Energy-Aware Computing

Lecture 12: Better than worst case computing

Outline

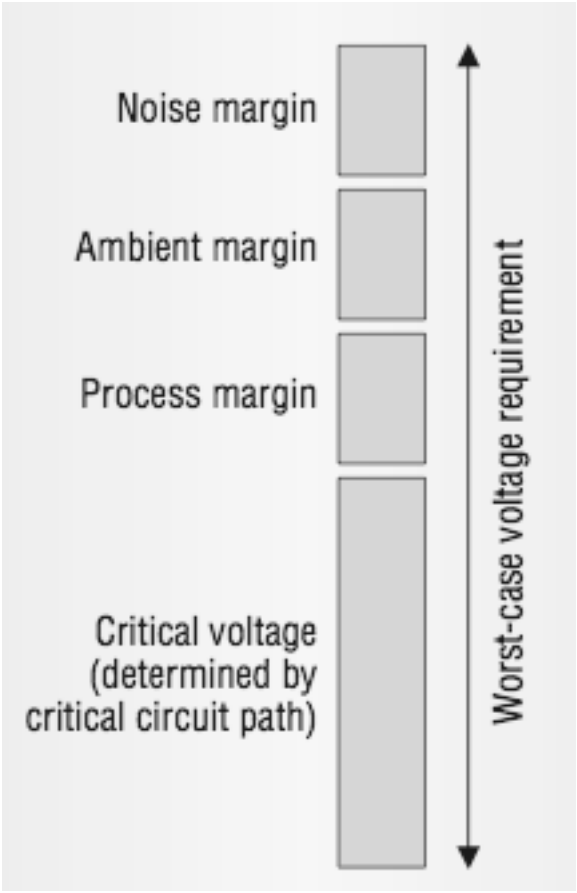
- Energy vs reliability
  - Scale voltage so low that errors may happen
  - TEAtime: Stop DVS just before errors appear
  - Razor: allow errors but recover without significant overhead
- Approximate circuits
  - Perform approximations to save energy

Design margins

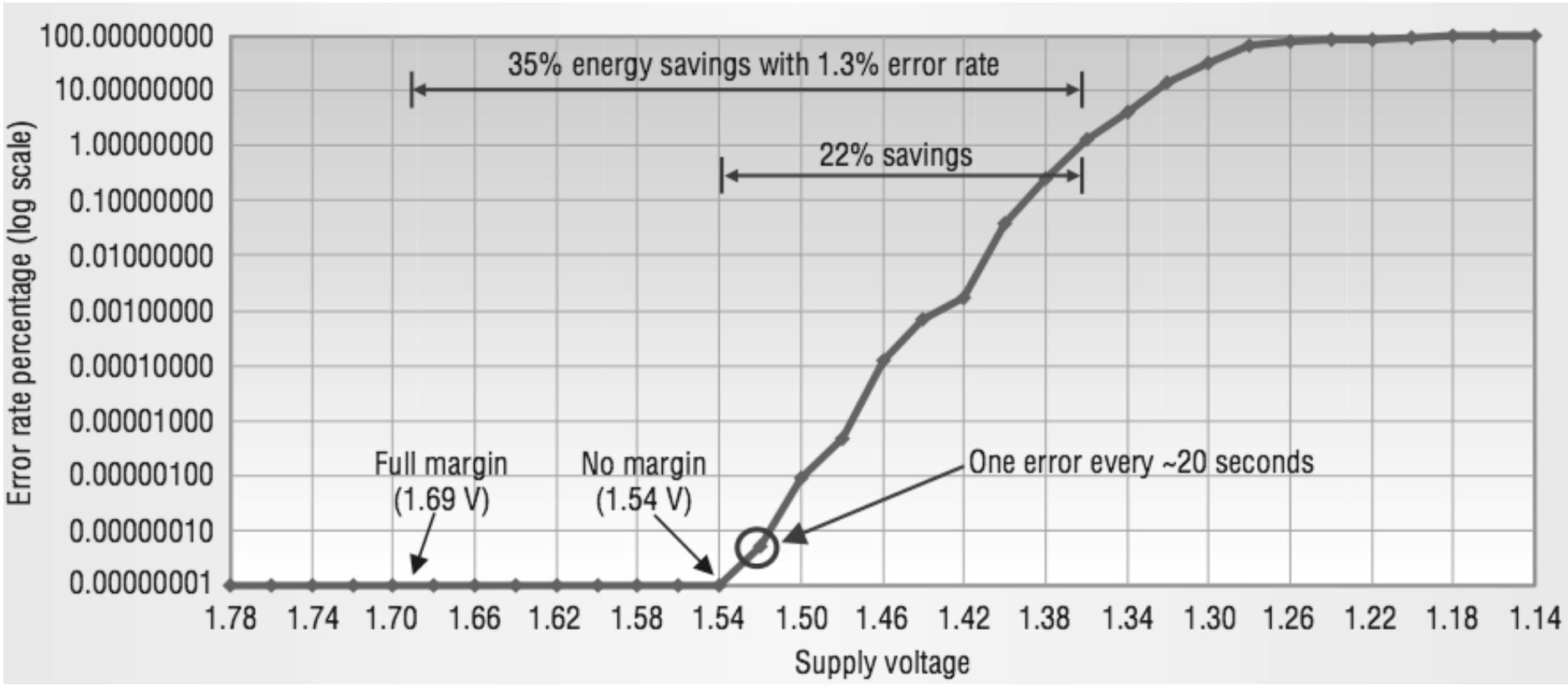
- Systems are designed for worst-case conditions
- Extra time margins are added due to:
  - Process
  - Ambient
  - Noise
- Design margins lead to a severe performance cost and/or power cost

