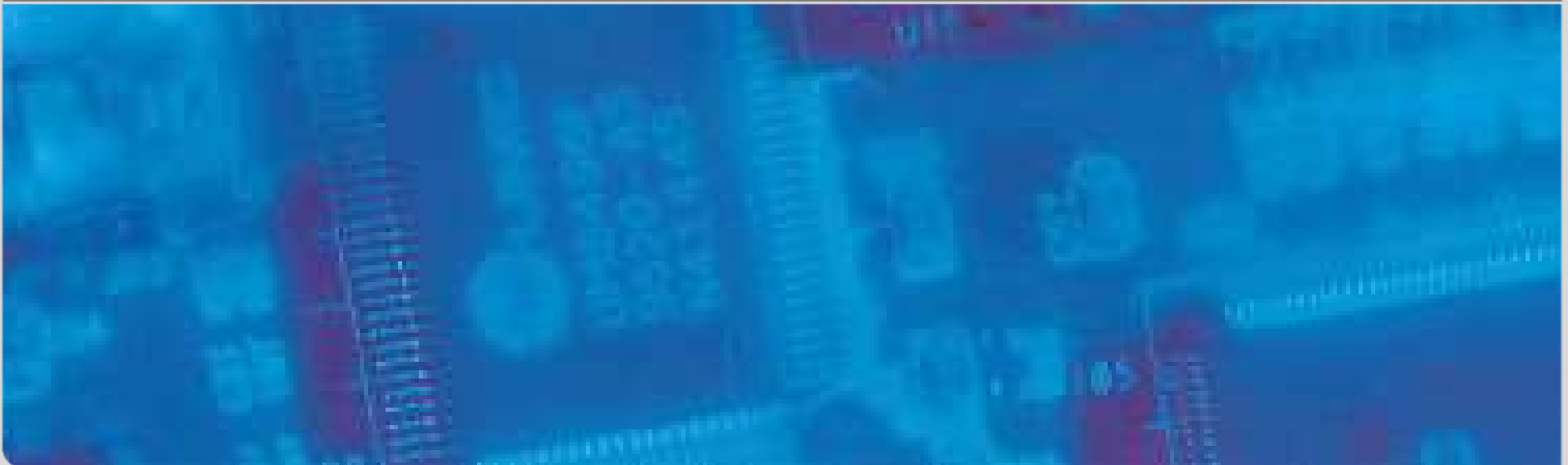


Lower Power Design

Lecture 4: Hardware Power Optimization and Estimation 1

Anuj Pathania on behalf of Prof. Dr. Jörg Henkel
Summer Semester 2017

CES – Chair for Embedded Systems



Organizational Issues

- Slides available for download -
 - http://cesweb.itec.kit.edu/teaching/LPD/s17_slides/
 - Username: student
 - Password: CES-Student
- Homework
 - Read a relevant scientific paper.
 - Discussion next class.
- Oral Exam
 - Make appointment with KIT CES secretary 6-8 weeks in advance.
 - Exam will be in English (or German if told in advance).
 - More information: <http://ces.itec.kit.edu/972.php>

Lectures

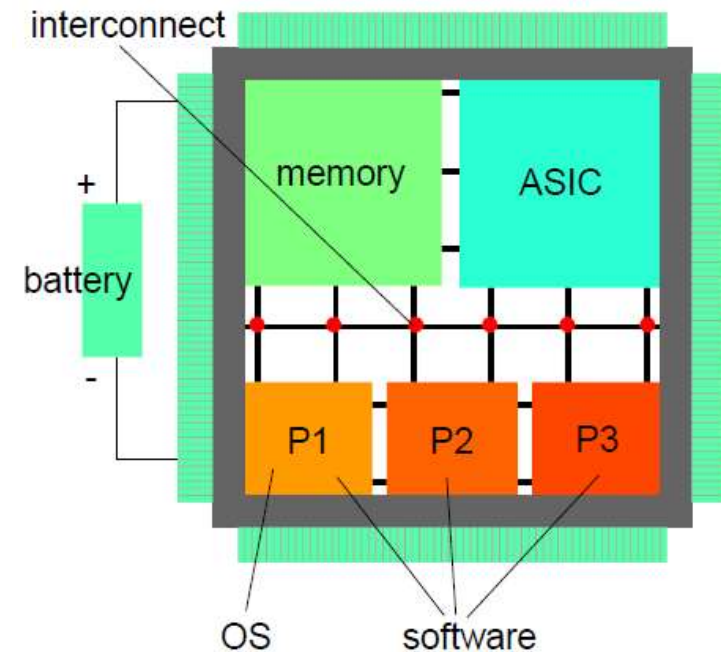
- 27.04.2017 – 0: Introduction
- 04.05.2017 – 1: Energy Sources
- 11.05.2017 – 2: Battery Modelling Part 1
- 18.05.2017 – 3: Battery Modelling Part 2
- 25.05.2017 – ~~Ascension Day (Holiday)~~
- 01.06.2017 – 4: Hardware Power Optimization and Estimation 1
- 08.06.2017 – 5: Hardware Power Optimization and Estimation 2
- 15.06.2017 – ~~Corpus Christi (Holiday)~~
- 22.06.2017 – 6: Hardware Power Optimization and Estimation 3
- 29.06.2017 – TBA
- 06.07.2017 – TBA
- 13.07.2017 – TBA
- 20.07.2017 – TBA
- 27.07.2017 – TBA

Overview for Today

- Power consumption in hardware.
- Operator scheduling.

