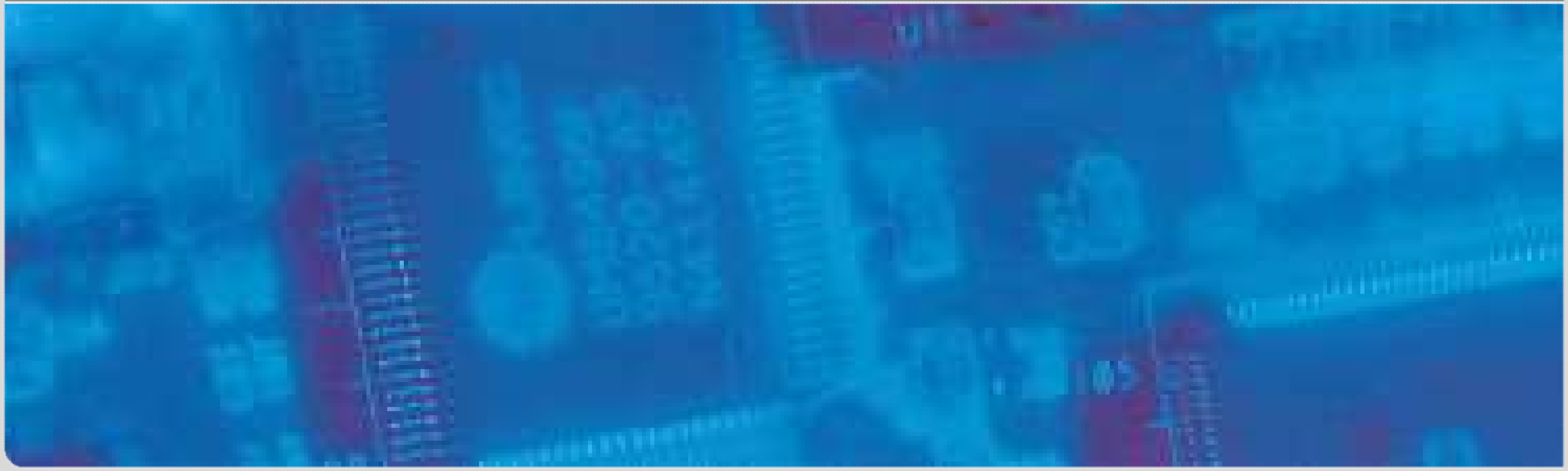
Lower Power Design

Lecture 6: Hardware Power Optimization and Estimation 3

Anuj Pathania on behalf of Prof. Dr. Jörg Henkel

Summer Semester 2017

CES – Chair for Embedded Systems



ces.itec.kit.edu

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 – ~~Lecture 0: Introduction~~
- 04.05.2017 – ~~Lecture 1: Energy Sources~~
- 11.05.2017 – ~~Lecture 2: Battery Modelling Part 1~~
- 18.05.2017 – ~~Lecture 3: Battery Modelling Part 2~~
- 25.05.2017 – Ascension Day (Holiday)
- 01.06.2017 – ~~Hardware Power Optimization and Estimation 1~~
- 08.06.2017 – ~~Hardware Power Optimization and Estimation 2~~
- 15.06.2017 – Corpus Christi (Holiday)
- 22.06.2017 – Hardware Power Optimization and Estimation 3
- 29.06.2017 – Software Compiler
- 06.07.2017 – Thermal Management
- 13.07.2017 – Aging-Aware Design
- 20.07.2017 – Temperature-Aware Design
- 27.07.2017 – Infrared Camera based Thermal Measurement

Overview for Today

- Recap: Clock Gating.
- Power Gating.
- Pre-Computation.
- Reducing Power by Scheduling.
- Operand Isolation.
- Register Sharing.
- Reducing Power Through the Controller

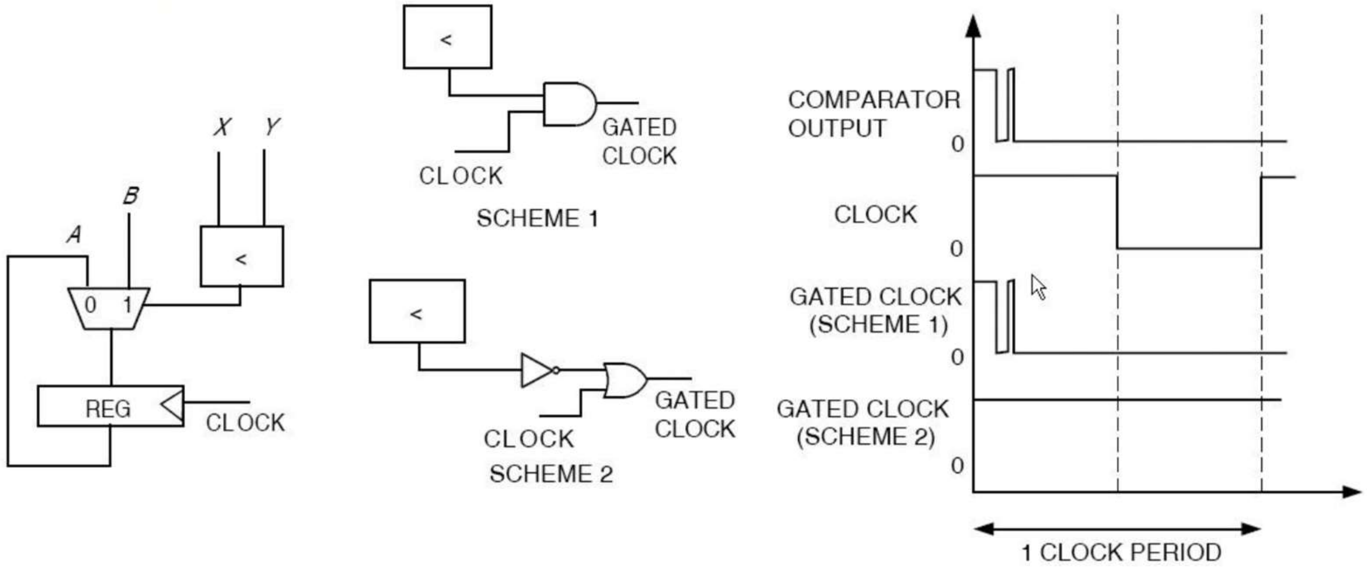
Recap: Clock Gating

- What is Clock Gating?
  - Suppress or disable transitions from propagating to parts of the clock network under specific conditions that are determined by clock gating circuitry.
  - Saves dynamic power but not leakage power.
- How can clock gating result in power savings?
  - Reduced capacitive switching in the clock network like
    - Clock buffers.
    - Interconnect of the clock network.
    - Latches/registers that are fed by the clock signal.
  - Also:
    - May prevent storage elements from loading unnecessary new values and thus saving power.