Better than worst-case

- For power/energy, these margins can be viewed as supply voltage margins
  - Voltage is increased from the bare minimum to accommodate these time margins
- Most devices work in better than worst case conditions
- Can the margins be traded of for power/energy?



Effect of margins to energy

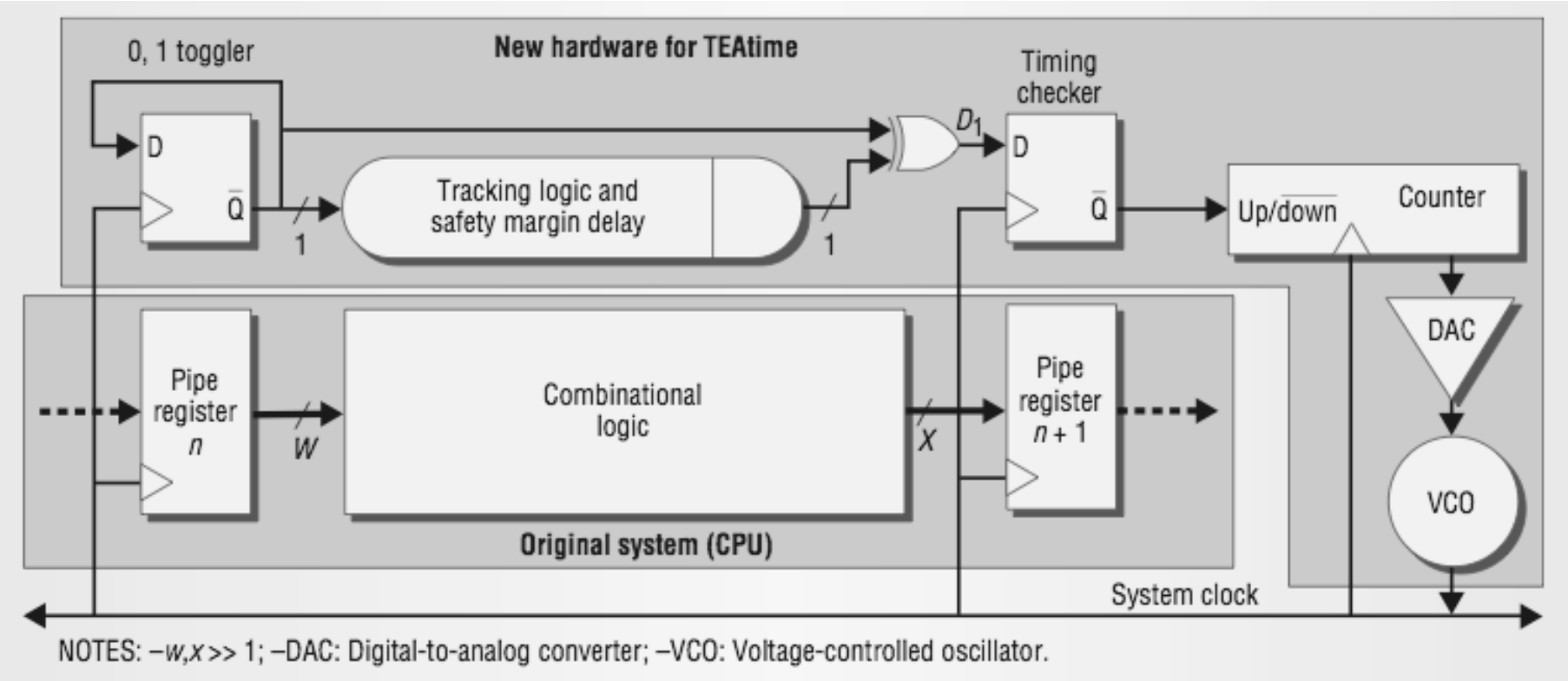


Experiments with multipliers implemented in FPGAs

Timing error avoidance (TEAtime)

