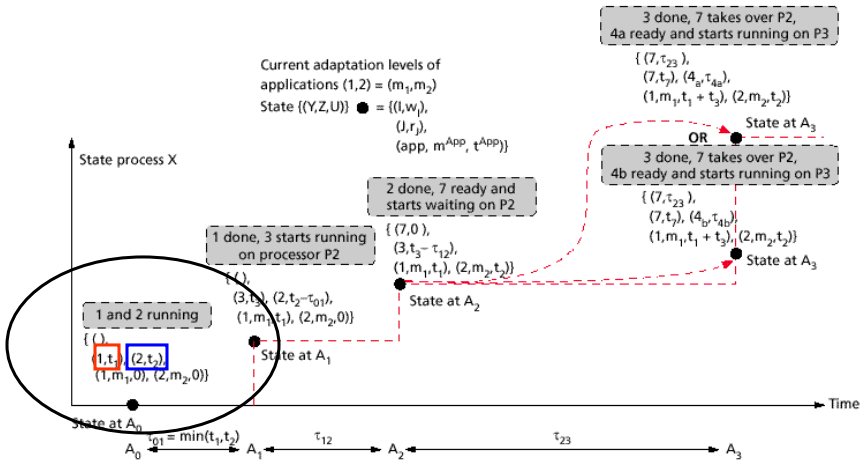
Adaptation Process



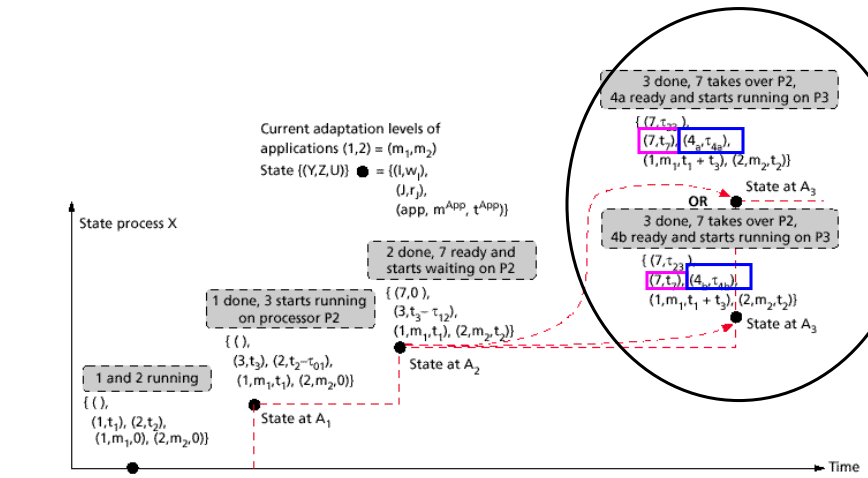
Mapping



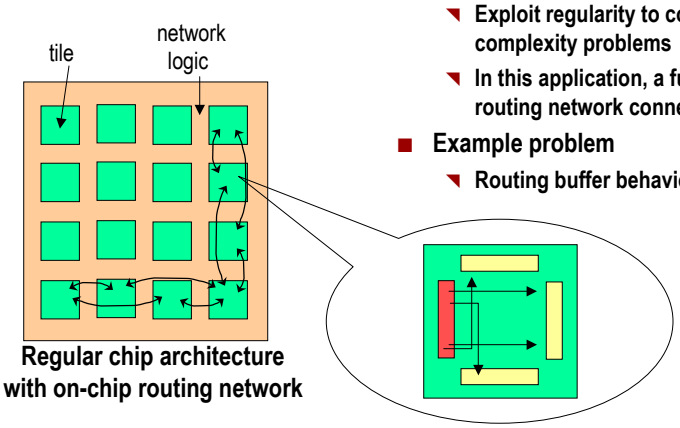
Performance Analysis



Performance Analysis



Another Example of Non-Markovian Analysis

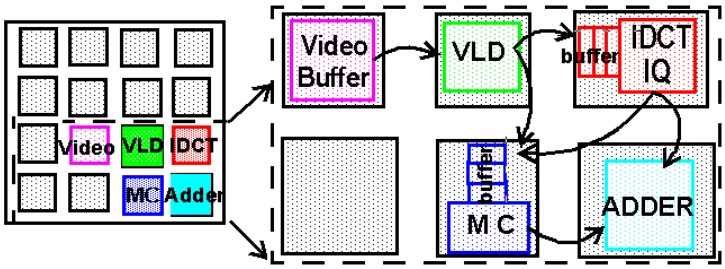


Regular chip architecture
with on-chip routing network

- Idea
  - Exploit regularity to combat growing complexity problems
  - In this application, a full packet routing network connects tiles
- Example problem
  - Routing buffer behavior in each tile

Router needed at each tile--
How much *buffer space* is needed?
What effects for too much? too little?

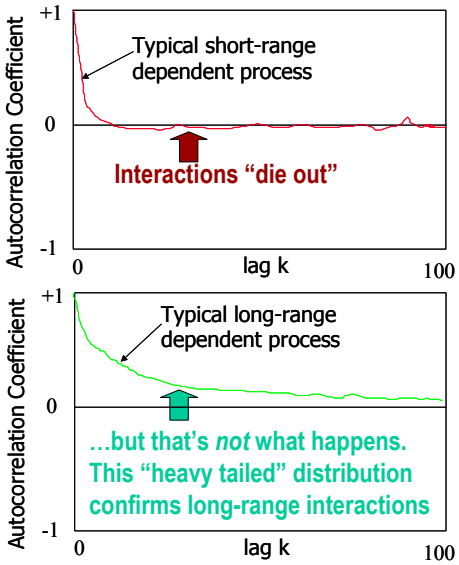
Analysis Example: Regularized MPEG-2



- Mapped MPEG app onto tiles
  - Used routing network for inter-tile communication
- Experiment
  - Collect network traffic traces via simulation (*time stamp*, *#bytes arrived*)
  - Analyze statistical properties of the resulting time series
  - Build formal models of packet size distributions, dependencies

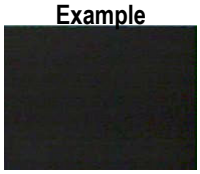
Interesting Result: Long-Range Dependencies

- **Classical expectation**
  - The distribution of the *arrival process* is short-range dependent i.e. exponential
- **Reality**
  - The rate at which autocorrelation decays, is related to the *Hurst parameter* (H)
  - Self-similar (fractal) processes are used to model long range dependence (0.5 < H < 1.0)

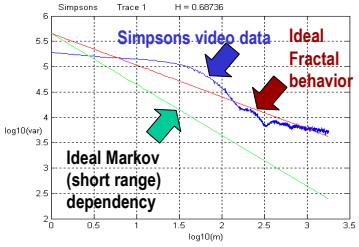


Surprising Result: On-Chip Fractal Behavior

Simpsons Video Example



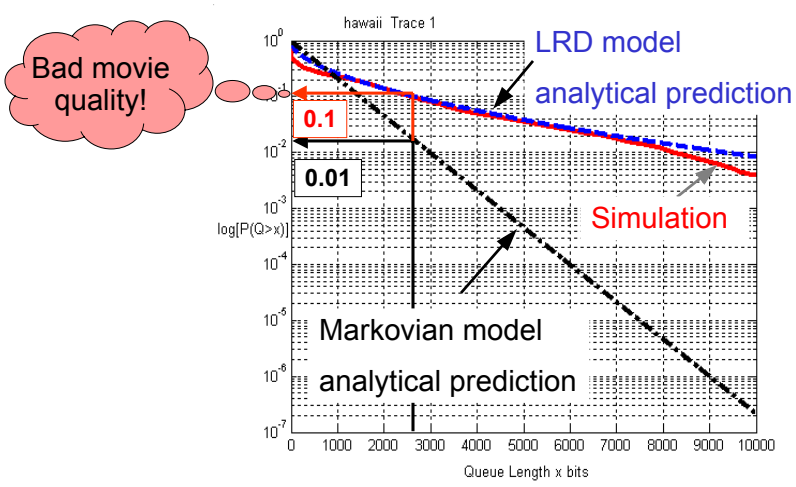
Analysis of One Trace from Regularized MPEG Chip



- The design of on-chip networks is quite unique
  - Buffer space must be kept to a minimum
  - High wire and power efficiency
- *Implications of long-range dependent traffic on on-chip network design*
  - The average delay of a buffer increases sharply at surprisingly low utilization factors
  - If ignored, this results in optimistic performance predictions and inadequate resource allocation
- Does this indicate the death of Poisson modeling?