Source: Raghunathan [1998]

Recap: Clock Gating 2

- Register re-loads previous value when comparator output is ‘0’
  - Register changes value only with clock signal (can be set to work with zero or one).
  - Clock can be gated to prevent reloading value.
  - Scheme 1: Register clock input would be forced to ‘0’ when comparator is ‘0’.
  - Scheme 2: Register clock input would be forced to ‘1’ when comparator is ‘0’.
  - Scheme 2 better than Scheme 1 because comparator output instability forwarded.

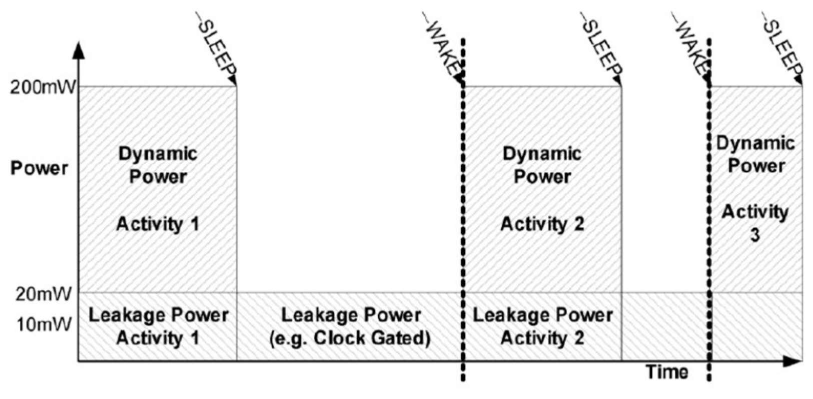


Source: Raghunathan [1998]

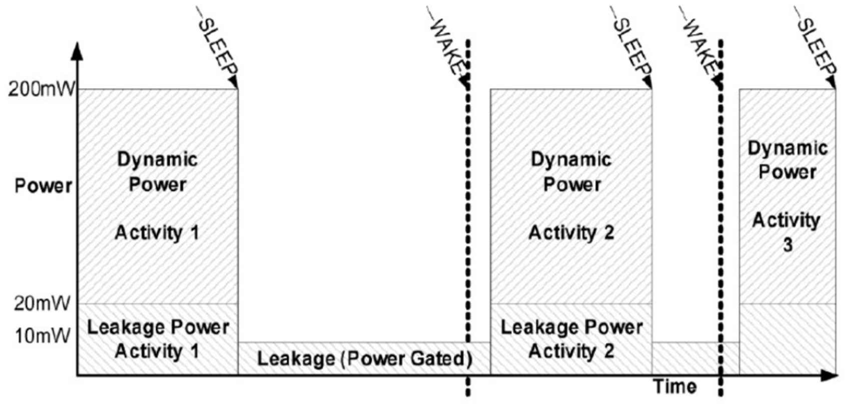
Power Gating

- Power Gating:
  - Bring both V_{dd} (Supply Voltage) and ground to the same voltage potential when circuit is idle.
- Advantages over Clock Gating:
  - Reduce not only switching power but also leakage power.
- Disadvantages over Clock Gating:
  - Slower (power gating requires time to drain/load capacitances of gated circuit).
  - Circuit state is lost.

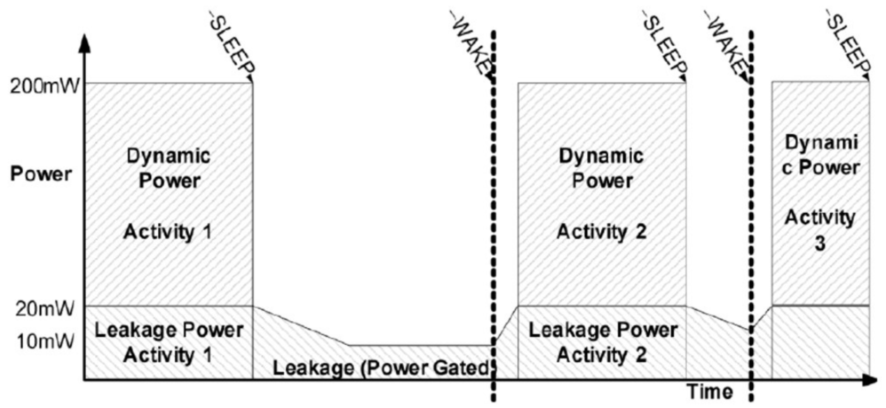
Power Gating 2



Clock Gating



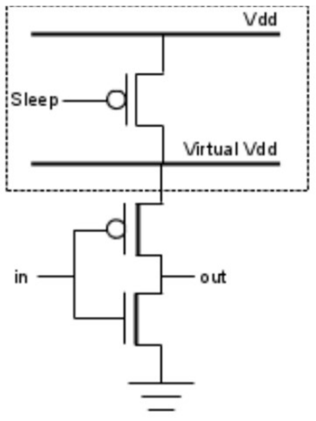
Power Gating

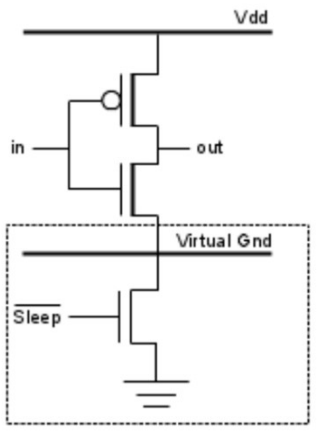


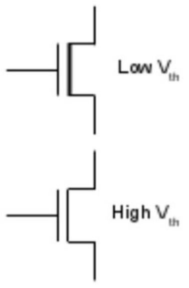
Power Gating (Real-World)

Source: Keating [2007]

Power Gating 3



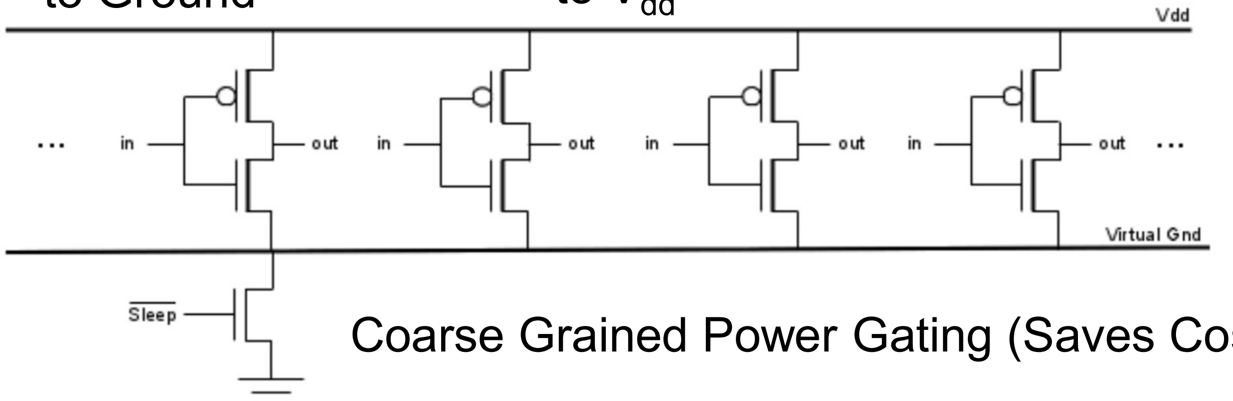




Gating transistor in High V_{th} to reduce leakage.