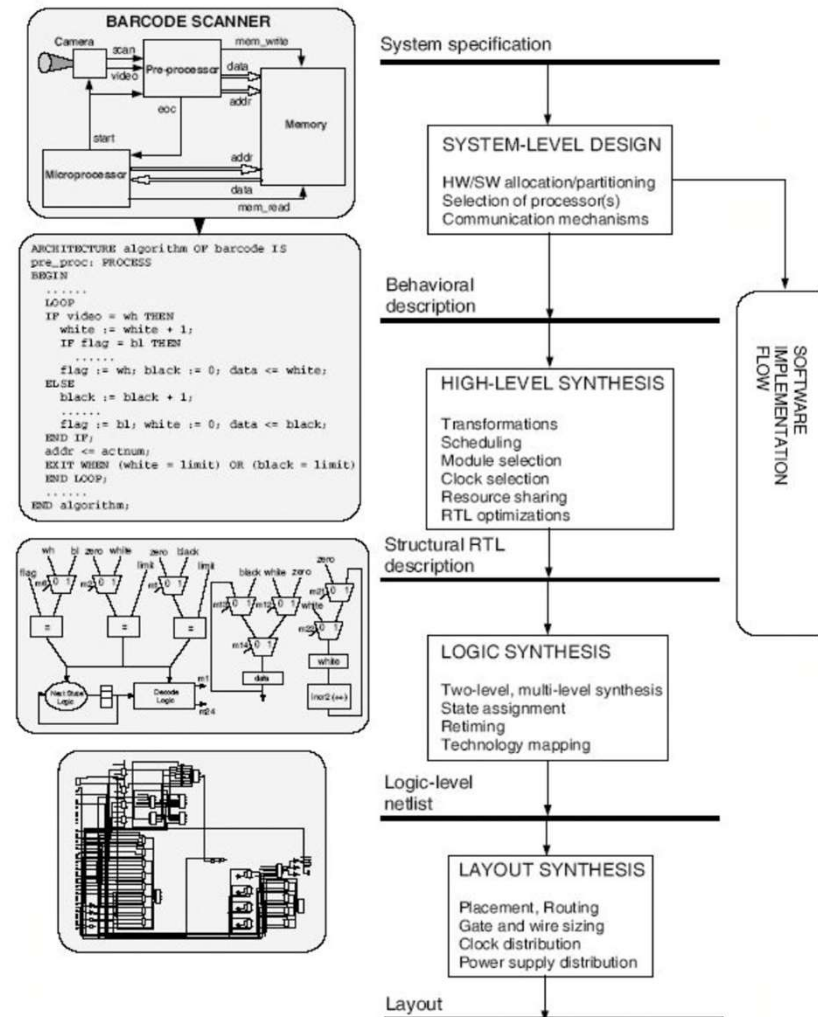
Overview

- Different levels of abstraction
 - System
 - RTL
 - Gate
 - Transistor
- Challenges
 - Optimize (Minimize power consumption)
 - Design/Co-Design (Synthesize, Compile, ...)
 - Estimate and Simulate



Generic Hardware Synthesis Flow

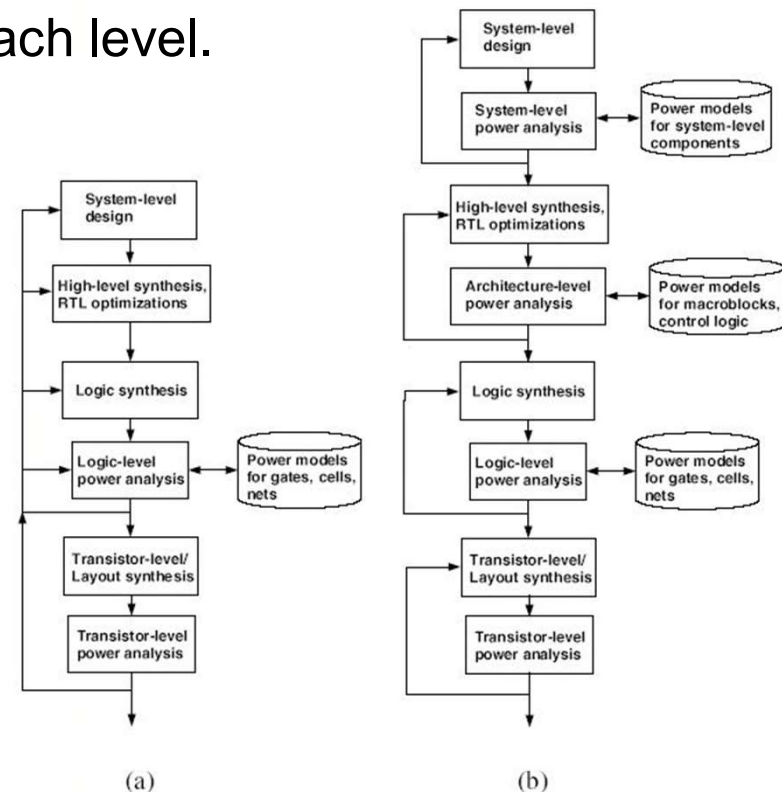
- System-Level Design
- High-Level Synthesis
- Logic Synthesis
- Layout Synthesis



Source: Anand [2012]

Low Power Hardware Design Flow

- Energy/power needs to be analyzed/optimized at each level of abstraction.
- Necessitates appropriate model for each level.



Design Flow without Power (a) and with Power (b)

Source: Anand [2012]

Power Consumption in Hardware

- In general, four components:
 - $P_{\text{sw.cap}}$ switching capacity power
 - $P_{\text{short-circuit}}$ short-circuit power
 - P_{leakage} leakage power
 - P_{static} static power

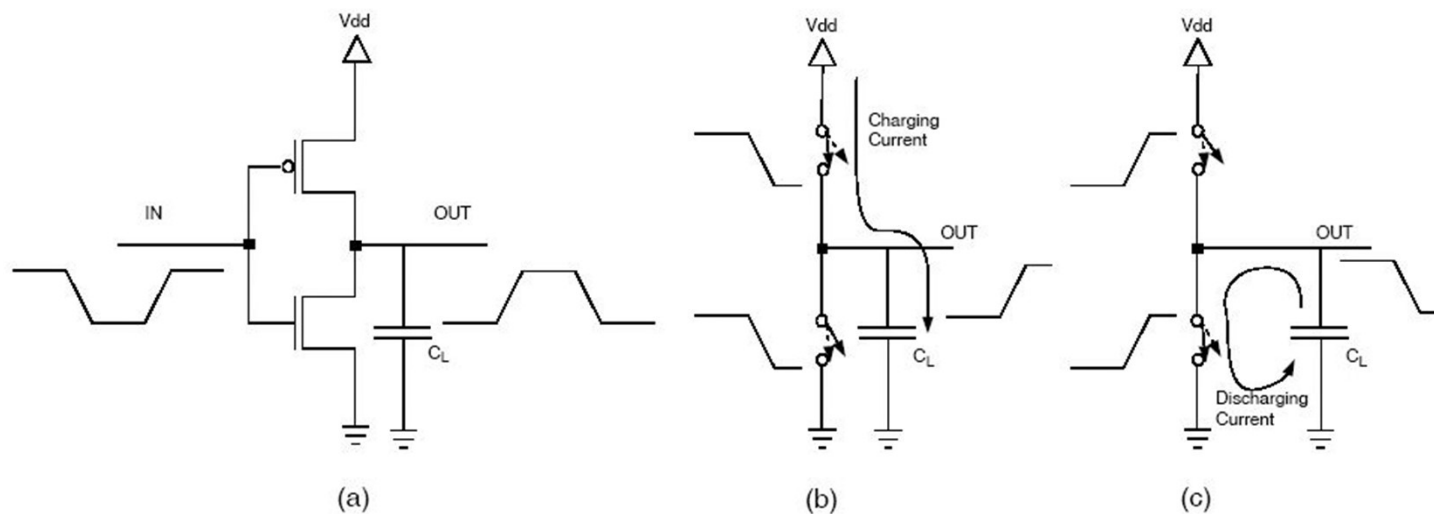
$$P_{\text{avg}} = P_{\text{sw.cap.}} + P_{\text{short-circuit}} + P_{\text{leakage}} + P_{\text{static}}$$