- Increase the operating frequency just below the point when an error would occur
  - Or drop the voltage
- Similar to “over-clocking”, but safer
  - Feedback loop says when errors would occur
- Hardware only approach

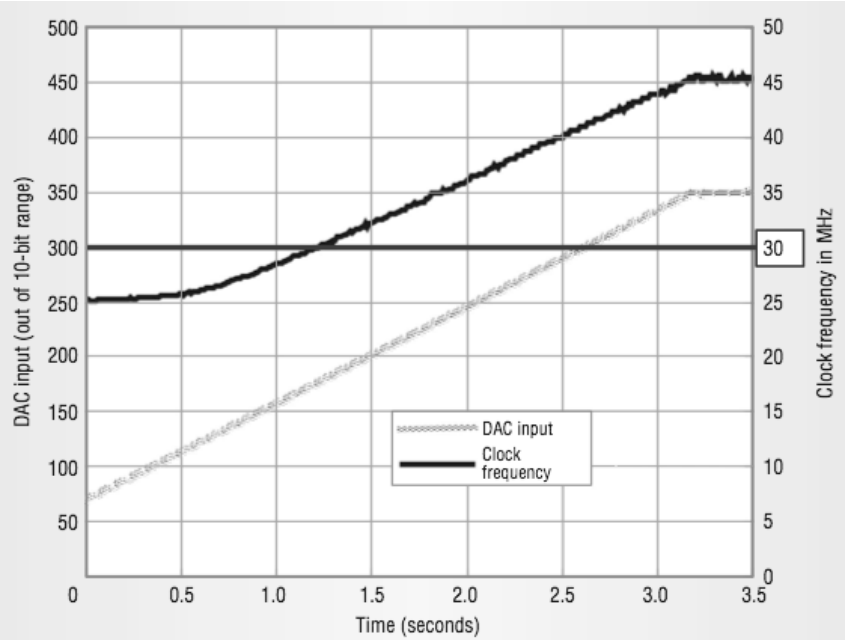
TEAtime tracker



- 1-bit wide replica of worst case path in system plus some extra delay
- D1 always transitions from 1->0 at end of cycle
- If FF is 1, wrong value read, drop speed