Buffer Length Prediction

- **Analytical prediction vs. simulation results**

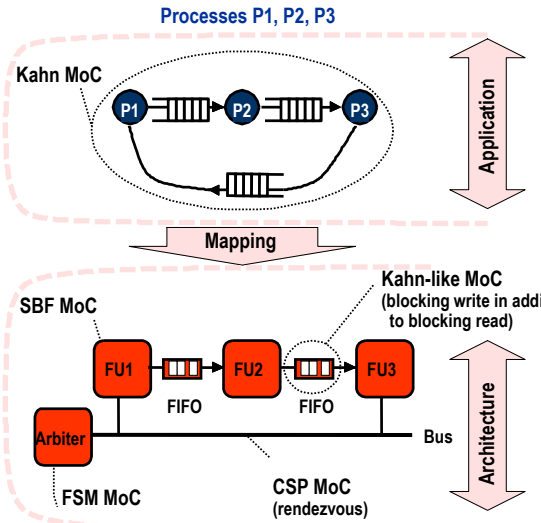


Outline

- **Specification, Modeling, Analysis**
  - **Formal methods in system-level design**
    - Motivation & basic issues
      - Analysis vs. simulation
  - **Specification and Modeling**
    - Concurrency & communication
    - Implementing processes
      - Matlab's Stateflow and other specification languages
  - **Analysis & Simulation**
    - Stochastic models
    - Transitional semantics
      - SAN formalism
    - Application/Architecture modeling
      - Markovian & non-Markovian analysis

    ➔ Simulation

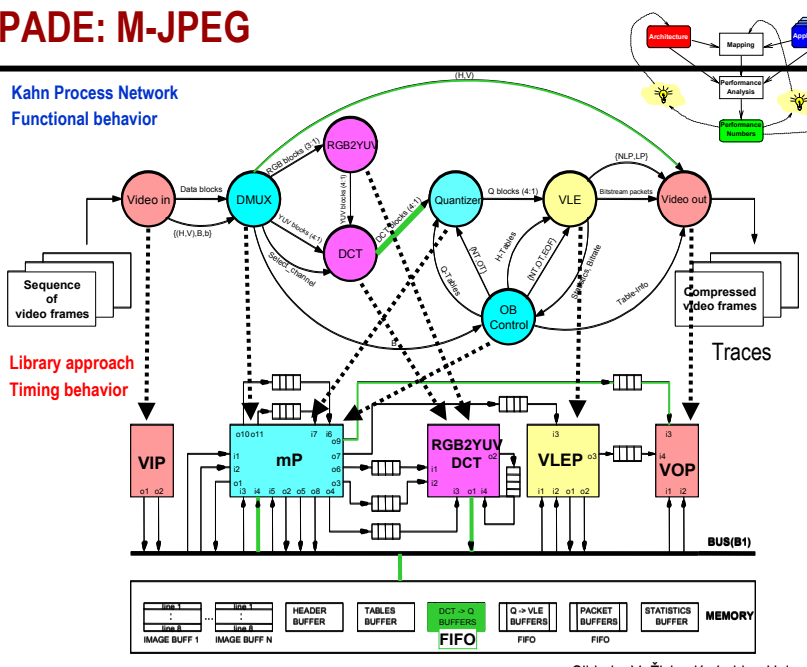
Formal Models in Simulation: MoA and MoC



Slide by V. Živković · Leiden University

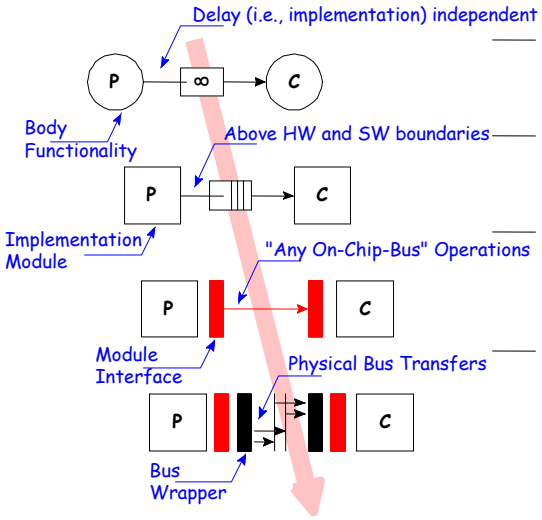
SPADE: M-JPEG

- Kahn Process Network
- Functional behavior
- Library approach
- Timing behavior



Slide by V. Živković · Leiden University

Co-design Simulation and Synthesis - COSY



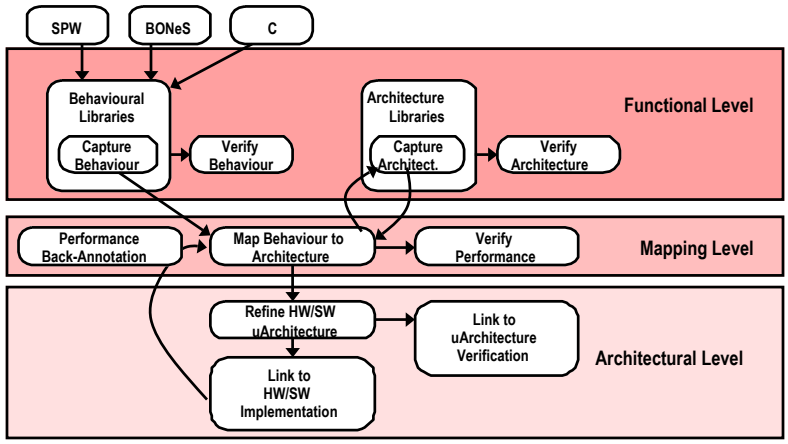
APP
Application Programming Interface

SYS
System Communication Interface

VCI
On-Chip-Bus
Virtual Component Interface

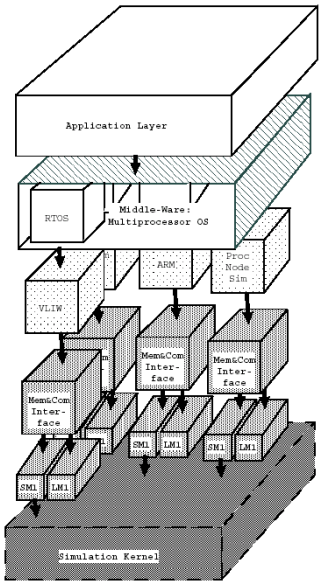
PHY
Physical Bus or Switching
Network Interface

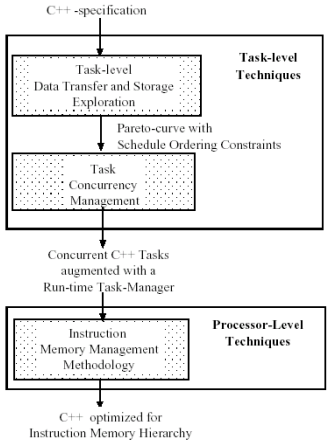
POLIS/VCC



POLIS and VCC (Cadence Design Systems, Inc.)

IMEC's Matador





Summary

- **Formal modeling and analysis of MM systems is possible!**
  - **Aim for specification at the highest level of abstraction**
  - **Exploit the separation of concerns**
    - Function-architecture
    - Computation-communication
  - **Application and architecture modeling**
    - PAs offer a powerful abstraction
    - Statecharts are a strong candidate for process implementation
    - Building and solving the stochastic model is the main challenge
  - **Mapping application onto architecture**
    - Scheduling is an important concern
    - Determine best power/performance tradeoffs
    - Provide quick feedback to the designer
    - Analysis results can be further used at micro-architectural level

Part IV: Communication-Based Design

Outline

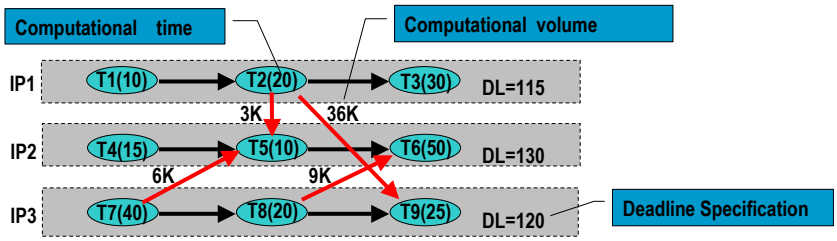
- **Communication-based design**
  - **Node-Centric Perspective**
    - Irregular architectures
      - P2P communication
    - Regular architectures
      - Energy
      - NOC power management
      - Fault-tolerance
  - **Network-Centric Perspective**
    - Channel modeling
    - Power-performance trade-offs

Bus-Based vs. P2P Communication

- **What's wrong with buses?**
  - **Performance**
    - Interconnections become dominant in DSM
    - Huge bandwidth requirements (tens of Gb/s for some applications) (buses are not scalable!)
  - **Power consumption**
    - Expanding market of mobile and other low-power applications
    - Increasing cooling costs (buses consume too much power!)
- **P2P Communication**
  - **Advantages**
    - Faster; no bus contention, no bus arbitration
    - Low-power solution
    - Can be independently optimized
  - **Disadvantage**
    - May need more wiring resources

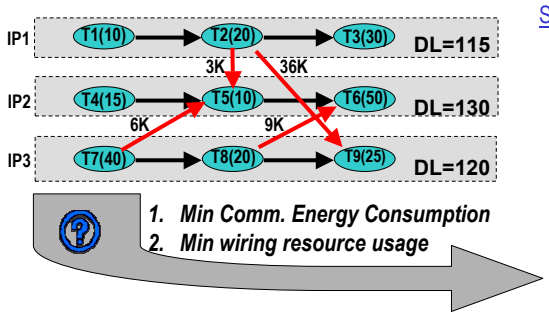
What Are the System Inputs?