Source: Anand [2012]

Switching Capacity Power

- Caused by charging/discharging of parasitic capacitances:
 - C_L : cumulative parasitic capacitance
 - N : expected number of transitions per clock cycle
 - f : clock frequency

$$P_{sw.cap} = \frac{1}{2} C_L V_{dd}^2 N f$$



CMOS Inverter

Source: Anand [2012]

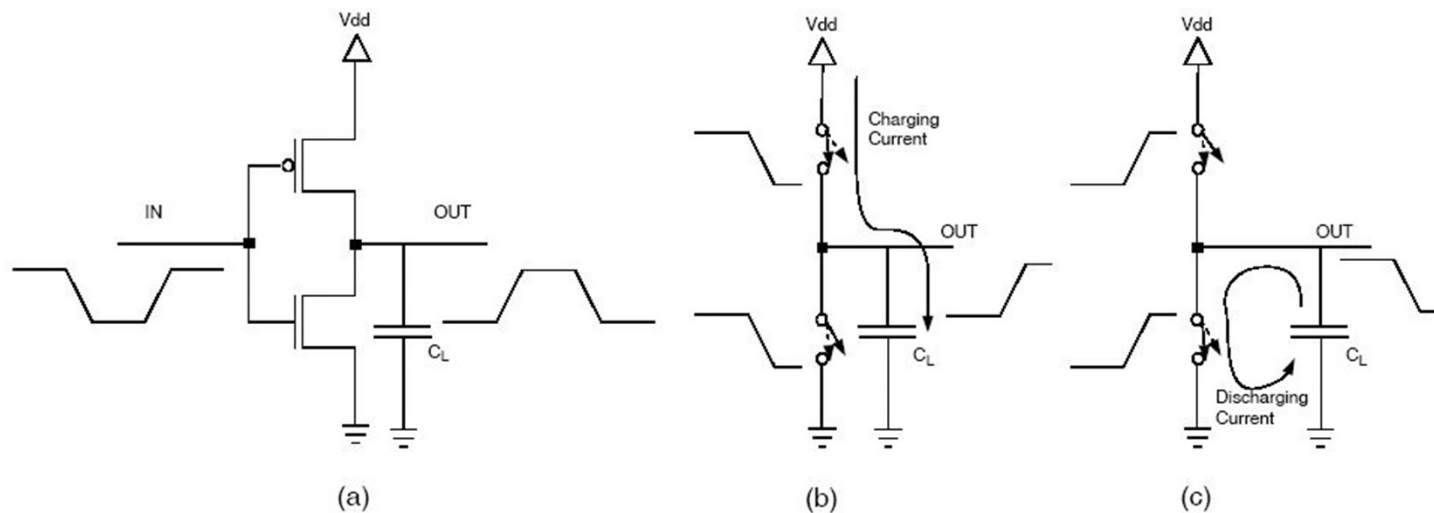
Parasitic Capacitance

- Also known as *Stray Capacitance*.
- Exists between all electronic components due to proximity.
- Exists between conductors.
- Problematic in high frequency circuits.

Short-Circuit Power

- Caused by direct supply-to-ground path:
 - K: constant (Transistor Size, Technology)
 - V_T : expected number of transitions per clock cycle
 - τ : input rise/fall time

$$P_{short-circuit} = K (V_{dd} - 2V_T)^3 \tau N f$$

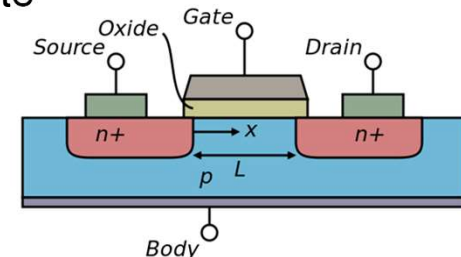


CMOS Inverter

Source: Anand [2012]

Leakage Power

- Loss of power from diodes due to current conduction by off transistors.
 - I_{diode} : diodes formed between diffusion region and substrate
 - I_{oxide} : electrons tunneling through the gate oxide
 - Drops exponentially with gate length.
 - K,S: technology parameters
 - W_{eff} : effective transistor channel width
 - Note: Leakage power is becoming more dominant with silicon node shrinking.



$$P_{leakage} = (I_{subthreshold} + I_{oxide} + I_{diode}) \cdot V_{dd}$$

$$I_{subthreshold} = K W_{eff} e^{\frac{V_{in} - V_{T_s}}{S}}$$

Source: Anand [2012]

Static Power

- Not relevant in CMOS circuits.
- Sometime included along the leakage power.
- Relevant in
 - other logic families
 - some nMOS circuits where there is a constant path supply-to-ground (e.g. ECL)

Dennard Scaling vs. Power Density

- Transistor and power scaling are no longer balanced.
 - Scaling is limited by power.
- Higher power density leads to thermal problems.
 - Accelerates aging effects.

Classical scaling (Dennard)

Device count	S^2
Device frequency	S
Device power (cap)	$1/S$
Device power (V_{dd})	$1/S^2$
Power Density	1

S: Scaling Factor

Power Limited Scaling

Device count	S^2
Device frequency	S
Device power (cap)	$1/S$
Device power (V_{dd})	~ 1
Power Density	S^2

Source: Venkatesh [2010]

Operator Scheduling for Low Power

- Scheduling (HLS) e.g.:
 - Operator scheduling
 - Module selection
 - Glitch power reduction
 - State transition reduction
 - ...