Header: V_{dd} is pulled to Ground

Footer: Ground is pulled to V_{dd}



Coarse Grained Power Gating (Saves Cost)

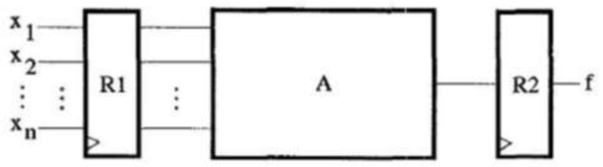
Source: Keating [2007]

Power Gating 4

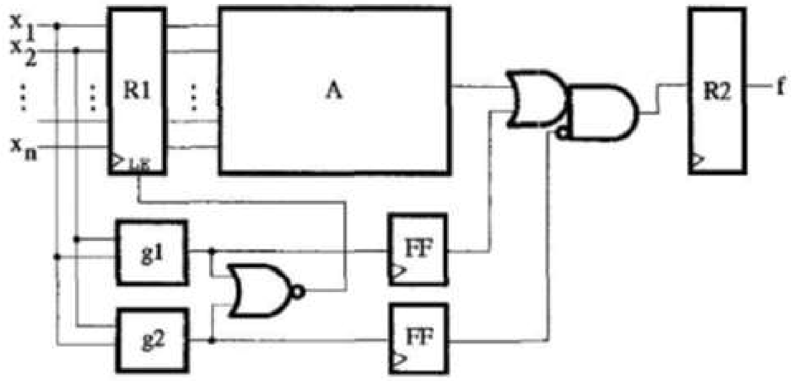
- **Highest time** requirement is loading circuit capacitance.
  - Header: Most delay is at the end of power gating (“turning on”).
  - Footer: Most delay is at the start of power gating (“turning off”).
- Trade-off between **sleep transistor** size and speed of gating
  - Large transistors can load/drain capacitances much quicker.
  - But also consume more leakage power themselves.
- Dynamic power is consumed when turning on/off power gating.
  - If power gated interval too low; may result in increase in power consumption.
  - Break-even time can be determined.

Power Savings through Pre-Computation

- Pre-compute (i.e. predict) output of a circuit one cycle ahead with additional logic and then switch off original logic.
  - For a majority of input values, the output might be computed with very simple logic but in order to cover all input, complex logic is necessary.



Original Circuit



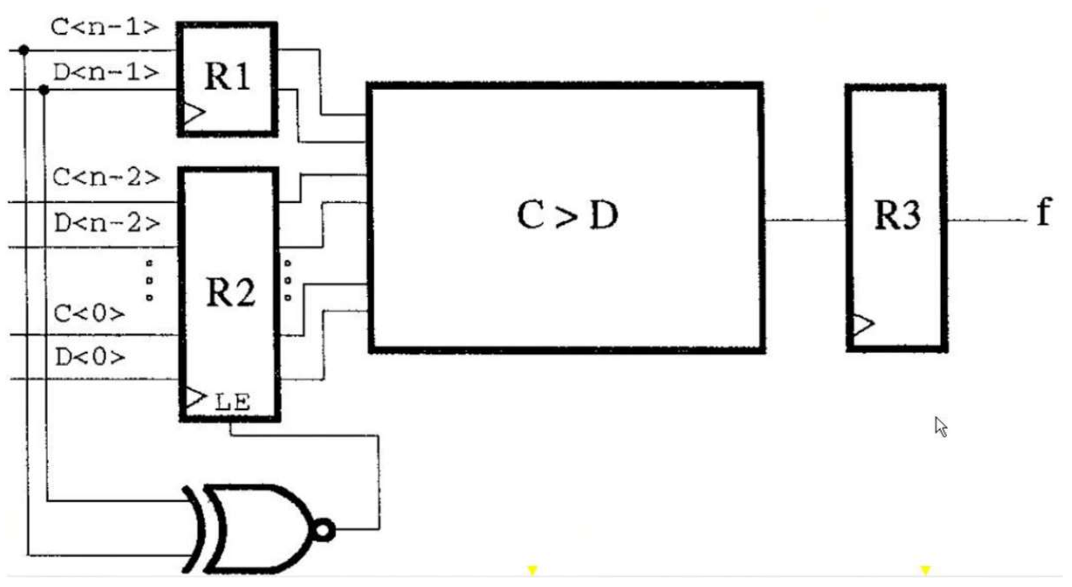
Pre-computational Circuit

  - Tasks of g1, g2: a) pre-compute, b) switch off original circuit in certain cases when prediction is possible.
  - g1, g2 only cover a subset of all x1,…,xn (desirable: a large coverage) b) imposes overhead in form of power area and probably performance.

Source: Alidina [1994]

Power Savings through Pre-Computation 2

- Example: A comparator of two n-bit values C and D that results in ‘1’ if C>D.
  - Precomputation can be defined as to test the MSB (Most Significant Bit) using XNOR gate.



  - With little effort the circuit can be predicted and power can be saved by switching off (gating) the original circuit.

Source: Alidina [1994]

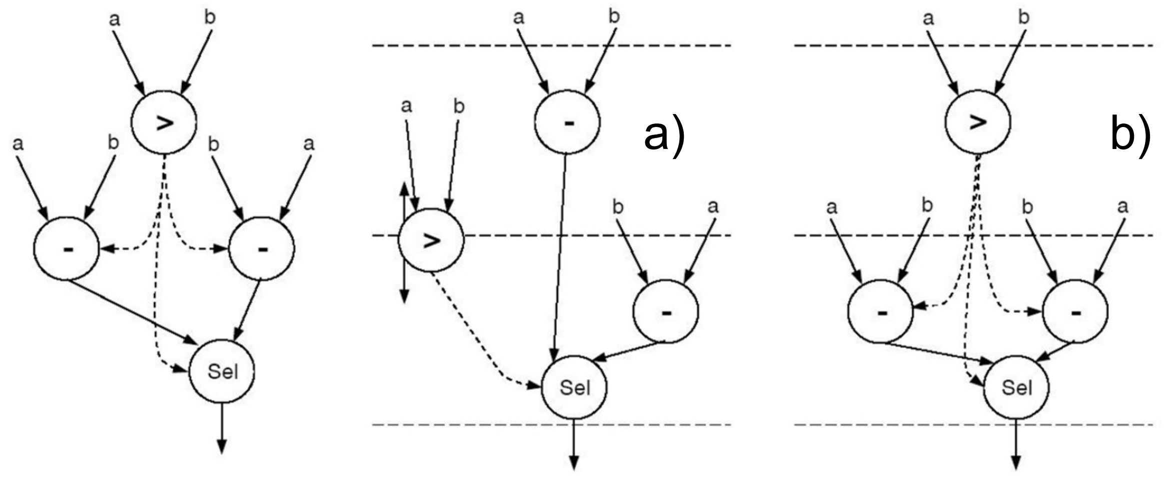
Managing Power Through Scheduling

- **Recall:** “pre-computation” is a shut-off technique based on clock cycle basis and is limited to the given structure of the logic. Can power be managed at a higher level?
- **Observation:** it is common for performance optimization to compute all outcomes of a conditional operation **in parallel** with the condition evaluation itself. The appropriate result will be chosen and the other one(s) will be discarded.
  - This is ineffective in terms of power consumption.
- **More power effective:** enforce control dependencies between operations in CDFG and conditional operation such that they depend during scheduling.
  - May be accomplished by meeting performance constraints first and then optimize power consumption.