- **A set of IPs:**
  - **Hard IP (*Width*length*, provided by different IP providers)**
  - **Soft IP (*Size* provided by synthesis or estimation)**
- **Communication Task Graph (CTG)**



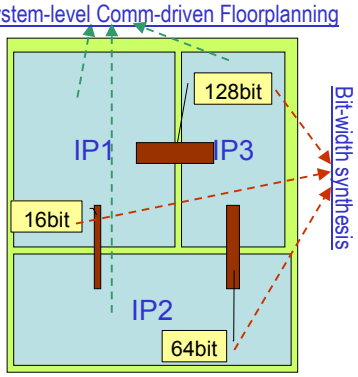
Communication Task Graph

Physical Planning for P2P Communication



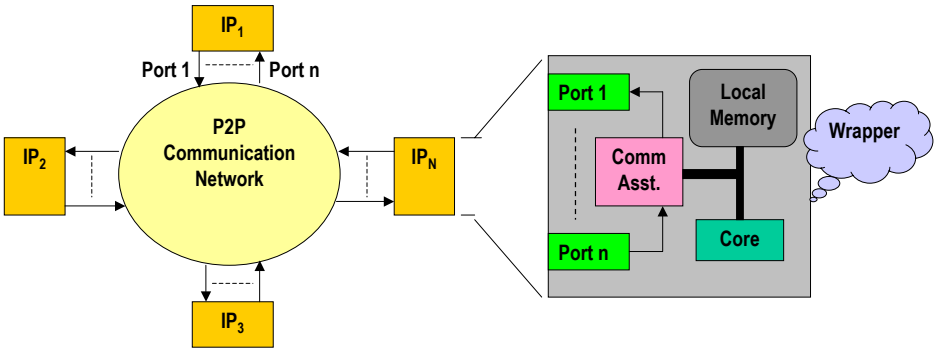
1. *Min Comm. Energy Consumption*
2. *Min wiring resource usage*

System-level Comm-driven Floorplanning



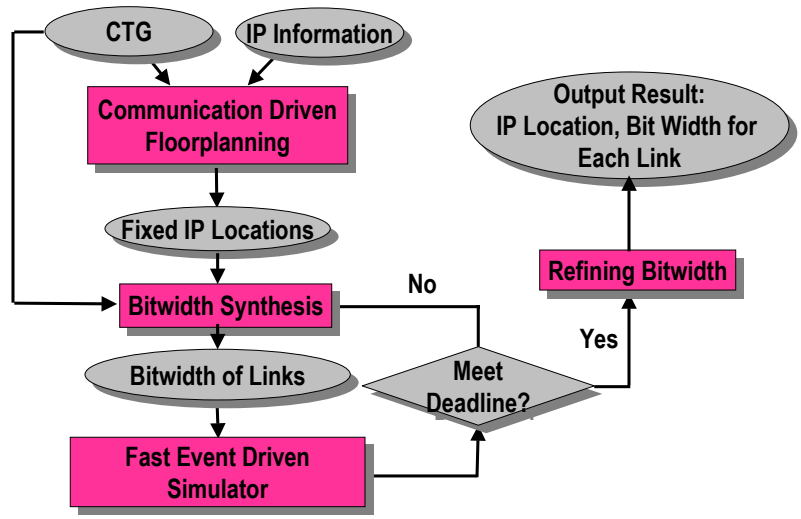
R. Marculescu

The Target Platform



R. Marculescu

The Design Flow



Communication Driven Floorplanning

- **Parameters:**
  - Vol_{ij}: communication volume between IP_i and IP_j
  - D_{ij}: Manhattan distance between IP_i and IP_j
  - Define metric M_{ij}: $M_{ij} = Vol_{ij} \times D_{ij}$
- **Objectives:**
  - *Minimize* communication latency and energy

$$Cost = \sum_{i=1}^{n} \sum_{j=1}^{n} M_{ij}$$

  - *Integrate* it with area minimization

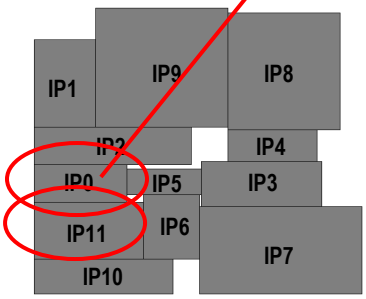
$$Cost = A_{chip} + \lambda \sum_{i=1}^{n} \sum_{j}^{n} M_{ij}$$

A Simple Floorplanning Example

	IP0	IP1	IP2	IP3	IP4	…	IP11
IP0	/	211	522	149	34	…	559
…	…	…	…	…	…	…	…

Communication volume among IPs

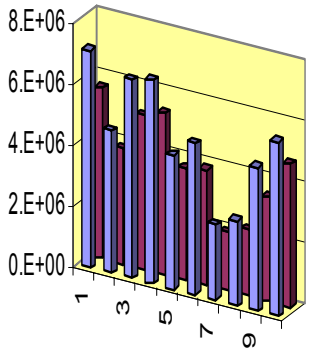
Floorplanning considering only area



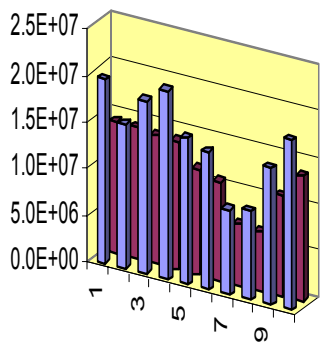
Floorplanning considering communication *and* area

R. Marculescu

Results Using Random Graphs



30 IPs

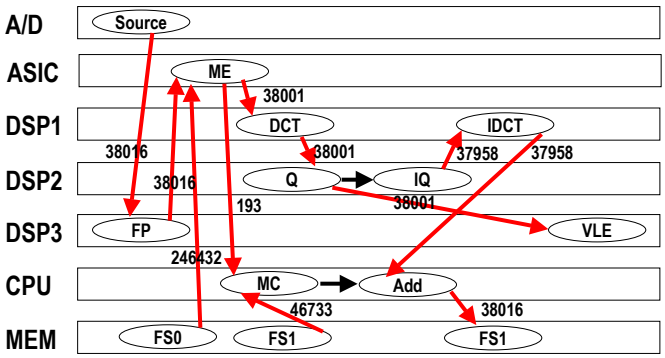


50 IPs

Energy savings (21.33%, 26.34%) with area overhead within 5%

R. Marculescu

MPEG-2 Video Encoder



IP Sizes (Gate counts)

AD converter: 100K, 7.21 mm^2
ASIC: 14K: 1.01mm^2
DSP1: 8x8DCT (33K): 2.38mm^2
DSP2,3: M320C50 (40K): 2.88mm^2
CPU: POWERPC401S (50K):3.61mm^2
128K SRAM: 5.652mm^2

Results Using MPEG-2

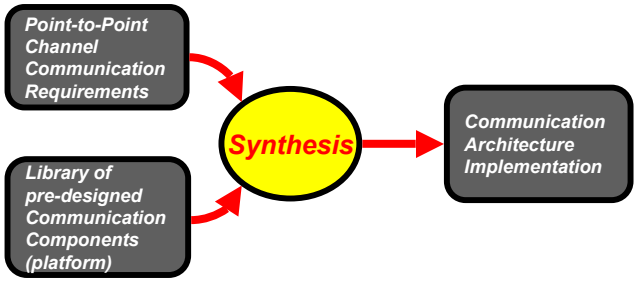
- **Area/Wirelength/Energy comparison**

	Traditional	Communication-driven	Savings
Chip area (mm^2)	26.6805	27.772	-4%
Wirelength(mm)	275980	234200	15.1%
Energy(μJ/frame)	3.17	2.49	21.6%

- **Comparison with bus-based implementation**

	Akiyo	*Toy Box*	*Cup*	*Color Hand*
Bus(μJ/frame)	11.21	8.65	6.26	5.73
P2P(μJ/frame)	2.49	1.81	1.32	1.09
Savings	77.8%	79.1%	78.9%	80.1%

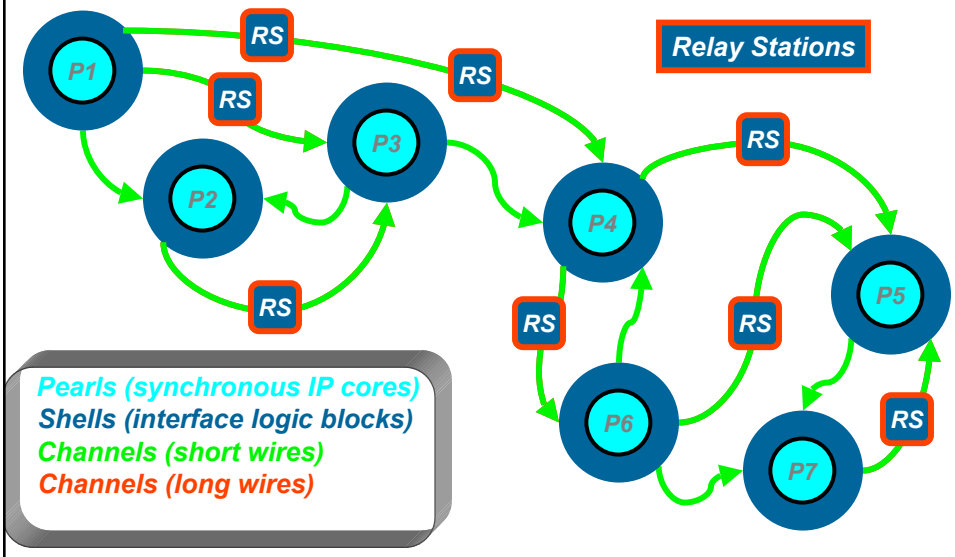
Constraint-Driven Synthesis of Communication Architectures



- System modules communicate by means of P2P channels
- High-level communication constraints for each channel are captured as a Constraint Graph
- Similarly, the characteristics of all components in the Communication Library are captured as a set of feature resources together with their cost figure
- The synthesis result is represented by an Implementation Graph and obtained by solving a constrained optimization problem

Slide by A. S.-Vincentelli · U.C. Berkeley

Latency-Insensitive Design



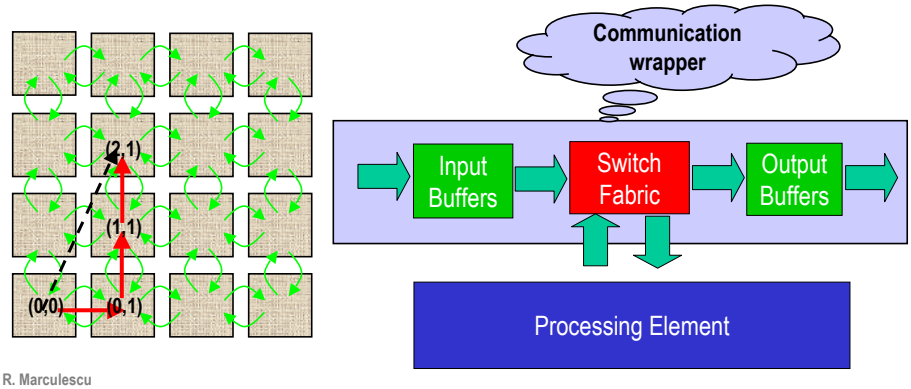
Slide by A. S.-Vincentelli · U.C. Berkeley