Source: Anand [2012]

Operator Scheduling for Low Power

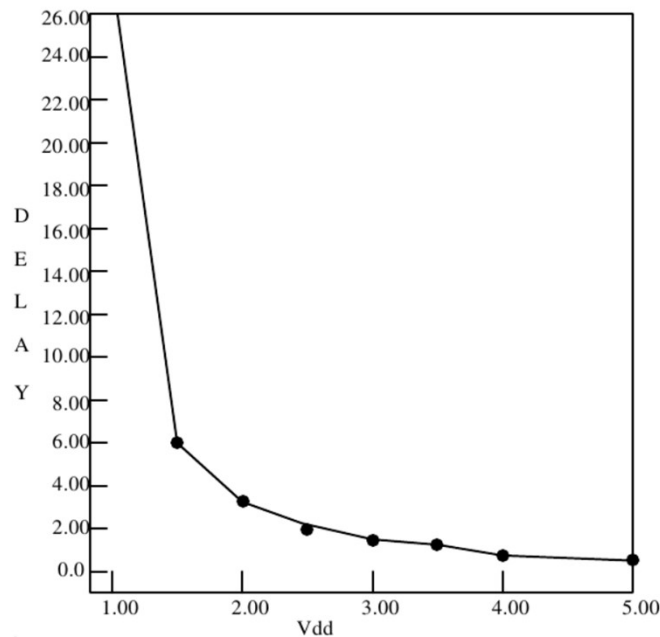
- Scheduling (HLS) e.g.:
 - Assigns operations in the behavioral description to control steps or controller states.
 - Determines cycle-by-cycle behavior (i.e. sequence in which operations are performed).
 - Determines the sequence in which the various operations of the behavioral description are performed.
 - Dictates which operations and variables can share the same functional units and registers.
 - Can be used to enable resource sharing for low power by ensuring that correlated variables and operations with correlated operands are appropriately sequenced so that they can share the same resources.
- Some repetition from ESI:
 - *Multi-cycling*: Operation can take more than one cycle.
 - *Chaining*: Multiple operations executed in one cycle.

Operator Scheduling for Low Power 2

- Scheduling can be performed so as to enable maximum resource sharing between operations that belong to instances of the same computational pattern, resulting in maximal exploitation of regularity during resource sharing.
- Scheduling can be used to distribute the slacks or mobilities of various operations in the DFG appropriately so that some operations may be performed using slower, more energy-efficient functional units. Thus, scheduling has an impact on the power trade-offs through module selection
- Scheduling determines the distribution of operations over time, and hence affects the profile of the power consumption in the implementation over time (control steps or clock cycles). Reducing peak power is important due to packaging, cooling, and reliability considerations. The effect of scheduling on peak power will be illustrated later.

Delay vs V_{dd}

- How to assign voltages to functional units?



Normalized Delay vs V_{dd}

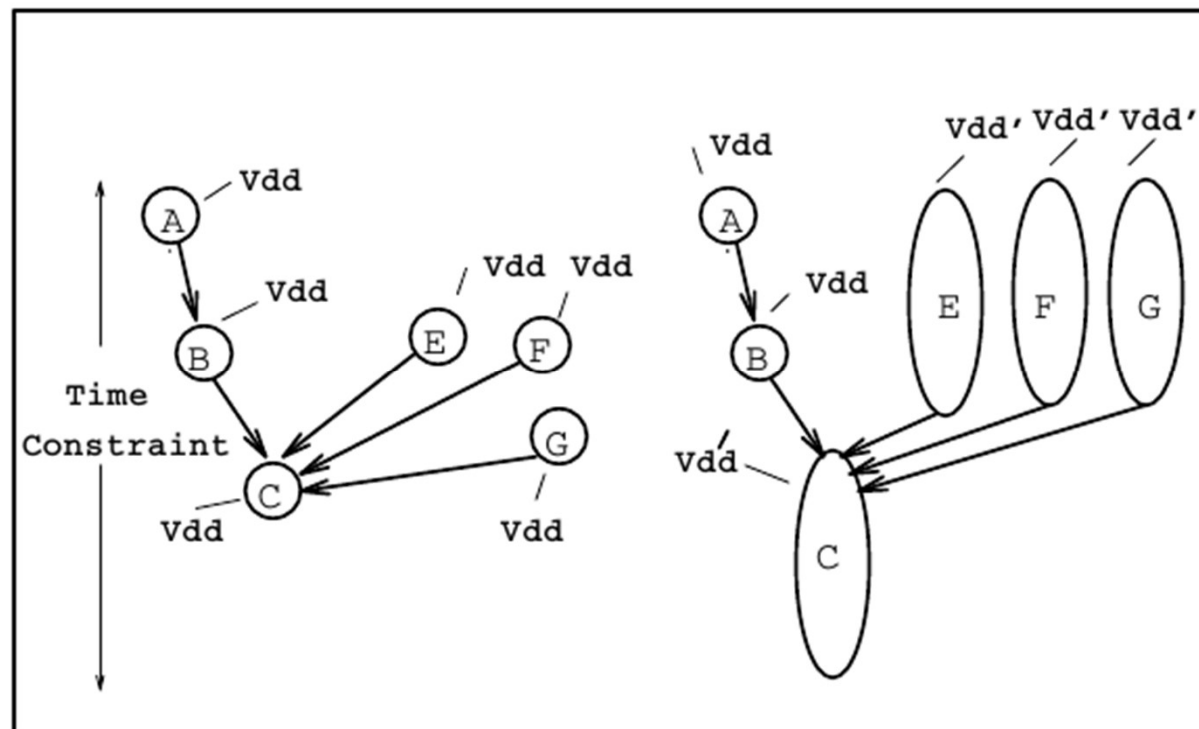
$$delay = \frac{k \cdot V_{dd}}{(V_{dd} - V_T)^2}$$

$$P_{dyn} \sim V_{dd}^2 C_{load} f_{switch}$$

Source: Sarraf [1995]

Data Flow Graphs (DFGs)