Source: Raghunathan [1998]

Managing Power Through Scheduling 2

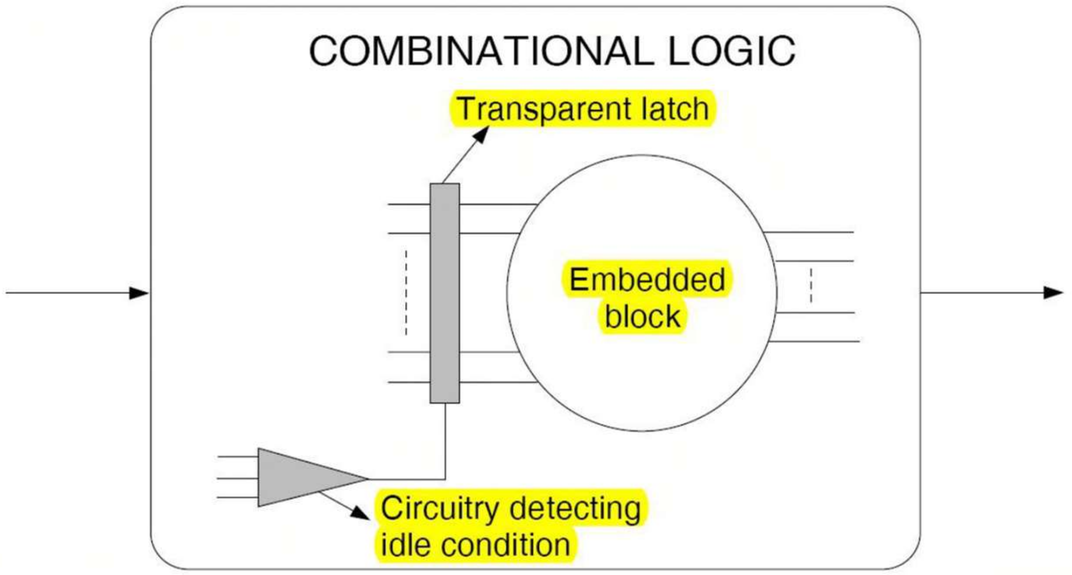
- Example: Evaluate expression |a – b|.
  - **Assumption:** Each op '-' and '>' takes one clock cycle and 'sel' operation may be chained with any other operation.
  - **Constraint:** a schedule within 2 cycles (2 Possibilities)
  - **Schedule a):** ignores control dependencies; a-b and b-a are executed independently of '>'; There is a flexibility in scheduling '>'. Problem: from power point of view b) is inefficient: both a-b and b-a are always executed.
  - **Schedule b):** a-b or b-a are activated exclusively due to outcome of '>'. a-b, b-a may be assigned to same or to two different subtractors (latter case: one needs to be shut down).



Source: Raghunathan [1998]

Power Saving Through Operand Isolation

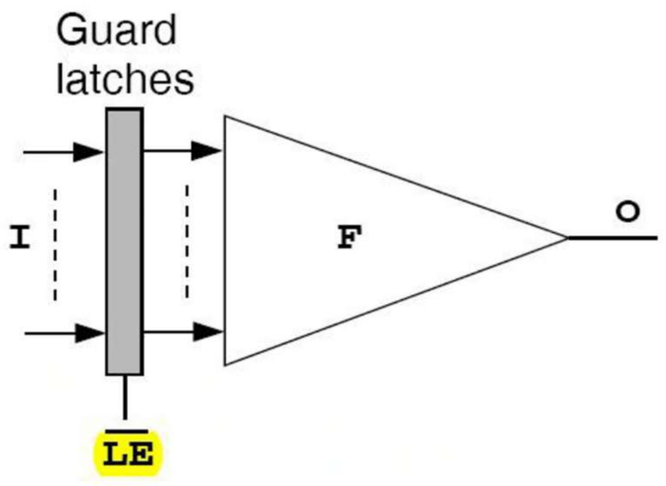
- Operand Isolation:
  - Disable transitions at inputs of variables.
  - Insert transparent latches at all inputs of embedded block.
  - If block does not perform any useful operation: a) transparent latches at inputs are disabled, b) retain previous cycle’s values.
    - avoids unnecessary power consumption.



Source: Raghunathan [1998]

Power Saving Through Operand Isolation 2

- Guarded Evaluation:
  - **o** – signal in a combinational circuit.
  - **F** – logic that is computing o.
  - **I** – set of inputs to F.
  - **ODC$_o$** – observability don't care set with respect to o, i.e. set of primary input assignments to the entire circuit such that the value of o has no influence on the values at the primary outputs.
  - **LE** – an arbitrary value of the existing circuit such that LE ⇒ ODC$_o$, i.e. Thus, when LE = 1, the value on o is not needed to compute the primary outputs.



Source: Raghunathan [1998]

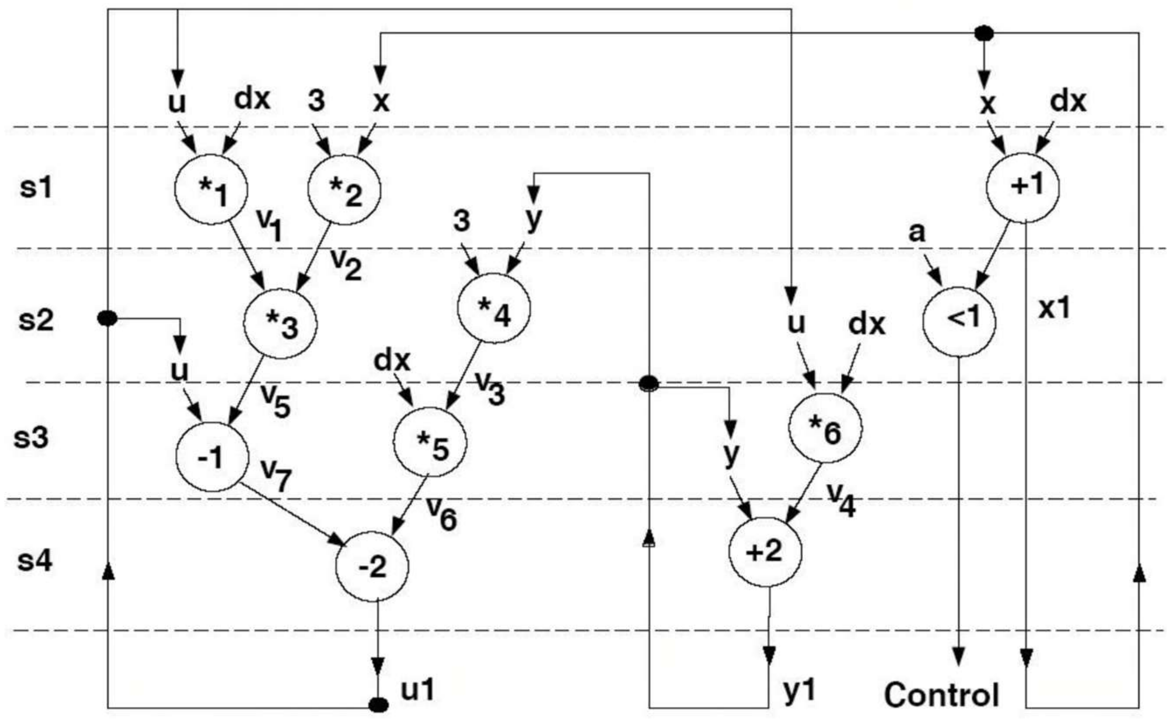
Power Saving Through Operand Isolation 3