Outline

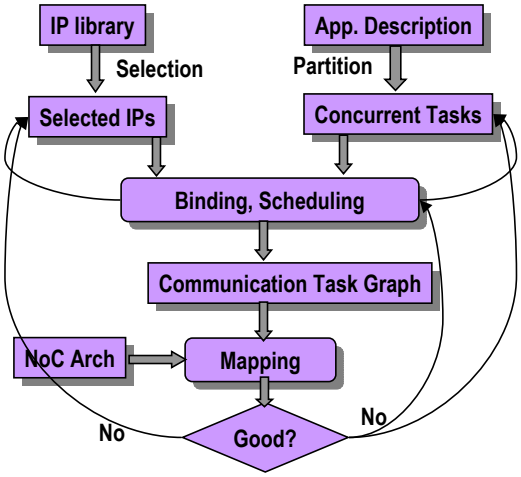
- **Communication-based design**
  - **Node-Centric Perspective**
    - Irregular architectures
      - P2P communication
    - ➔ Regular architectures
      - Energy
      - NOC power management
      - Fault-tolerance
  - **Network-Centric Perspective**
    - Channel modeling
    - Power-Performance trade-offs

Packet-Based On-Chip Communication

- **Performance and power dissipation are two major design constraints**
  - **Interconnections become dominant in DSM era**
  - **Huge bandwidth requirements (tens of Gb/s for some applications)**
    - *Buses are not really scalable and consume too much power!*
  - **Regularized, tile-based network-on-chip architecture**
    - Pros/Cons?

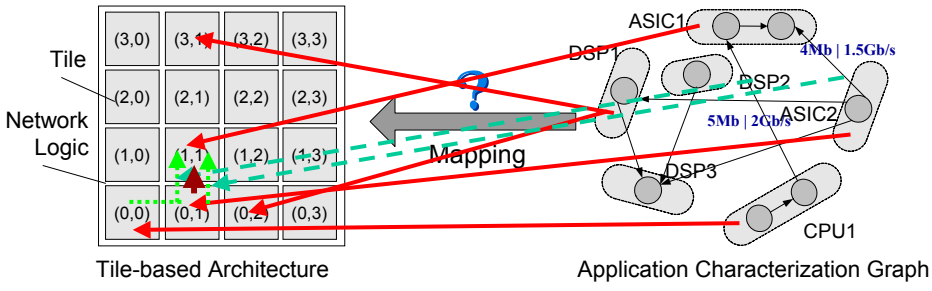


A New Design Flow



R. Marculescu

Energy-Aware Mapping for Tile-based Architectures



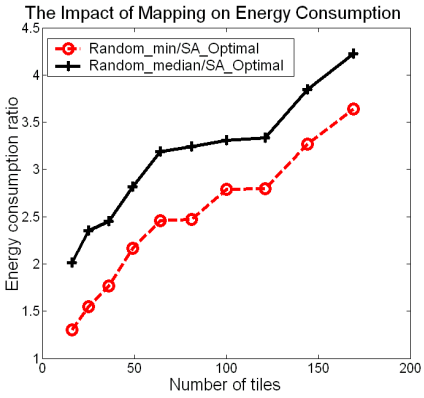
Objective: minimize the total communication energy consumption

Constraint: meet the communication performance constraints (specified by designer)

R. Marculescu

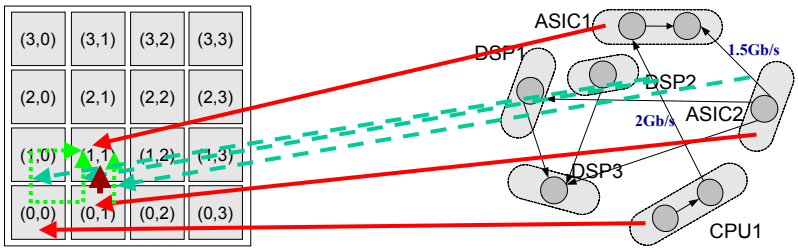
Is The Energy-Aware Mapping Important?

- For a 4X4 tile architecture, there can be 16! mappings
  - Impossible to enumerate
- Mapping can have a huge impact on energy consumption
  - Impact increases as the size of the problem scales



Can We Do Better?

When the link bandwidth is only 3.0Gb/s

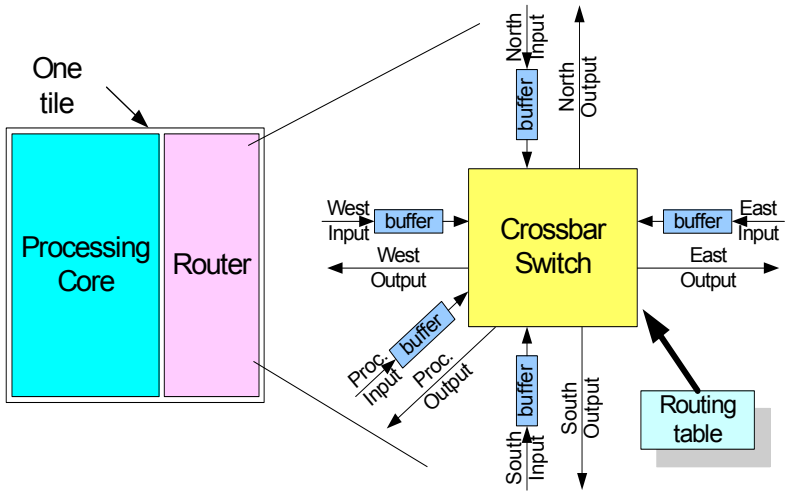


Tile-based Architecture

Communication Task Graph

Exploiting routing flexibility helps expanding the solution space but makes the problem even more complex!

The Tile-based Architecture Platform



The Energy Model

- $E_{bit} = E_{Sbit} + E_{Bbit} + E_{Wbit} + E_{Lbit}$
- **Basic facts**
  - **Negligible buffering energy consumption**
    - Each tile has size in the order of mm^2
    - Buffer implemented using latches or flip-flops
  - **Negligible internal wire energy consumption**
  - **Equation reduced to $E_{bit} = E_{Sbit} + E_{Lbit}$**
  - **With deterministic, minimal routing, we have:**

$$E_{bit}^{t_i,t_j} = n_{hops} \times E_{Sbit} + (n_{hops} - 1) \times E_{Lbit}$$

Problem Formulation

Given an APCG and ARCG with
size(APCG)≤ size(ARCG)

Find a mapping function map( ) from the APCG to ARCG and a deadlock-free, minimal routing function R( ) which minimizes:

$$\min\{Energy = \sum_{\forall a_{i,j}} v(a_{i,j}) \times e(r_{map(c_i),map(c_j)})\}$$

Such that:

$$\forall c_i \in C,\ \ map(c_i) \in T \qquad (1)$$

$$\forall c_i \neq c_j \in C,\ \ map(c_i) \neq map(c_j) \qquad (2)$$

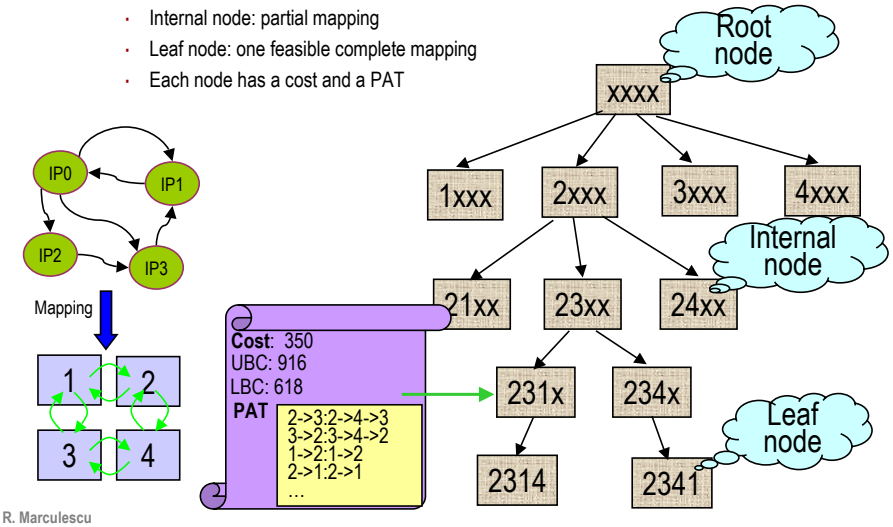
$$\forall \text{ link } l_k, B(l_k) \geq \sum_{\forall a_{i,j}} b(a_{i,j}) \times f(l_k, \mathcal{R}(r_{map(c_i),map(c_j)})) \qquad (3)$$

Where $B(l_k)$ is the bandwidth of link l_k and:

$$f(l_k, p_{m,n}) = \begin{cases} 0 & : \quad l_k \notin L(p_{m,n}) \\ 1 & : \quad l_k \in L(p_{m,n}) \end{cases}$$

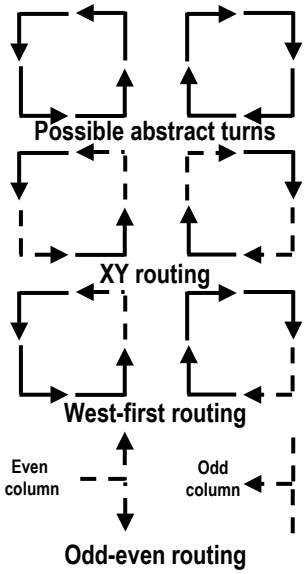
The Energy-Aware Mapping

- This is an NP Problem! (Use a Branch-and-bound algorithm)
  - Searching tree
    - Internal node: partial mapping
    - Leaf node: one feasible complete mapping
    - Each node has a cost and a PAT