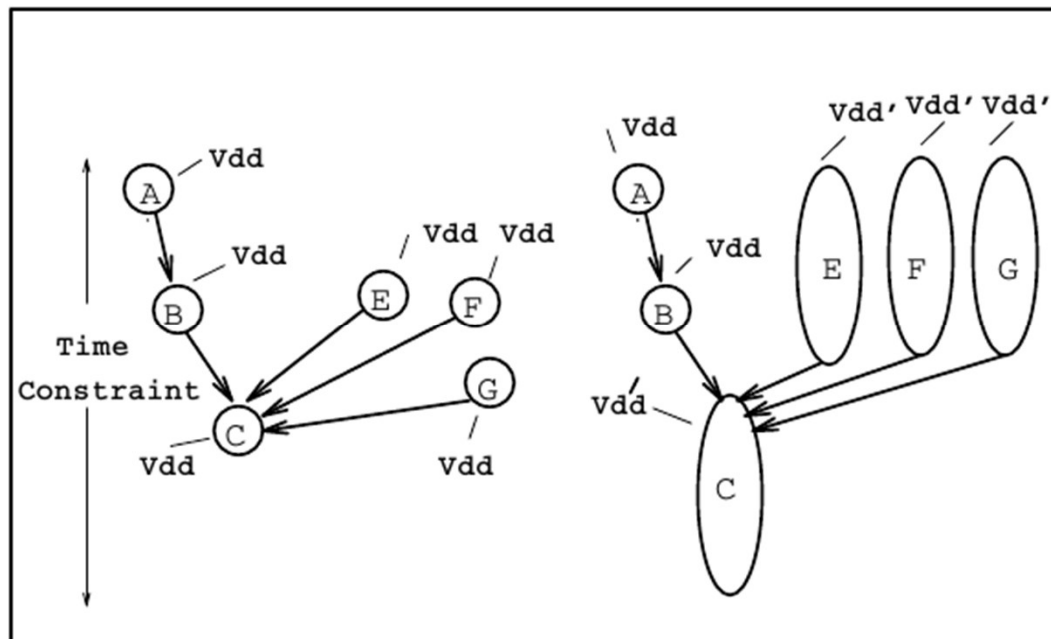
- DFG: Directed Acyclic Graph $G=(V,E)$
- Directed edge from v_i to v_j : v_i must precede v_j



Source: Sarraf [1995]

Data Flow Graphs (DFGs)

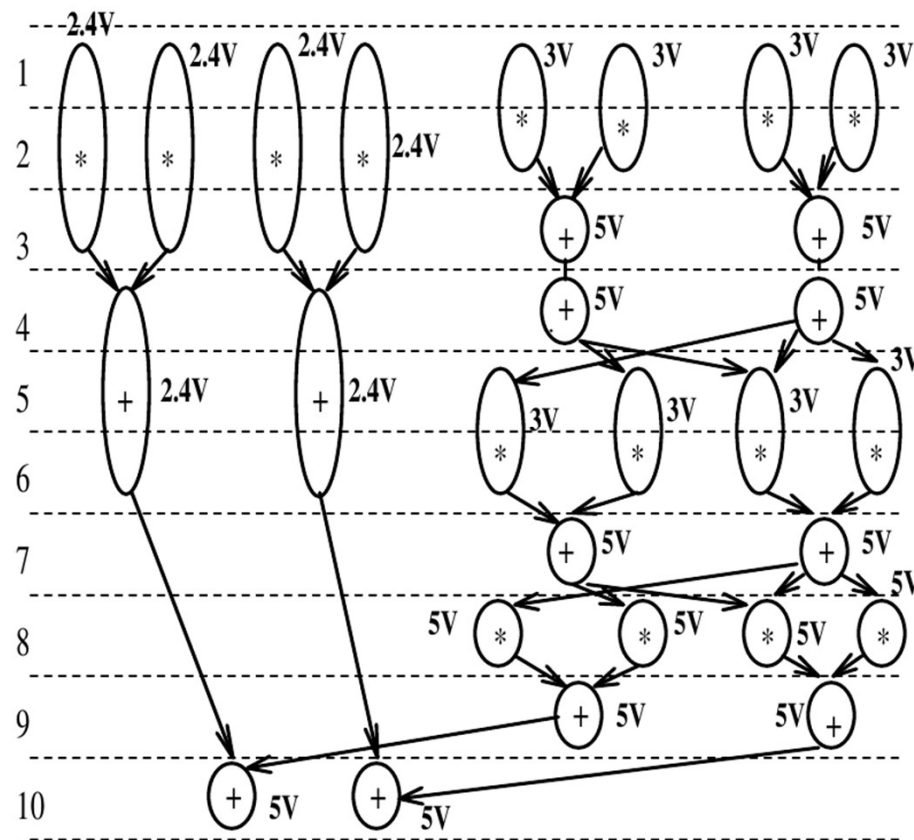
- A critical path of a system is defined as the path in the DFG, $\{v_1, v_2, \dots, v_k\}$, such that the summation of the latencies of the nodes in the path is maximal among all the paths of the DFG. The sum C_p is termed as the critical path length.



Source: Sarraf [1995]

Motivation

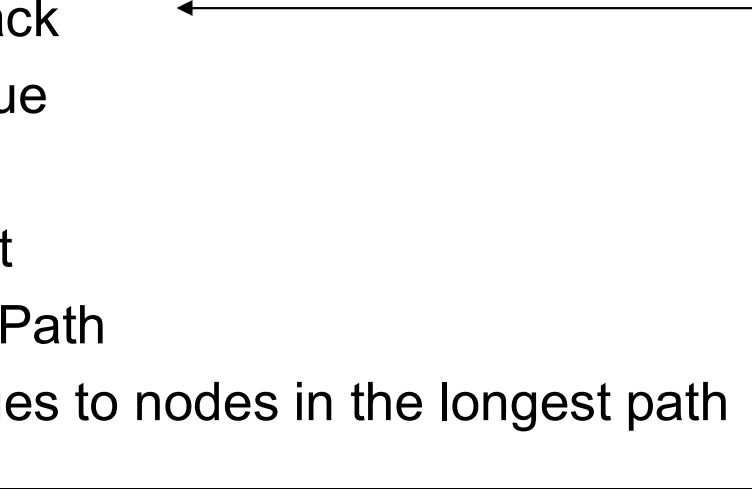
- How do we know the optimal voltages?