- Guarded Evaluation advantages over Pre-Computation:
  - Pre-computation needs additional circuitry; Guarded Evaluation is derived from within the circuit.
  - Pre-computation may require re-synthesis to efficiently derive additional circuits whereas Guarded Evaluation leaves circuit as is (especially important in hand-optimized circuitry).

Source: Raghunathan [1998]

Power Saving Through Operand Isolation 4

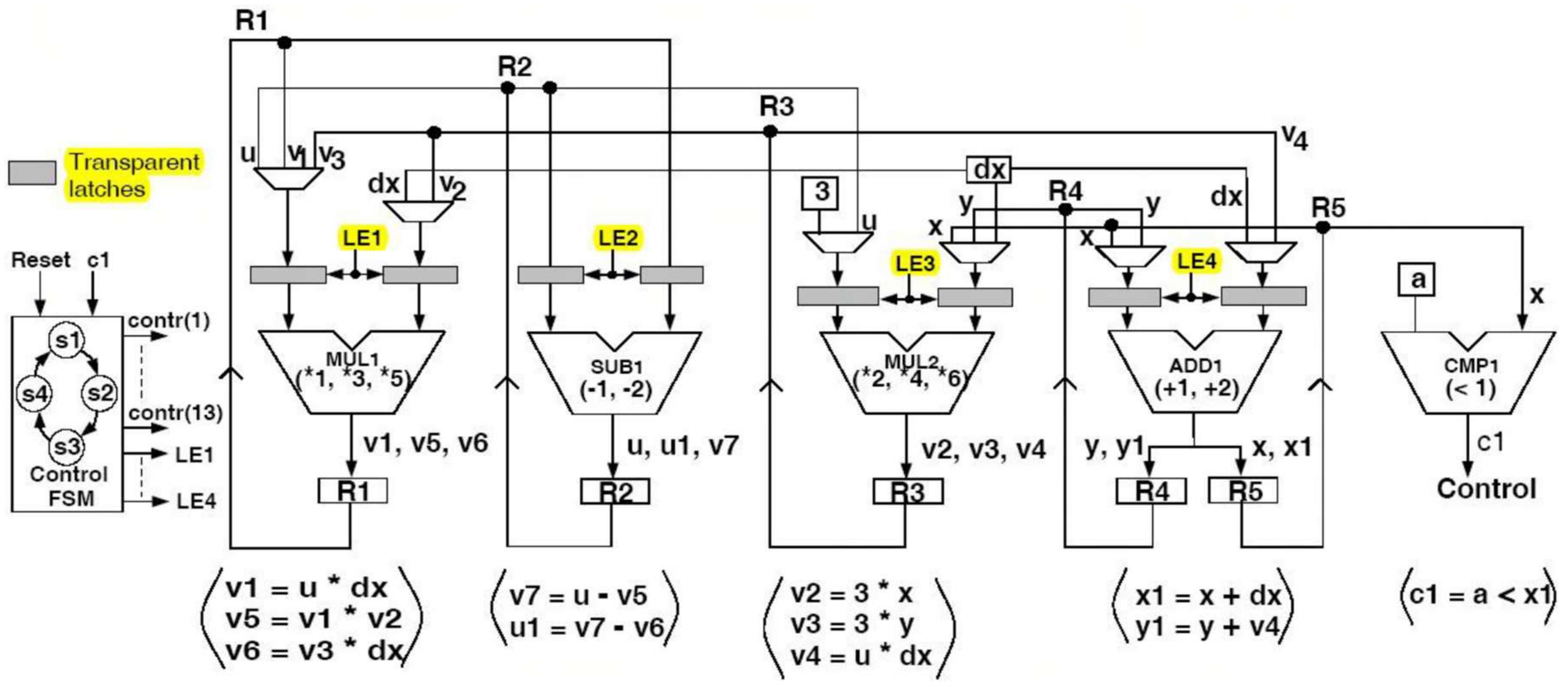
- Idea: apply operand isolation from logic level to high-level synthesis.
- In fact: the conditions under which a resource (e.g. a functional unit FU) is not used are readily available from the scheduling and resource sharing!



Source: Raghunathan [1998]

Power Saving Through Operand Isolation 5

- Corresponding RTL



Source: Raghunathan [1998]

Power Saving Through Operand Isolation 6

- Idle cycles of FUs
  - MUL1, MUL2: s4
  - ADD1 : s2, s3
  - SUB1 : s1,s2
  - CMP1 : s1, s3, s4
- Insert transparent latches at the FU's input to perform operand isolation.
  - LE1 = LE3 = x4
  - LE2 = x1 + x2
  - LE4 = x2 + x3
- No latches at the inputs of CMP1 as input values do not change in idle cycles.

Source: Raghunathan [1998]

Power Saving Through Operand Isolation 7

- Operand Isolation disadvantages:
  - Incurring **power and area overheads** due to the addition of extra circuitry.
  - Operand isolation also requires some **delay constraints** (the disabling transition at the transparent latch enable input should arrive before its data input can change).
  - Satisfaction of the delay constraints may require the addition of extra circuit delay in the critical path, which **may not be acceptable for high-performance designs**.