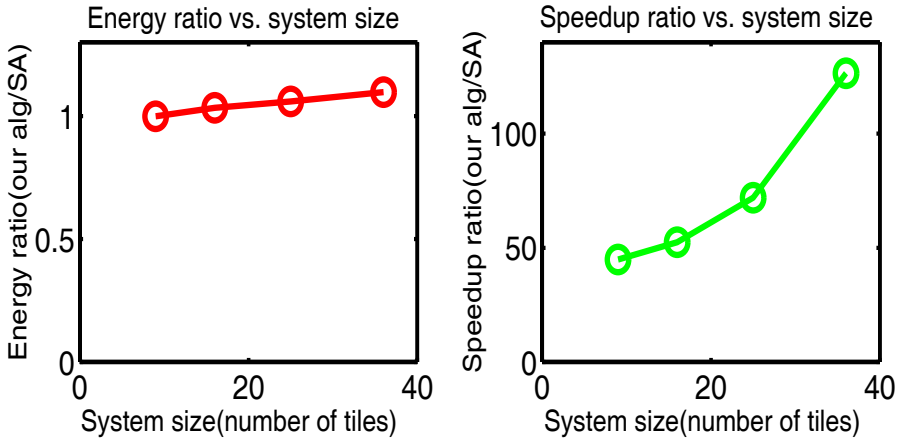
Routing Path Allocation

- XY routing
  - Route packet along X axis first
  - Half of the turns are forbidden
- West-First (WF) routing
  - Turn to west is forbidden
  - ¼ of the turns are forbidden
- Odd-Even (OE) routing
  - Rule1: No EN or ES turn in even-column
  - Rule2: No NW or SW turn in odd-column
  - ¼ of the turns are forbidden



Results - Random Applications

- Four categories of random benchmarks
  - Categories (I,II,III,IV) contain 10 benchmarks with 9, 16, 25 and 36 IPs, respectively
- Use Simulated Annealing (SA) as a reference

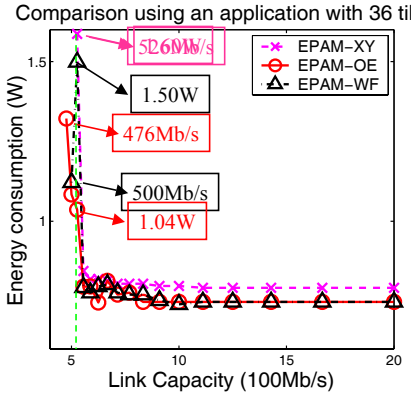


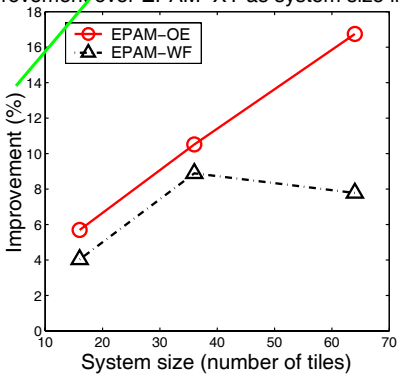
Comparison between EPAM-XY and Simulated Annealing

Does Routing Help?

Comparison using an application with 36 tiles Improvement over EPAM–XY as system size increase

$$\frac{BW_{XY}^{min} - BW^{min}}{BW^{min}}$$

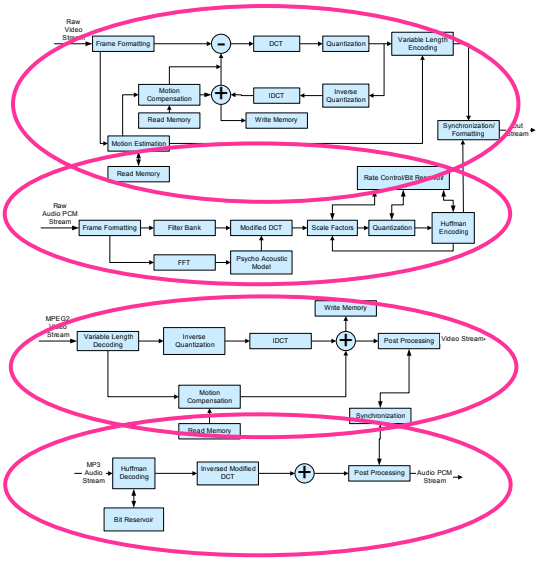




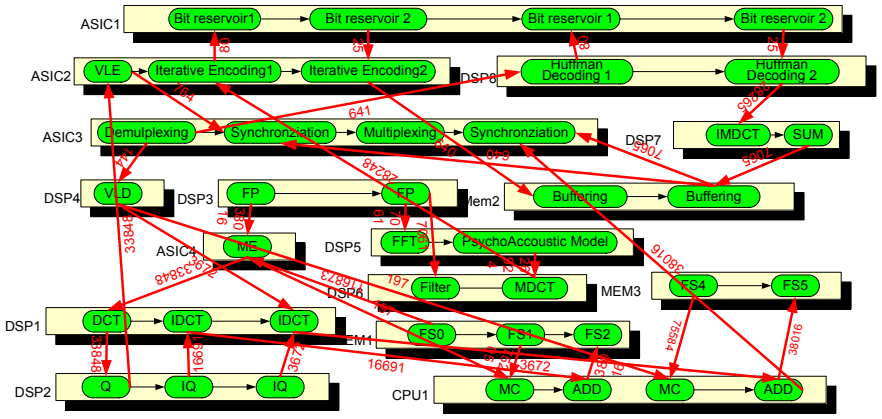
1. Helps in finding solutions for architectures with lower link bandwidth
   → Lower implementation cost
2. Leads to solutions with less energy consumption

More Experiments - A MultiMedia System

- **System composition:**
  - **H263 video encoder**
  - **H263 video decoder**
  - **MP3 audio encoder**
  - **MP3 audio decoder**



The Communication Task Graph



R. Marculescu

Results (MMS)

Power comparison between ad-hoc and EPAM-OE

Movie clips	Ad-hoc (mW)	EPAM-OE (mW)	Savings
Toybox/Hand	171.2	82.8	51.6%
Akiyo/Cup	226.2	104.8	53.7%
Akiyo/Cup	133.3	66.88	49.8%

Comparison between SA and EPAM-OE

	Simulated Annealing	EPAM-OE	Improvement
Run Time (sec)	25.55*	0.31	82.419
Power (mW)	119.36	105.12	11.9%

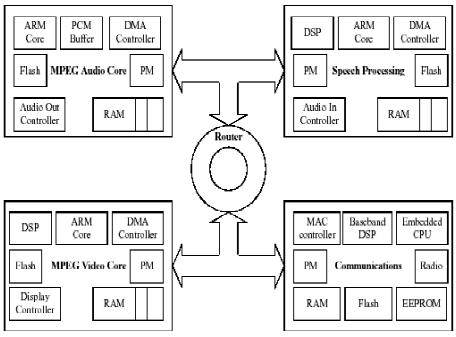
* As the problem size increases, the run time of SA also increases significantly.
For a 10x10 system, SA did not finish in more than 40 hours!.

R. Marculescu

Outline

- **Communication-based design**
  - **Node-Centric Perspective**
    - Irregular architectures
      - P2P communication
    - Regular architectures
      - Energy
      - ➔NOC power management
      - Fault-tolerance
  - **Network-Centric Perspective**
    - Channel modeling
    - Power-Performance trade-offs

NOC Power Management



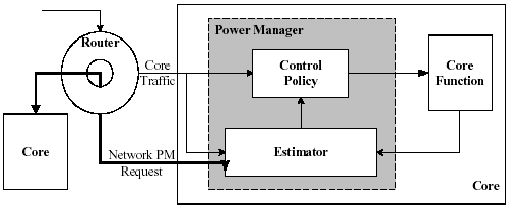
- **Node-centric**
  - **PM of the core determines state based on the state of the core and the workload**
- **Network-centric**
  - **PM supports the incoming power management requests from the network and performs transitions according to such requests**

CONJECTURE

System-level Power Management for future SOCs will gradually evolve from a **node-centric** toward a **network-centric** view

Local Power Manager

- Management of energy consumption under QoS formulated as a closed-loop stochastic control problem



- Power manager contains
  - Controller
    - Gives commands to the core that determines its performance and energy (frequency and voltage) in the active state – DVS
    - Chooses when to transition the core into one of the available low power states when the core is idle – DPM
  - Estimator
    - Observes the requests coming into the core's queue (Core Traffic) and the state of the core
    - Based on the observations, it estimates the parameters needed to recalculate the power management policy and thus closes control loop

Network-centric Power Management

- Advantages
  - Ability to make better predictions about the future workloads
  - Networking interface has to be defined in a way that protocols support passing power management messages between the cores
  - Network power management adds very few overhead packets to the overall communication stream between cores
  - Amount of energy wasted while the core is idle is reduced, as the local PM knows ahead of time that no requests are arriving in near future
  - The cost of waking up the core can be masked, as the other network elements may be able to notify the local core ahead of time when it's services are needed

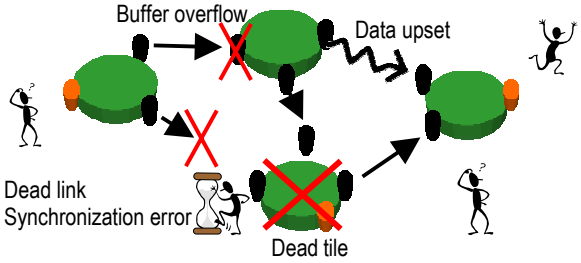
Outline

- **Communication-based design**
  - **Node-Centric Perspective**
    - Irregular architectures
      - P2P communication
    - Regular architectures
      - Energy
      - NOC power management
      - ➔ Fault-tolerance
  - **Network-Centric Perspective**
    - Channel modeling
    - Power-Performance trade-offs

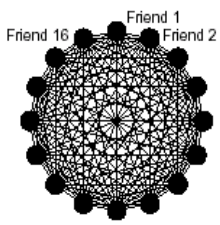
Why Fault-Tolerant Communication?