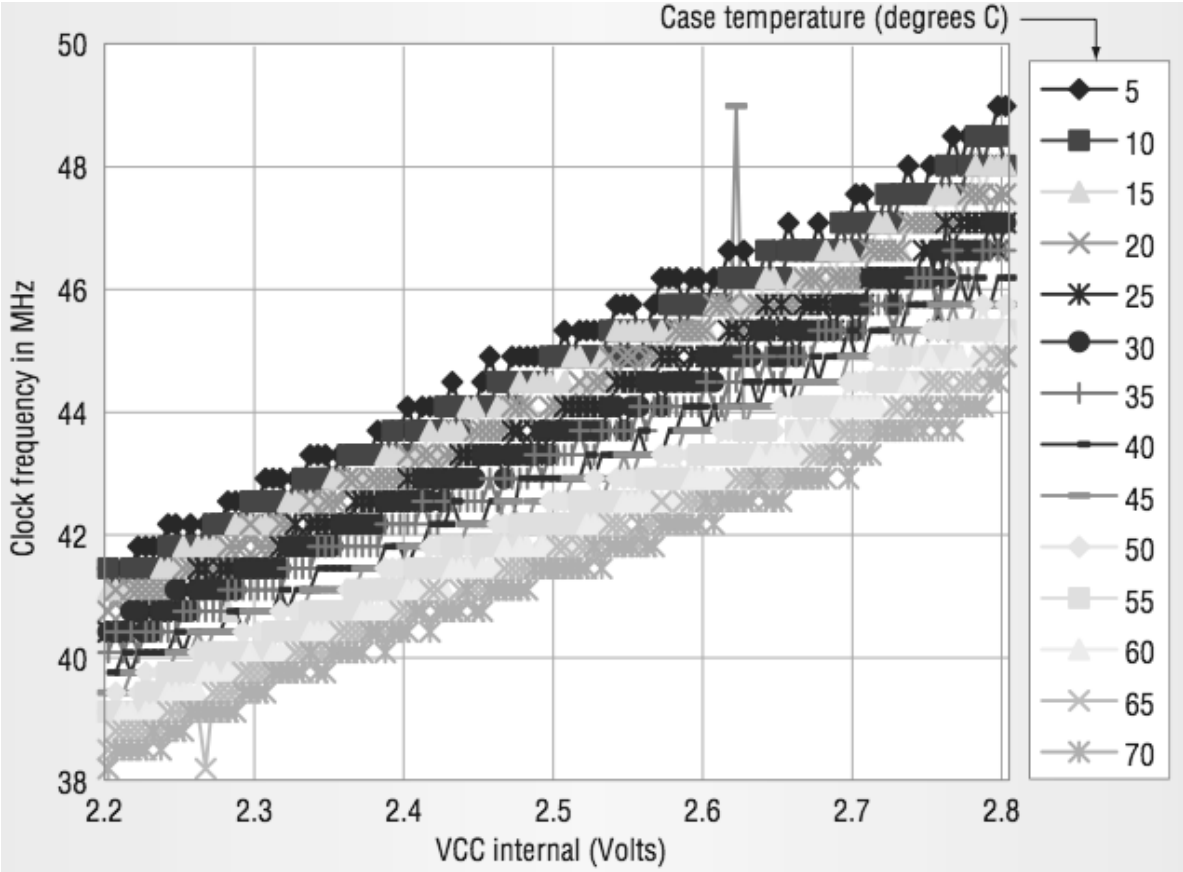
Teatime experiment

- FPGA implementation of simple 5-stage CPU
- Max clock determined from EDA tools as 30MHz



- TEA-time system ramped up clock from 25MHz upwards
- System frequency stabilised at 45MHz
- 50% speed improvement from worst-case