Lowest Power DFG for AR-Lattice Filter

Source: Sarraf [1995]

Algorithm Overview (Exact Algorithm)

- Step 1: Initialization
 - Step 2: Computation of Slack
 - Step 3: Maximal Slack Value
 - Step 4: Dual graph H_p
 - Step 5: Weight Assignment
 - Step 6: Longest Weighted Path
 - Step 7: Reassigning voltages to nodes in the longest path
 - Step 8: Goto Step 2
- 

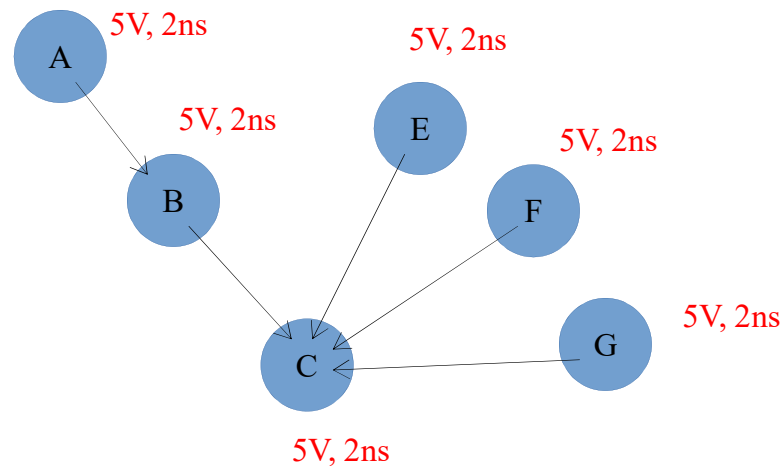
Source: Sarraf [1995]

Step 1: Initialization

Step 1: *Initialization*

Each of the nodes, $v \in V$, in $G(V, E)$ is originally assigned a voltage V_c (also denoted by V_{c_0}), $\tau(v) = V_c$. $d(v)$ value is therefore initialized.

$$S = \{5\text{ V}, 3\text{ V}, 2.4\text{ V}\}$$

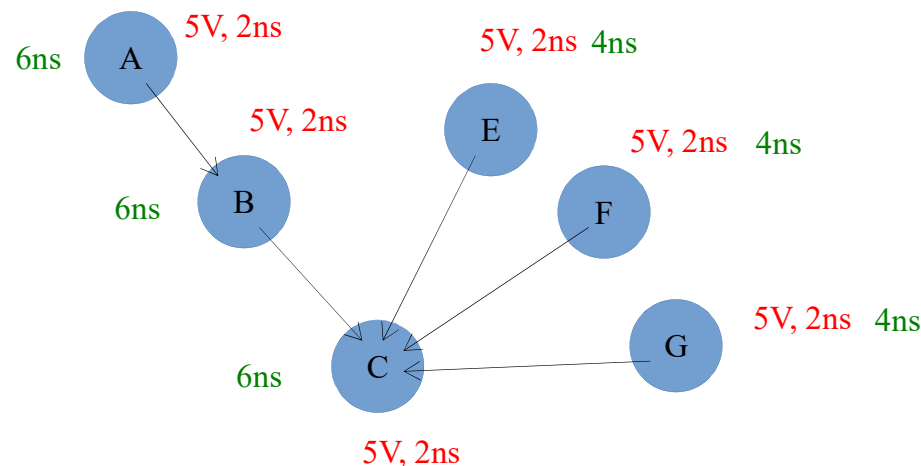


Source: Sarraf [1995]

Step 2: Computation of Slack

Step 2: *Computation of slack*

Using *depth first search* calculate $l(v)$ values for each of the nodes $v \in V$ and hence obtain $s(v) = kt_c - l(v)$.



$$\pi^-(v) = \{v' | (v'v) \in E\}$$

$$\pi^+(v) = \{v' | (v, v') \in E\}$$

$$t_i(v) = d(v) + \max_{z \in \pi^-(v)} t_i(z)$$

$$t_o(v) = d(v) + \max_{z \in \pi^+(v)} t_o(z)$$

$$l(v) = t_i(v) + t_o(v) - d(v)$$

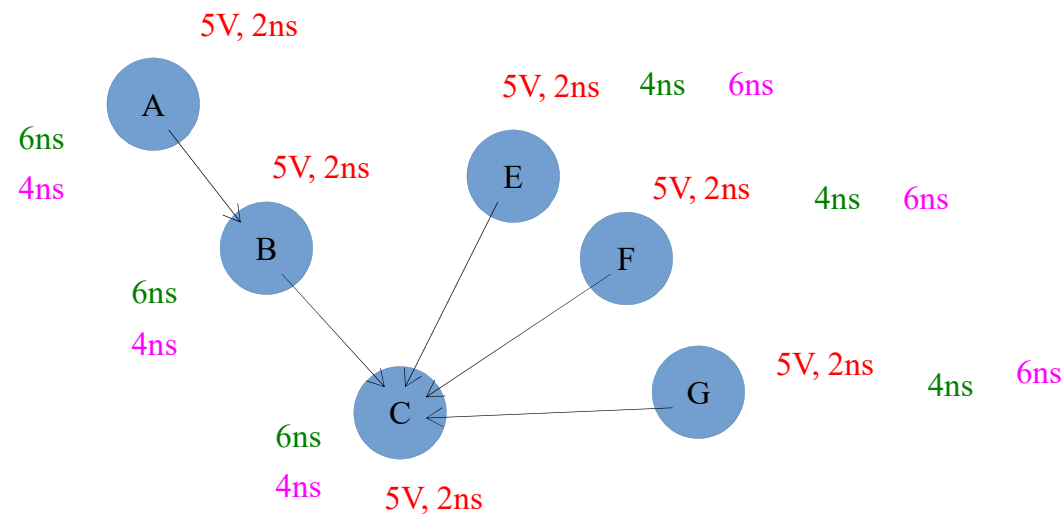
Source: Sarraf [1995]

Step 3: Maximal Slack Value

Step 3: *Maximal slack value*

Identify the maximum slack value s^* in the graph G and all the nodes, v , such that $s(v) = s^*$. If the maximal slack value $s^* = 0$ terminate the algorithm; we have obtained an optimal voltage assignment. The set of nodes with maximal slack value s^* , P , induces a subgraph $G_p(P, E')$ in G .