- **CMOS technology scaling is fastly approaching the physical limits**
  - **Circuits are subject to new types of malfunctions and failures**
  - **Current CAD tools *cannot* solve these new problems**
    - With technologies < 100nm, failures are very hard to predict and avoid
    - Relaxing the requirement of 100% correctness of devices and interconnects drastically reduces the design and verification costs

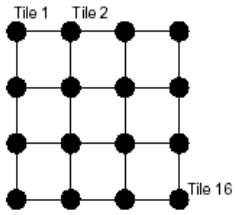
- **NoC protocols must be tolerant to common faults**
  - **Data upsets (p_{upset})**
    - **Crosstalk, EMI**
  - **Buffer overflows**
  - **Node/link failures**
  - **Synchronization errors**



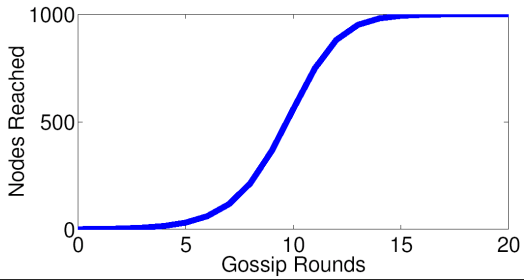
Probabilistic Broadcast



a) Fully connected network

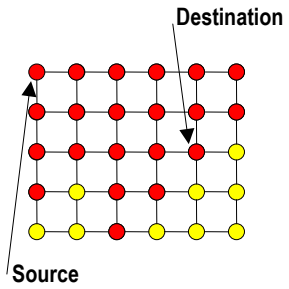


b) Square grid network

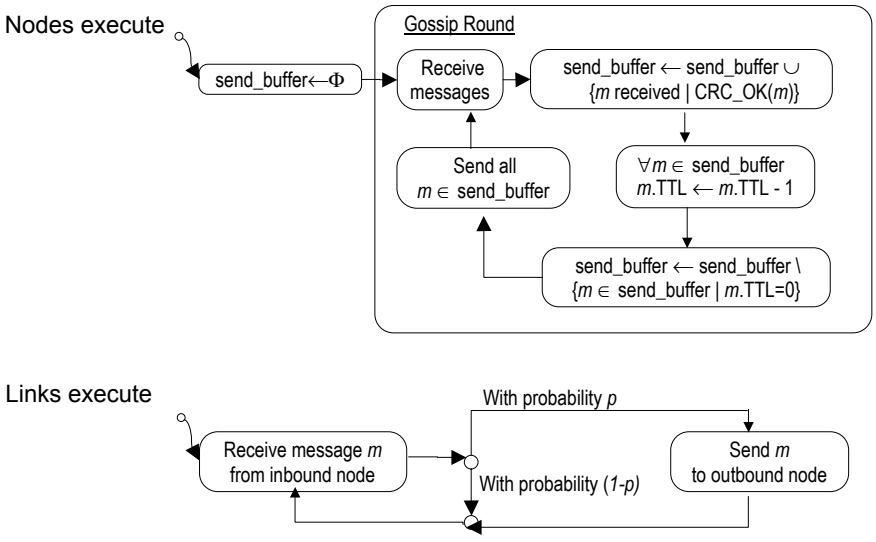


What We Propose?

- ***Stochastic Communication***
  - **Use an error detection / multiple transmissions scheme based on a lightweight probabilistic algorithm**
  - **Packets are transmitted several times, on several different paths**
    - During one round, packets are sent to a randomly chosen subset of the neighbors
  - **Messages are disseminated exponentially fast among the nodes of the network**
    - If a packet is corrupted, it is discarded, as it will be received again with high probability
  - **Transmissions are protected by a CRC**
    - Easy to implement in hardware
    - Extremely resilient to all types of failures



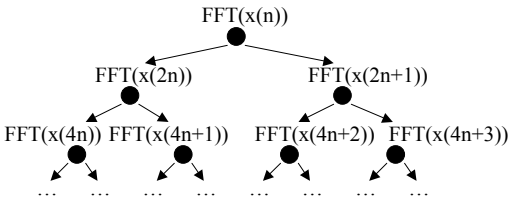
A Closer Look…

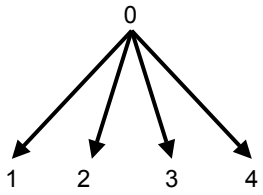


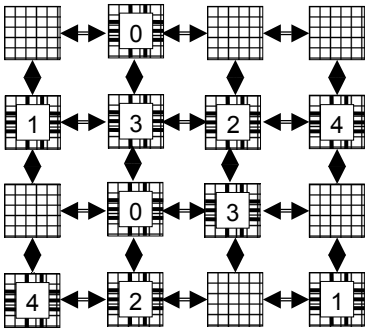
Performance Evaluation

- **Parameters of Stochastic Communication**
  - **Transmission probability *p***
    - The packets are then sent over the link with a certain probability *p*
  - **Time-to-live *TTL***
    - Upon creation, a packet is assigned a *TTL*
    - With each hop the *TTL* is decreased. When *TTL* = 0 the packet is destroyed
  - **Duration of a round T_R**
    - Tiles must be able to transmit their packets to the neighbors
- **Performance metrics**
  - **Latency**
    - Number of gossip rounds needed to reach destination
  - **Energy dissipation**
    - Depends on the total number of packets transmitted in the NoC
  - **Fault-tolerance**

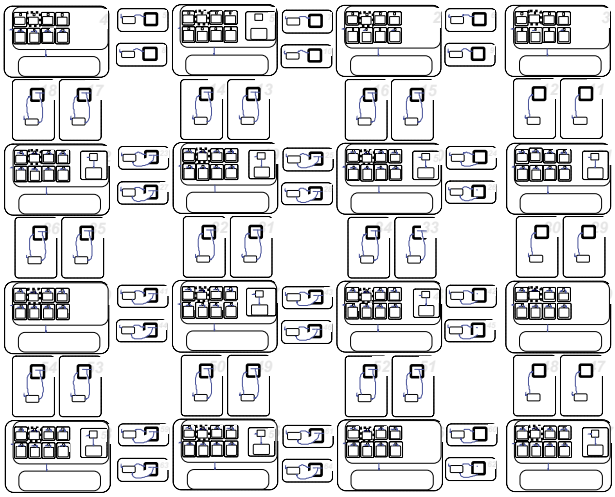
Case Study: FFT2





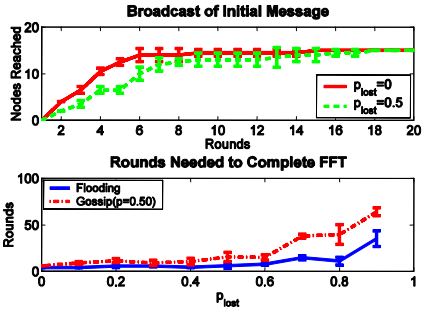


Stateflow Image



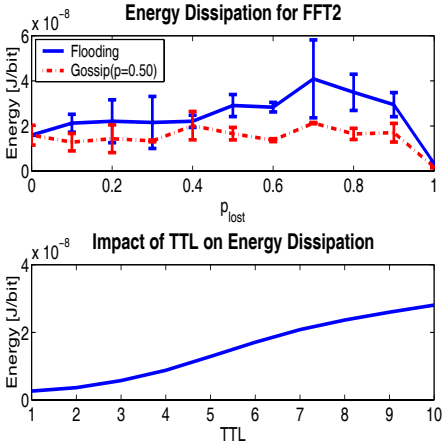
Results: Latency

- Parameters of stochastic communication
  - Failure probabilities
    - p_{tiles}
    - p_{links}
    - $p_{lost} = p_{upset} + p_{buff}$
  - Transmission probability
    - p = {1, 0.75, 0.5, 0.25}
  - Time-to-live *TTL*
- Broadcast of initial message has an explosive spread
  - Latency close to optimal for reasonabl levels of data upsets
  - Jitter is small



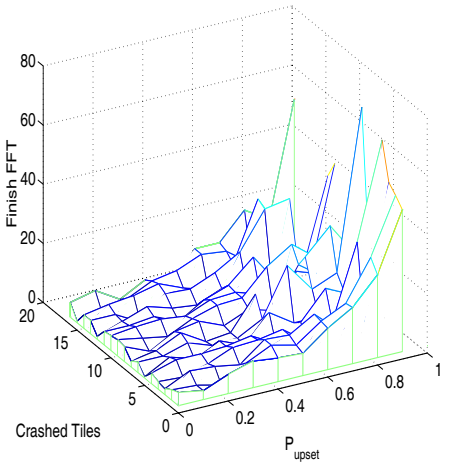
Results: Energy Dissipation

- Energy dissipation
  - Proportional to the transmission probability p
  - Increases almost linearly with the *TTL*
- Clear tradeoff between energy and performance
  - Can be tuned with the parameters p and *TTL*