Source: Raghunathan [1998]

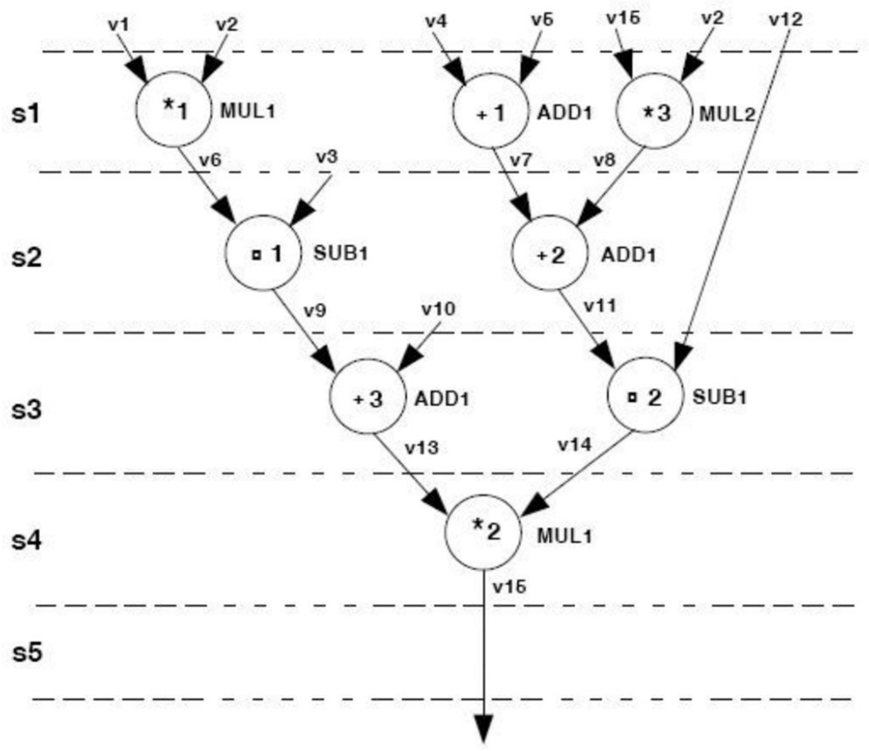
Power Saving Via Constrained Register Sharing

- **Idea**: rather than applying transparent latches to an already scheduled DFG, can't the scheduling, mapping, etc. already take into consideration power shut-down techniques?
- Impact of variable assignment on power consumption:
  - Two candidate assignments, Assignment 1 and Assignment 2, shown in next slide.
  - Architectures obtained using these assignments were subject to:
    - logic synthesis optimizations, and placed and routed using a cell library.
    - The transistor-level netlists extracted from the layouts were simulated using a switch-level simulator with typical input traces to measure power.
  - For the circuit Design 1, synthesized from Assignment 1, the power consumption was 30.71mW, and for the circuit Design 2, synthesized from Assignment 2, the power consumption was 18.96mW !

Source: Raghunathan [1998]

Power Saving Via Constrained Register Sharing 2

- Example: DFG and two possible register assignments that differ significantly in power consumption.

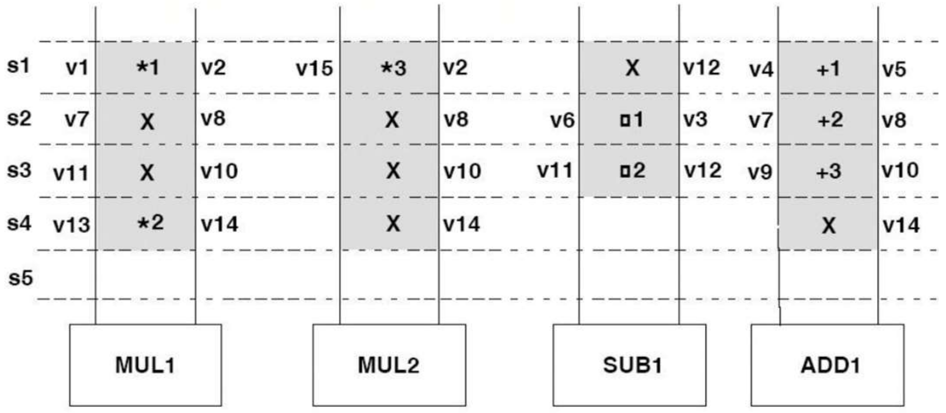


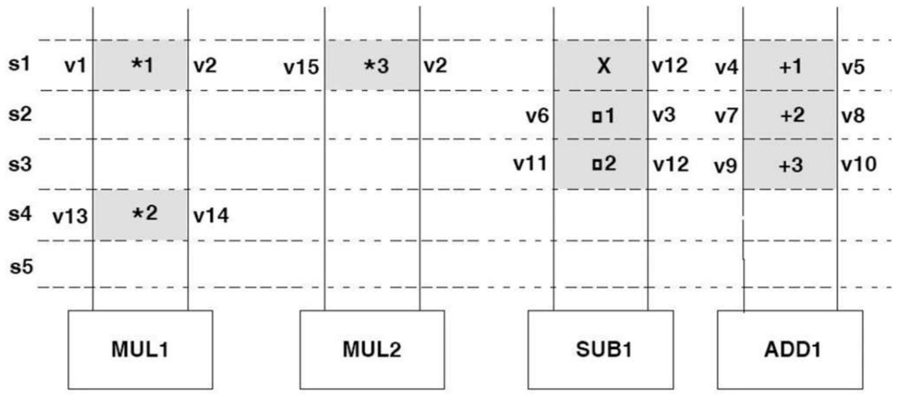
Register	Assignment 1	Assignment 2
R1	v1, v7, v11, v13	v1, v13
R2	v2, v8, v10, v14	v2
R3	v3, v5, v9	v4, v8, v10
R4	v4, v6	v5, v7, v9
R5	v12	v12
R6	v15	v3
R7	-	v6, v11
R8	-	v14
R9	-	v15

Source: Raghunathan [1998]

Power Saving Via Constrained Register Sharing 3

- **Two distinct schedules**: Shown: FU (in box), input variables left and right of operation, grey-shaded: variables at input of respective operation change value; spurious input transitions i.e. those that do not correspond to an DFG operations are marked with an 'X'.
- **Observation**: A functional unit that does not alter its input does not perform a spurious operation.
- **Conclusion**: Constrained register sharing can save significant energy/power/
  - Assignment 1 (Left): 7 operations that do not correspond to a DFG operation.
  - Assignment 2 (Right): only one such operation.
  - Note: Number of control steps has not increased and number of HW resources (FUs) is still the same.

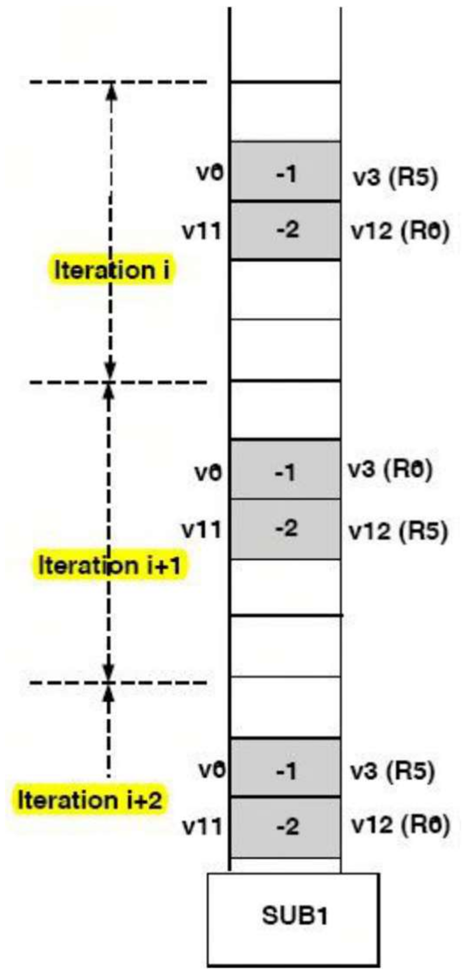




Source: Raghunathan [1998]