Assume: $k_{tc} = 10\text{ns}$

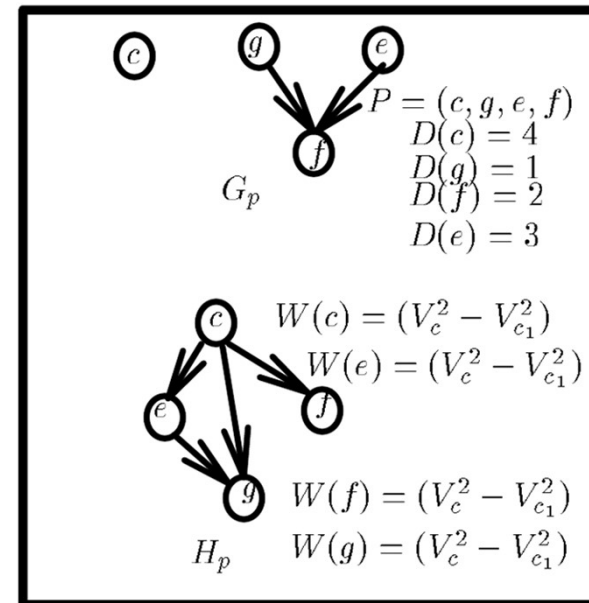


Source: Sarraf [1995]

Step 4: Dual Graph of H_p

Step 4: *Dual graph H_p*

A dual graph $H_p(P, E_p)$ is obtained for the graph $G_p(P, E')$: Obtain a Depth-First Search (DFS) ordering of the graph G_p . Let $D(v)$ be the order in which the node v is visited during the DFS. The dual graph, H_p , is constructed on the node set P . If $u, v \in P$ and there does not exist a path from v to u and from u to v in G_p , then if $D(u) > D(v)$ then $(u, v) \in E_p$.



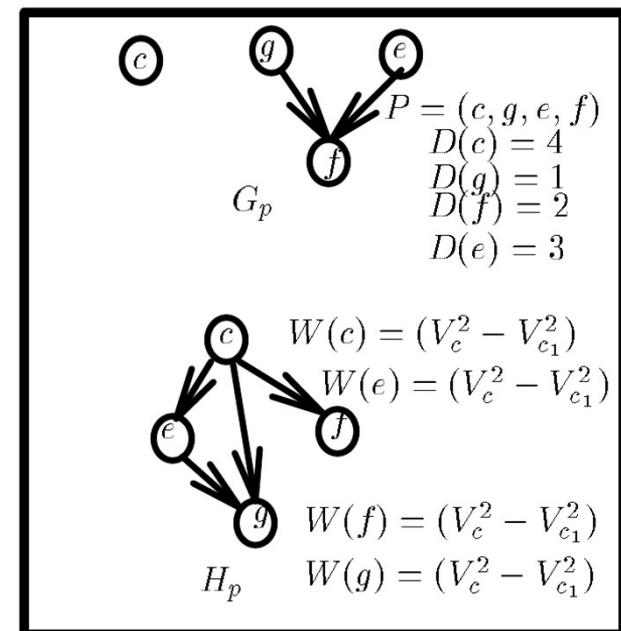
Source: Sarraf [1995]

Step 5: Weight Assignment

Step 5: *Weight assignment*

If a node $v \in P$ is assigned a voltage $\tau(v) = V_{c_k}$ then assign a weight $W(v) = (V_{c_k}^2 - V_{c_{k+1}}^2)$ in H_p .

$$P_{\text{dyn}} \sim V_{\text{dd}}^2 C_{\text{load}} f_{\text{switch}}$$

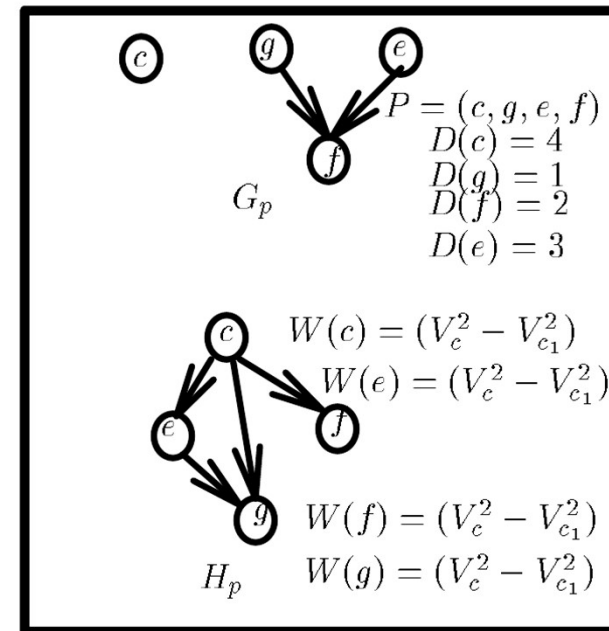


Source: Sarraf [1995]

Step 6: Longest Weighted Graph

Step 6: *Longest weighted path*

Obtain a longest weighted path in H_p . The longest weighted path in a directed acyclic graph is defined as the path in the graph which has the maximum total node weight and it can be obtained using a single *breadth first search*.

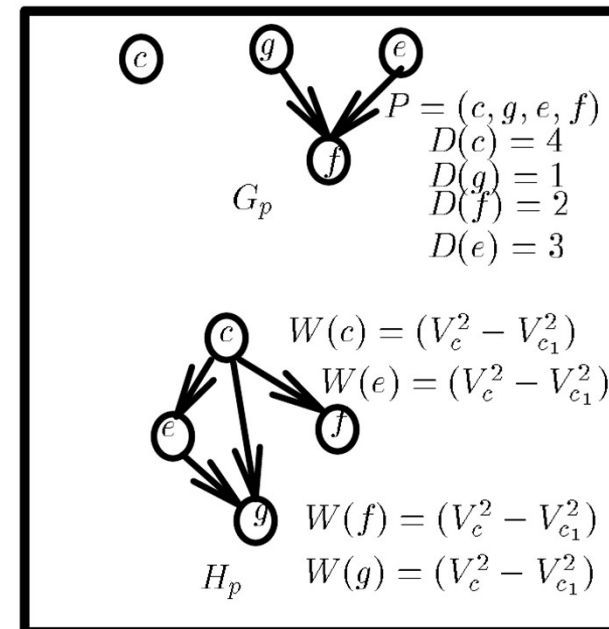
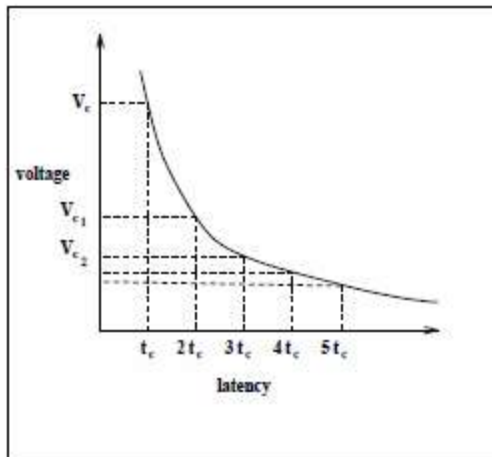


Source: Sarraf [1995]

Step 7: Reassigning Voltages to Nodes in Longest Path

Step 7: *Reassigning voltages to nodes in the longest path*

Reassign voltages to nodes in the longest weighted path obtained in the previous step. If a node in the longest path, v , has a prior voltage assignment $\tau(v) = V_{c_k}$ then change this assignment to $\tau(v) = V_{c_{k+1}}$. The new assignment of voltages to nodes in P changes the delay values ($d(v)$ values) for nodes.