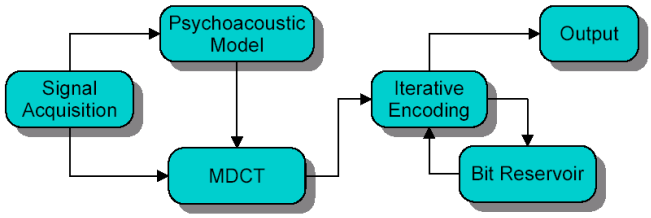
Results: Fault-Tolerance

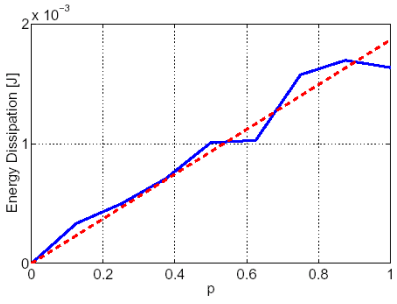
- **Tile failures vs. data upsets**
  - **Tile failures**
    - Little impact on latency
    - Will cause communication to fail
  - **Data upsets**
    - Big impact on latency
    - Will not cause communication to fail (except when ≈ 100%)

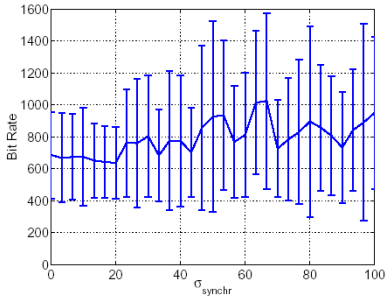


R. Marculescu

More Results: MP3





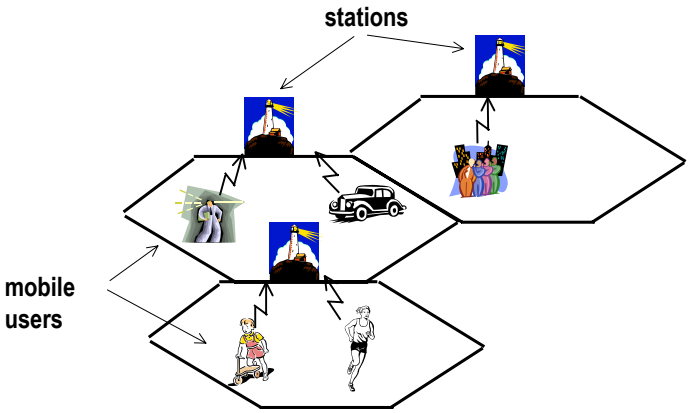


R. Marculescu

Outline

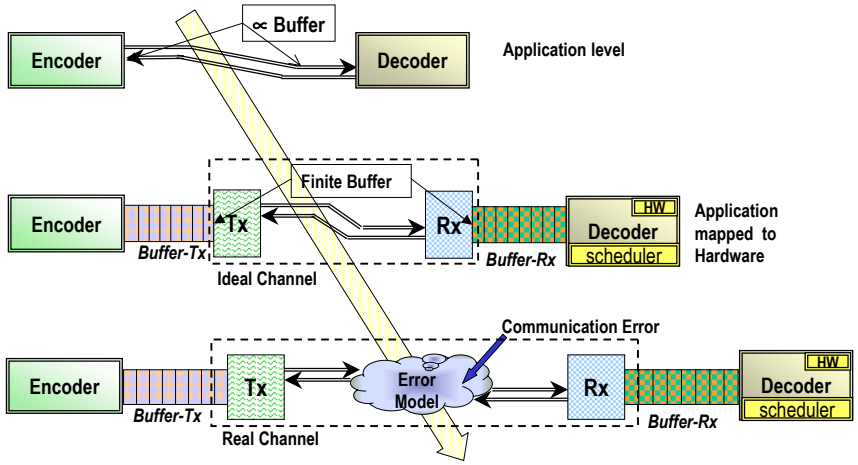
- **Communication-based design**
  - **Node-Centric Perspective**
    - Irregular architectures
      - P2P communication
    - Regular architectures
      - Energy
      - NOC power management
      - Fault-tolerance
  - ➔ **Network-Centric Perspective**
    - Channel modeling
    - Power-Performance trade-offs

Designer's Playground: The Pico-Cell Environment

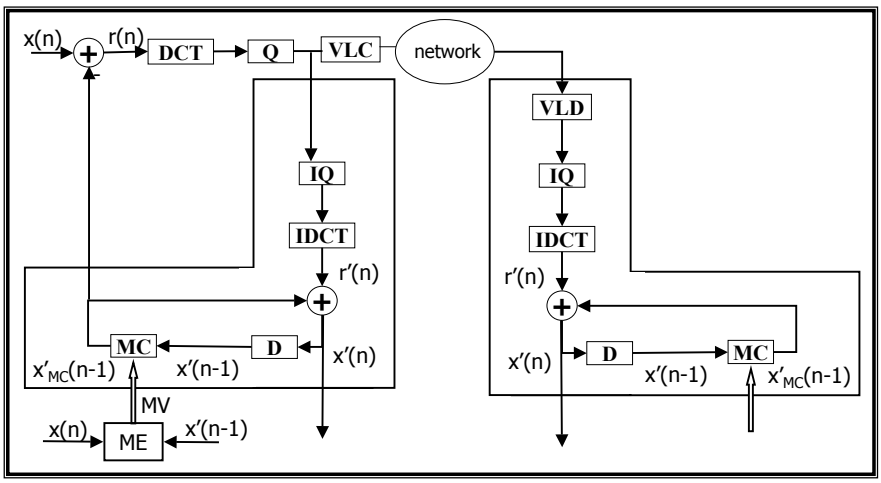


Both *computation* and *communication* aspects are important

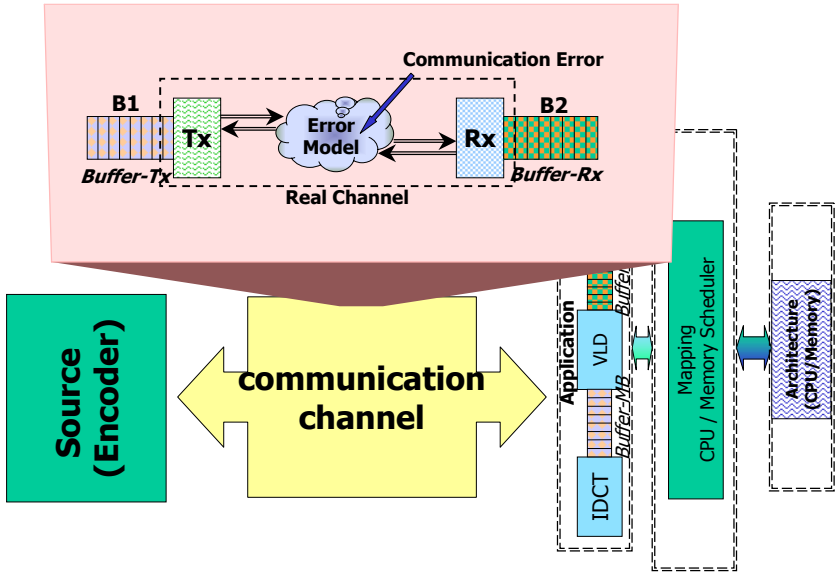
The Wireless Communication Channel



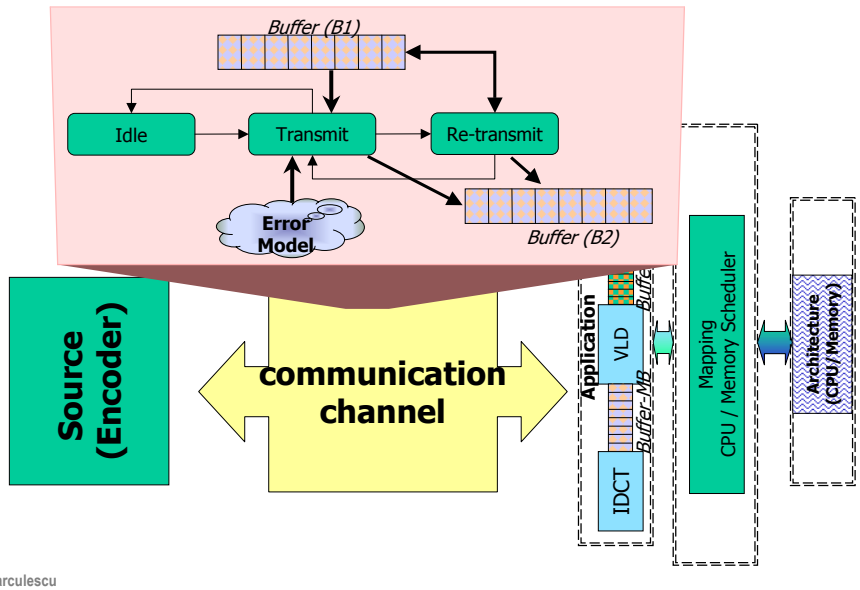
The MPEG-2 Decoder Application



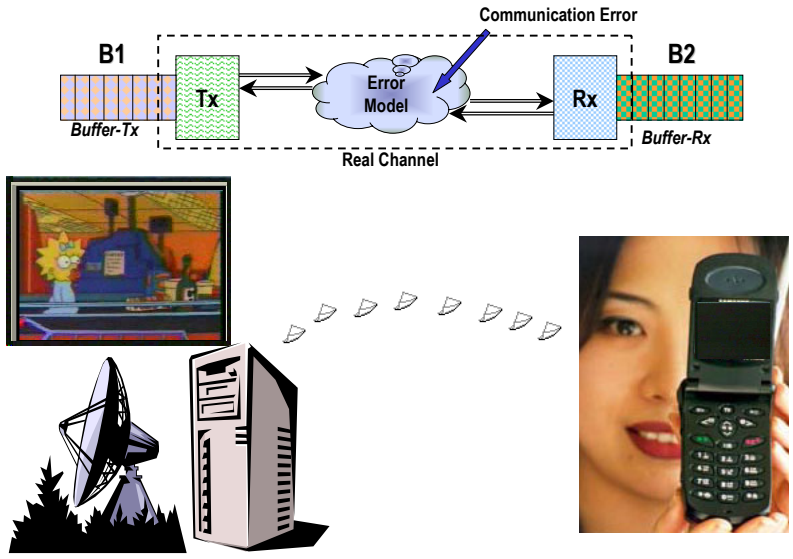
How About the Channel?