Power Saving Via Constrained Register Sharing 4

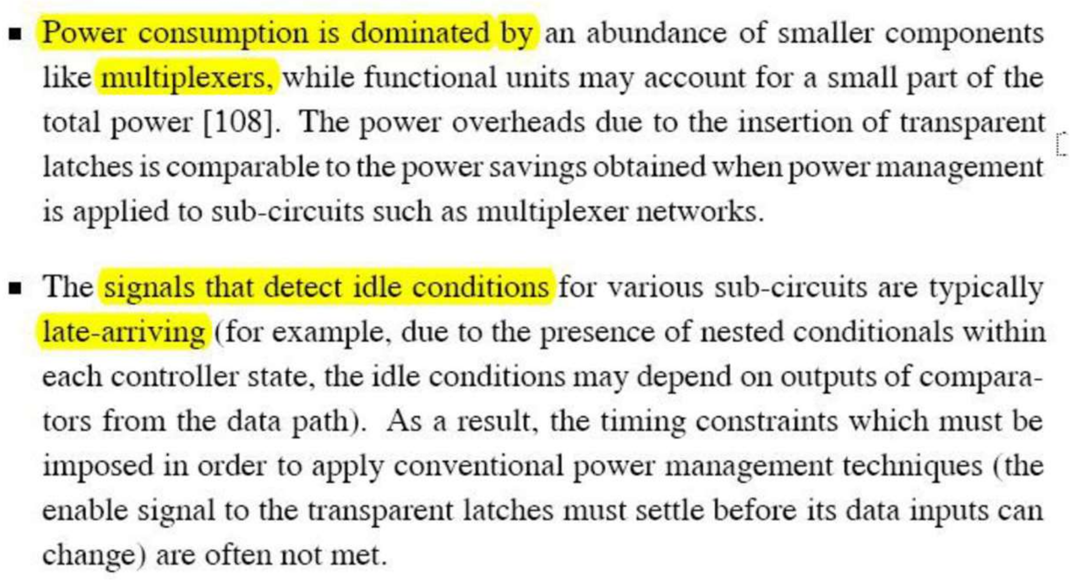
- **Problem:**
  - There is still a spurious operation is control step s1 of SUB1 operation.
  - MUX selects R5 (to which v12 is assigned) from control step s3 of each iteration to control step s1 of the next iteration.
  - v12 acquires a new value at step s1.
- **Idea:** Combine dynamic variable rebinding with variable assignment to completely eliminate spurious operation.
- **How**:
  - Need to preserve old (previous iteration) value of v12 at input of SUB1 until new value of v3 in current iteration is generated and then, spurious operation can be eliminated by swapping the variables assigned to registers R5 and R6.
- **Result:** see Figure on the right.



Source: Raghunathan [1998]

Managing Power Through the Controller

- **Idea:** Find way to reduce power without having to spend large overhead in terms of hardware (like transparent latches, etc.).
- **Rather**: Re-design existing control logic in order to reconfigure the multiplexer networks and functional units in the data path.
- Might not completely eliminate activity but is low-cost.
- Best suited to control flow intensive designs:

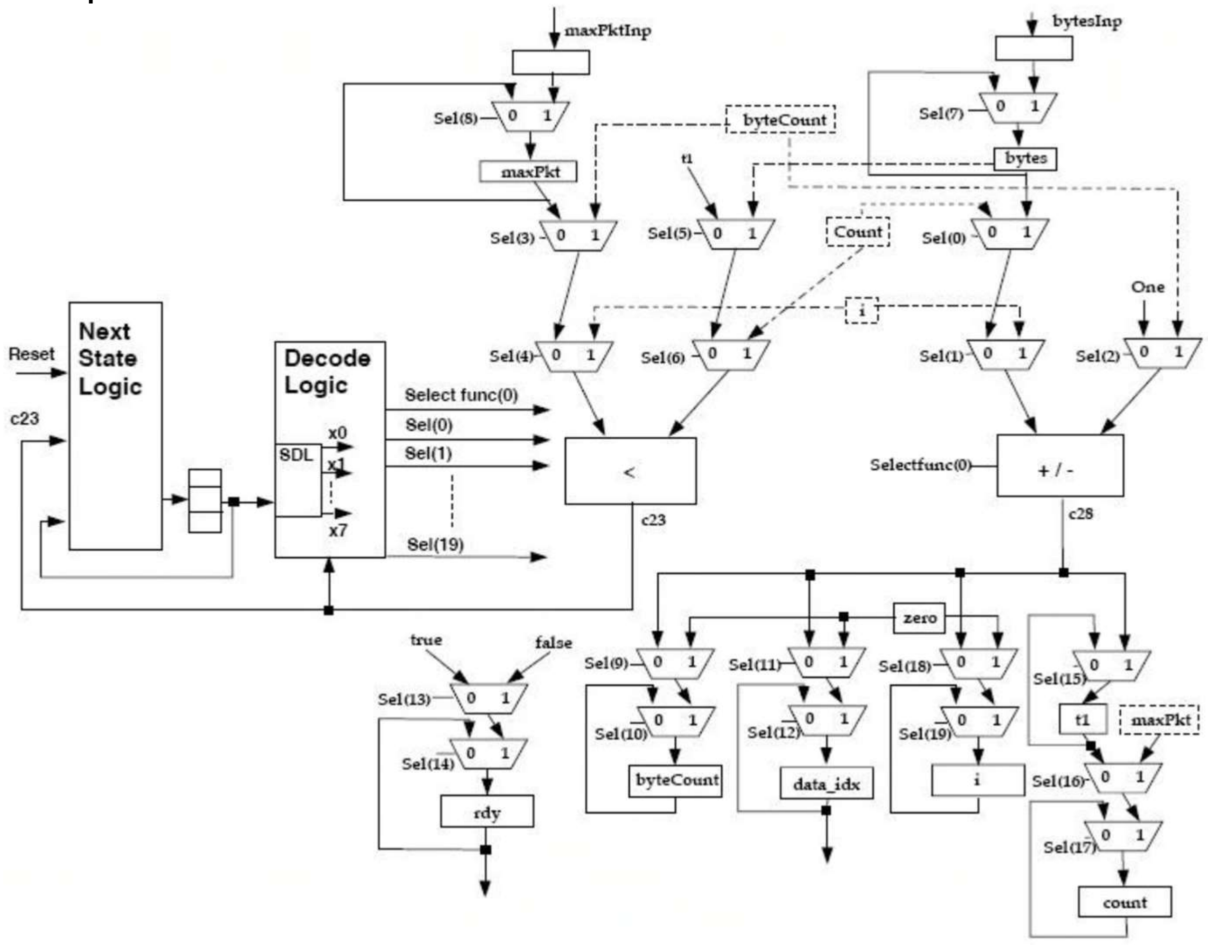


- Power consumption is dominated by an abundance of smaller components like multiplexers, while functional units may account for a small part of the total power [108]. The power overheads due to the insertion of transparent latches is comparable to the power savings obtained when power management is applied to sub-circuits such as multiplexer networks.
- The signals that detect idle conditions for various sub-circuits are typically late-arriving (for example, due to the presence of nested conditionals within each controller state, the idle conditions may depend on outputs of comparators from the data path). As a result, the timing constraints which must be imposed in order to apply conventional power management techniques (the enable signal to the transparent latches must settle before its data inputs can change) are often not met.