Source: Sarraf [1995]

Theorem 1: Correctness

Theorem 1 *Given a set of allowable voltages S and data flow graph $G(V, E)$, the above algorithm produces a mapping $\tau : V \rightarrow S$ that minimizes $\sum_{v_i \in V} \tau(v_i)^2$.*

Source: Sarraf [1995]

Theorem 2: Complexity

Theorem 2 *The time complexity of the algorithm is $O(kn^2)$, where kt_c is the timing constraint and n is the number of nodes in G .*

Source: Sarraf [1995]

Experimental Results: Simulation using HLS

Bench- mark	Timing Constraint k	Power using 5V	Power using 5V, 3V	x1 % reduc.	Avg. % reduc.	Power using 5V, 3V, 2.4V	x2 % reduc.	Avg. % reduc.
Diffeq	4 †	275	195	29.1	40.73	195	29.1	44.56
	5	275	179	34.91		172.52	37.27	
	6	275	147	46.55		130.8	52.44	
	7	275	131	52.36		111.56	59.43	
FIR	9 †	525	349	33.53	40.38	326.32	37.84	46.4
	10	525	317	39.62		287.84	45.17	
	11	525	301	42.67		265.36	49.46	
	12	525	285	45.71		246.12	53.12	
AR-Lattice Filter	8 †	700	604	13.71	27.43	584.56	16.49	30.20
	9	700	540	22.86		520.56	25.63	
	10	700	476	32.0		456.56	34.77	
	12	700	412	41.14		392.56	43.92	
EWFilter	15 †	850	690	18.82	25.88	677.04	20.35	27.88
	16	850	642	24.47		629.04	25.99	
	17	850	610	28.24		590.56	30.52	
	18	850	578	32.00		555.32	34.67	

Table 1: Power Consumption Results for smaller Timing Constraints

†: Corresponds to the longest path length for the *DFG*.

Source: Sarraf [1995]

Experimental Results 2 : Higher Timing Constraints

Bench- mark	Timing Constraint k	Power using 5V	Power using 5V, 3V	x1 % reduc.	Avg. % reduc.	Power using 5V, 3V, 2.4V	x2 % reduc.	Avg. % reduc.
Diffeq	8	275	99	64	64	82.8	69.89	73.43
	12	275	99	64		63.36	76.96	
FIR	18	525	189	64	64	150.12	71.41	74.19
	27	525	189	64		120.96	76.96	
AR-Lattice Filter	16	700	252	64	64	232.56	66.77	71.87
	24	700	252	64		161.28	76.96	
EWFilter	30	850	306	64	64	280.08	67.05	72.00
	45	850	306	64		195.84	76.96	

Table 2: Power Consumption Results for larger Timing Constraints

Source: Sarraf [1995]

Module Selection

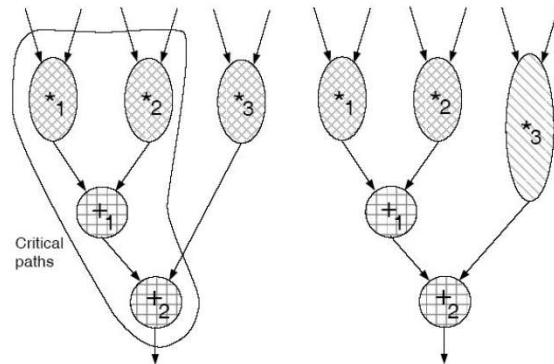
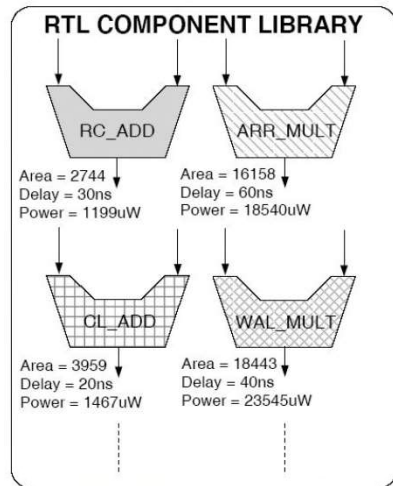
- The process of mapping operations from the CDFG to component templates from the RTL Library.
- Initially, only a functional unit template, not a specific instance, associated with each operation.
- Example: “+” Operation may be implemented using
 - ripple-carry adder (slower, but more switched capacitance efficient)
 - carry-lookahead adder (faster, incurs higher switched capacitance)
 - carry-select adder
 - ...
- Similar tradeoffs for other operations.
- Idea: Exploit tradeoffs to fulfill power constraints through module selection.

Module Selection 2

- Each operation in the DFG (middle) has been mapped to fast component in order to meet performance constraints (in that case constraint: 85ns).
- But is that really necessary? => no, not all ops need necessarily be mapped to fastest module. Focus (for timing constraint) should rather be on critical path.
- Idea: slack in off-critical path ops may be used to select slower functional units that may have a better efficiency in switched capacitance (see right DFG). There, mult-op uses less power (but not less energy).
- Important: to have a large module library with distinct switching capacity efficiencies and performance characteristics

Source: Anand [2012]

Module Selection 3



Source: Anand [2012]

Source

- Raghunathan, Anand, Niraj K. Jha, and Sujit Dey. *High-level power analysis and optimization*. Springer Science & Business Media, 2012.
- **Homework >>** Venkatesh, Ganesh, et al. "Conservation cores: reducing the energy of mature computations." *ACM SIGARCH Computer Architecture News*. Vol. 38. No. 1. ACM, 2010.
- **Homework >>** Raje, Salil, and Majid Sarrafzadeh. "Variable voltage scheduling." *Proceedings of the 1995 international symposium on Low power design*. ACM, 1995.