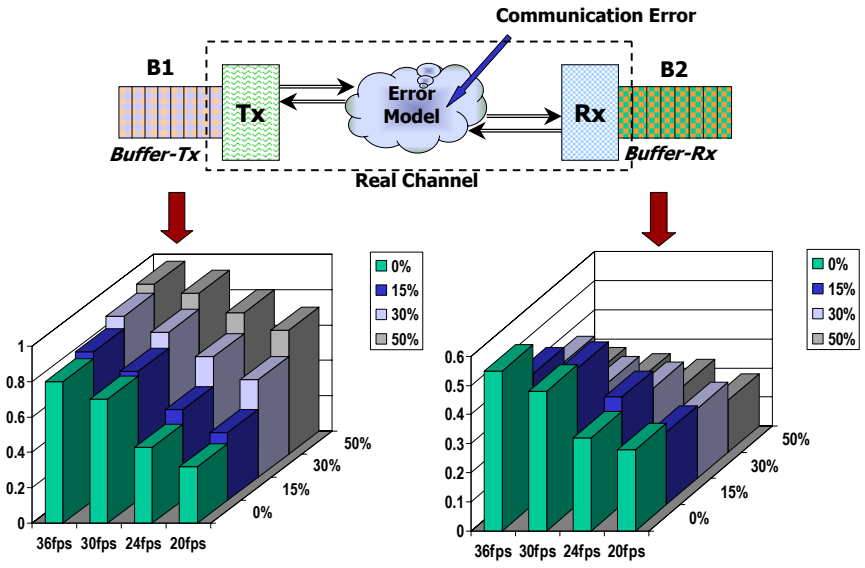
Putting Everything Together



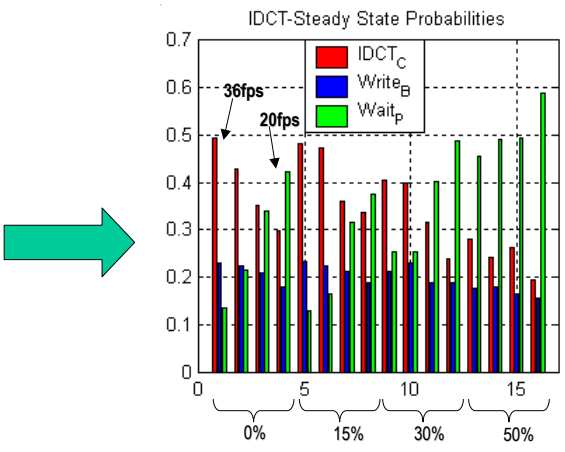
What Are We Trying to Analyze?



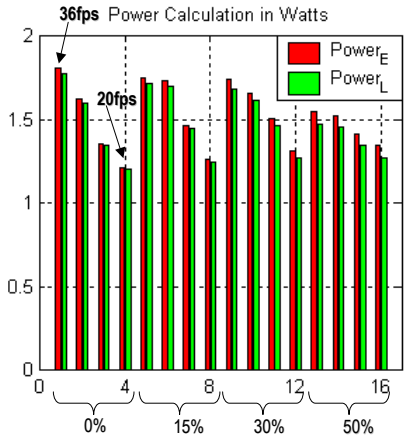
Again, the Network-Centric Perspective…

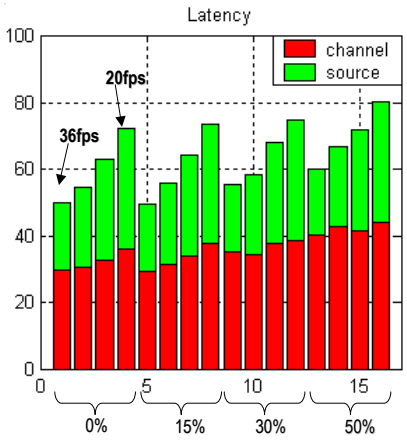


… and Some Node-Centric Results



How About Power? Latency?





Summary

- **Communication-based design is a fundamental paradigm (both on- and off-chip)**
  - **P2P communication is becoming increasingly important**
  - **Irregular architectures**
    - Need to synthesize the communication infrastructure
    - Communication-driven floorplanning plays a major part
    - Need to go deeper into physical-level effects and integrate them at system-level for physical planning
  - **Regular (tile-based) architectures**
    - There are pros and cons for the NOC approach
    - Energy, power management and fault-tolerance are very important
    - Buffer prediction needs to take care of limited resources
    - Mapping and routing are crucial for exploiting this architecture

References (for Part III and Part IV)

- **J. Hu, R. Marculescu, 'Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints,' in Proc. ASPDAC, Kitakyushu, Japan, Jan. 2003.**
- **J. Hu, R. Marculescu, 'Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures,' to appear in Proc. DATE, Munich, Germany, March 2003.**
- **T. Dumitras, S. Kerner, R. Marculescu, 'Towards On-Chip Fault-Tolerant Communication,' in Proc. ASPDAC, Kitakyushu, Japan, Jan. 2003.**
- **T. Dumitras, R. Marculescu, 'On-Chip Stochastic Communication,' to appear in Proc. DATE, Munich, Germany, March 2003.**
- **G. Varatkar, R. Marculescu, 'Traffic Analysis for On-chip Networks Design of Multimedia Applications,' in Proc. DAC, New Orleans, LA, June 2002.**
- **A. Pinto, L. Carloni, A. Sangiovanni-Vincentelli, Constraint-Driven Communication Synthesis,' in Proc. DAC, New Orleans, LA, June 2002.**
- **T. Simunic, 'Managing Power Consumption in Networks On Chips,' in Proc. DATE, Paris, France, March 2002.**
- **J. Hu, Y. Deng, R. Marculescu, 'System-Level Point-to-Point Communication Synthesis Using Floorplanning Information,' in Proc. ASPDAC-VLSI, Bangalore, Jan. 2002.**

References

- Paul Marchal, P.Yang, M.Jayapala, S. de Souza, F.Catthoor, 'Matador: A Multiprocessor Platform Simulation Environment,' in Journal of Circuits, Systems, and Computers, Oct. 2002.
- F.N. van Wijk 1, J.P.M. Voeten, A.J.W.M. ten Berg, 'An Abstract Modeling Approach Towards System-Level Design-Space Exploration,' in Proc. FDL02, Marseille, France, Oct. 2002.
- R. Marculescu, A. Nandi, L. Lavagno, A. Sangiovanni-Vincentelli, 'System-Level Power/Performance Analysis of Portable Multimedia Systems Communicating Over Wireless Channels,' in Proc. ICCAD, San Jose, CA, Nov. 2001.
- A. Nandi, R. Marculescu, 'System-level Power/Performance Analysis for Embedded Systems Design,' in Proc. DAC, Las Vegas, NV, June 2001.
- P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, R. Lauwereins, 'Energy-Aware Runtime Scheduling for Embedded-Multiprocessor SOCs,' Design & Test of Computers, Sept. 2001.
- P. Lieverse, P. van der Wolf, K. Vissers, and E. Deprettere. "A Methodology for Architecture Exploration of Heterogeneous Signal Processing Systems". *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, Vol. 29, No. 3, 2001.
- M. Sgroi, L. Lavagno, A. Sangiovanni-Vincentelli, 'Formal Models for Embedded Systems Design,' in *IEEE Design and Test of Computers,* Vol. 17, April-June 2000.
- T. Simunic, L. Benini, P. Glynn, G. De Micheli, 'Dynamic Power Management for Portable Systems,' in *Proc. Mobicom*, Boston, MA, Aug. 2000.

References

- K. Keutzer, S. Malik, R. Newton, J. Rabaey and A. Sangiovanni-Vincentelli, 'System Level Design: Orthogonalization of Concerns and Platform-Based Design,' in IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 19, No. 12, Dec. 2000.
- J-Y. Brunel, E.A. de Kock, W.M. Kruijtzer, H.J.H.N. Kenter, W.J.M. Smits, 'Communication Refinement in Video Systems On Chip,' in Proc. CODES, Rome, 1999.
- P. van der Wolf, P. Lieverse, M. Goel, D. La Hei, K. Vissers, 'An MPEG-2 Decoder Case Study as a Driver for a System Level Design Methodology,' in Proc. CODES, Rome, 1999.
- J.-Y. Brunel, A. Sangiovanni-Vincentelli, R. Kress, and W. Kruytzer, 'COSY: A Methodology for System Design Based on Reusable Hardware and Software IP's,' in J.-Y. Roger, Ed., T*echnologies for the Information Society*, IOS Press, 1998.
- A. Kalavade, P. Moghe, 'A Tool for Performance Estimation of Networked Embedded End-Systems,' in Proc. DAC, San Francisco, CA, June 1998.
- B. Kienhuis, E. Deprettere, K. Vissers, and P. van der Wolf, 'An Approach for Quantitative Analysis of Application-Specific Dataflow Architectures,' in *Proc. ASAP '97*, Los Alamitos, CA, 1997.
- B. Plateau, K. Atif, 'Stochastic Automata Network for Modelling Parallel Systems,' in IEEE Trans. on Software Engineering, Vol. 17, Oct. 1991.
- R. Milner, '*Communication and Concurrency,'* Prentience-Hall, Englewood Cliffs, New Jersey, 1989.