Teatime adaptability



- System clock can adapt with temperature and supply voltage

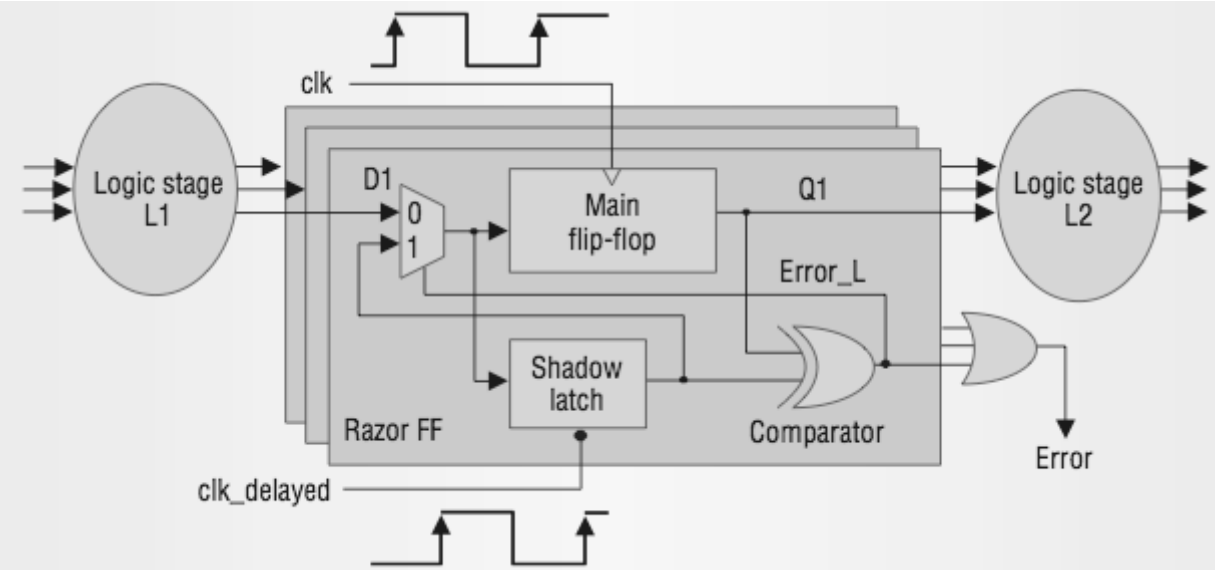
Teatime issues

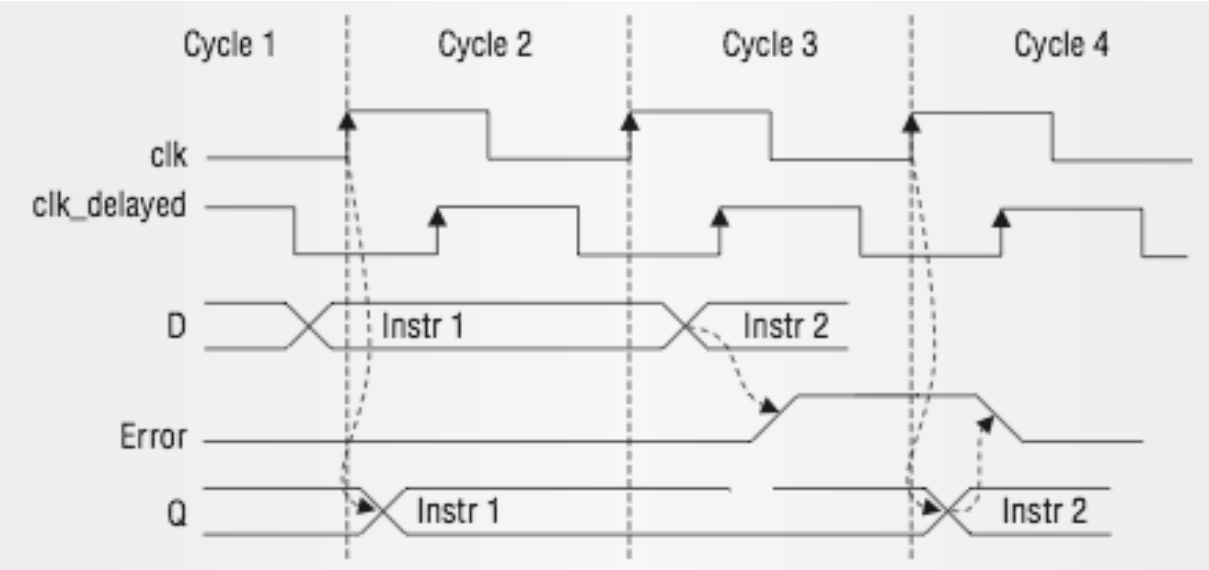
- Multiple worst case paths
  - Tracking circuit for each case
  - Only if *all* trackers agree, change clock freq
- Metastability
  - FF input changing too close to clock edge, output can go metastable: neither 1 nor 0
  - Metastability can last indefinitely
  - Solution: modify circuit to look into the timing of many cycles - gives time for metastability to settle

Razor

- If system doesn't fail sometimes, it's not working hard enough!
- Scale supply voltage down when errors start to happen
- Add support for recovery from errors
  - Costing extra time/energy
- Set voltage at point where the EDP is minimum

Razor flip-flops





Razor pipelines

- Assume timing error occurs at L1 at cycle c
  - i.e. data not available on time
- Data at L2 at cycle c+1 is wrong
  - Must be squashed
  - Output of XOR gate flags this situation
- The correct data are in shadow latch
  - Can be given to L2 at c+2
- No need to redo the “faulty” operation at L1
- But must redo all operations at stages <= L1 from cycles c+1 onwards
  - Either clock gating or counterflow pipeline

Razor design issues

- Metastability
  - A stabilize stage is added before state commit
- Minimum delay requirement
  - New input to a stage cannot destroy old output before shadow latch captures it
  - May require adding some delay in some paths (consume energy)
- Which flops to razorise?
  - 192 out of 2408 in their prototype

Razor evaluation

- Baseline: alpha processor 0.18um tech
  - 425mW 200MHz, error free mode
  - 12.2mW overhead for delay elements (3.1%)
- Razor prototype (simulated)
  - Each benchmark has an optimal voltage level which minimises energy
  - Energy savings between 23.7 to 64.2%
  - 2.49% speed loss

Approximation circuits

- Replace a logic function with an approximation
- Errors are detected and recovered from
- Being simpler, approximation functions should be faster, low energy

Approximate adders

- Based on carry look ahead
- Instead of considering all inputs, look at fewer input bits (look-ahead k)
- Result is incorrect only if carry-chain longer than k
  - Easy to detect
- Delay of full carry chain for N bit adder is O(logN)
- If k is sqrt(N), delay will be halved
- E.g. 64-bit adder with k=8, correct 95% of time for random data

Approximation in other units

- Rename logic
  - Add a smaller CAM memory to implement the mapping table
  - Keep full-table for recovery (1 cycle penalty)
- Issue logic
  - Reduce the window used for waking up instructions that are ready
  - If N entried in the issue queue, only look at the top sqrt(N) entries for matches
  - No false results can happen, so no recovery needed

Evaluation

- Methodology:
  - Theoretical analysis and simple-scalar simulation
  - All machines have the same cycle time
  - Approximation takes 1 cycle, full operation takes 2 cycles
- Analysis showed good scope for speed-up
- Simulation results were worse
- Prediction accuracy:
  - Adder (32-b) 90% correct with 4-bit chains (less than sqrt)
  - Rename: 80%
  - Issue: 40%