Source: Raghunathan [1998]

Managing Power Through the Controller 2

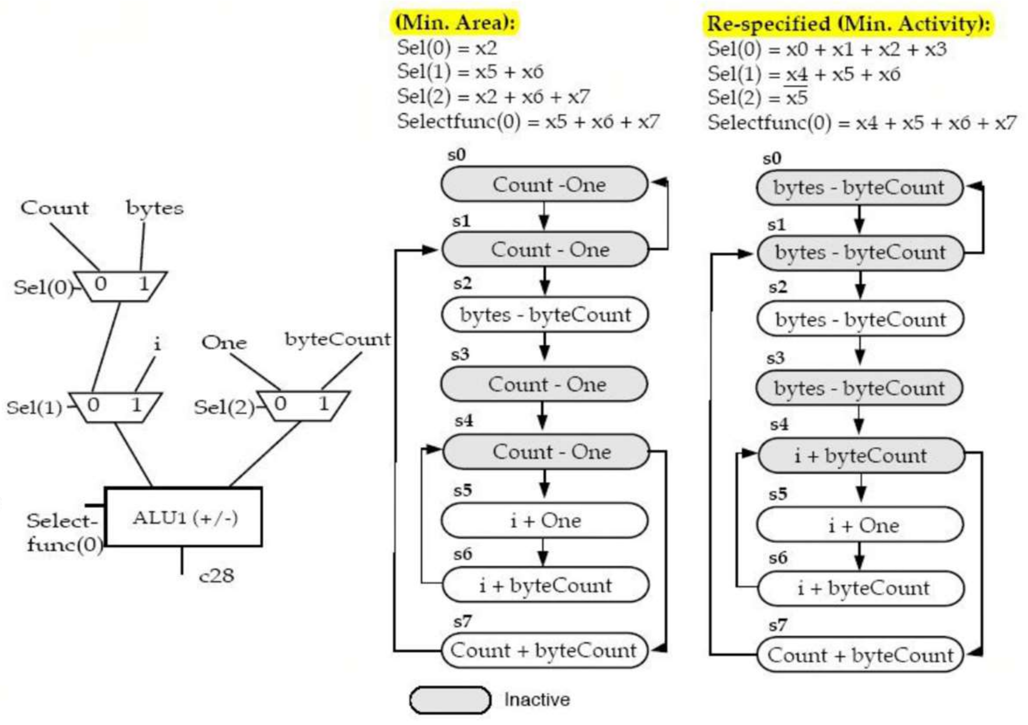
- **Example:** X.25 protocol



Source: Raghunathan [1998]

Managing Power Through the Controller 3

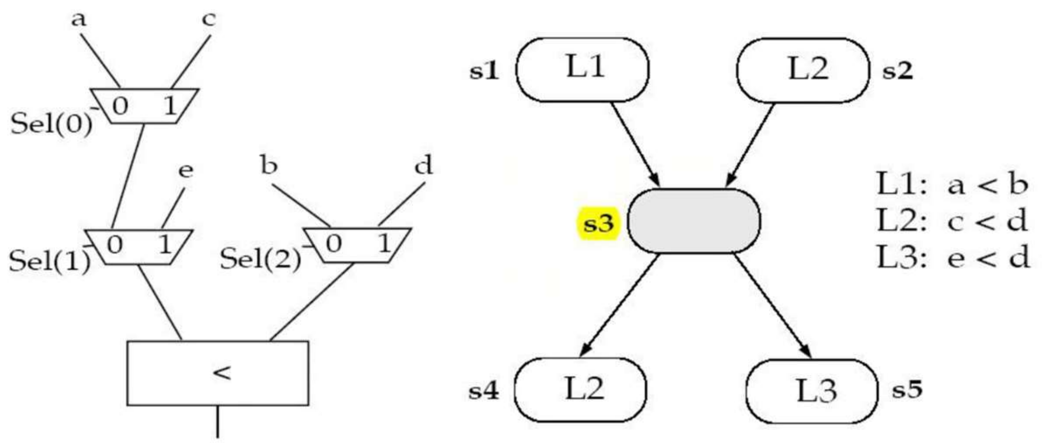
- **Shown:**
  - Left: a part of the data path.
  - Middle: logic expression for control signals (xi=1 => controller is in state si);
  - Right: activity graphs of ALU (state transitions with actions involved; ex: sel(0) sel(1), sel(2), SelectFunc(0) are 1, 0, 1, 0, respectively => “bytes-byteCount” is performed.
- **Observation**:
  - Some states are actually idle states (gray shaded). This can be found out through scheduling info from HL synthesis.
- **Idea**:
  - Re-specify idle states such that switching i minimized.
  - Example: state transition s6->s4 : same signals are on muxes such that same operands stay stable. Since they do not change no switching => no unnecessary power consumption.



Source: Raghunathan [2012]

Managing Power Through the Controller 4

- **Re-labeling activity graphs: How to label an idle vertex in an activity graph?**
  - Different incoming and outgoing transitions into the idle state have different execution probabilities.
  - The values of data operands fed to the mux trees may themselves change => Only selecting the same operand does not ensure that switching activity is minimized.



- Goal: Using s3 for one of the labels L1, L2, L3 such that activity at the input of the "<" is minimized.
  - P(si - > sj) - probability of the controller state transition from si to sj.
  - AM $_{si\text{->}sj}$ - activity matrix stores the cost of (average bit transitions for respective state transition).
- Nodes in the state transition graph are to be re-labeled while minimizing labeling costs:

$$Labeling\ Cost = \sum_{all si \rightarrow sj} AM_{si \rightarrow sj}[L(si), L(sj)].P(si \rightarrow sj)$$

Source: Raghunathan [1998]

Conclusion

- HW power sources:
  - Data path
  - Control path
  - Clock tree
- Optimization strategies:
  - Operator scheduling for low power
  - Hardware power management (clock gating)
  - Re-labeling of controller
- Very often there is a tradeoff – reduced power may lead to:
  - more logic
  - more complex design
  - reduced performance

Source: Raghunathan [1998]

Source

- Raghunathan, Anand, Niraj K. Jha, and Sujit Dey. *High-level power analysis and optimization*. Kluwer Academic Publishers, 1998.
- Flynn, D., Aitken, R., Gibbons, A., & Shi, K. (2007). *Low power methodology manual: for system-on-chip design*. Springer Science & Business Media.
- Homework >> Alidina, M., Monteiro, J., Devadas, S., Ghosh, A., & Papaefthymiou, M. (1994). Precomputation-based sequential logic optimization for low power. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2(4), 426-436.