EECS 151/251 A
Discussion 12

April 12, 2024

Berkeley
UNIVERSITY OF CALIFORNIA

Content

- Parallelism

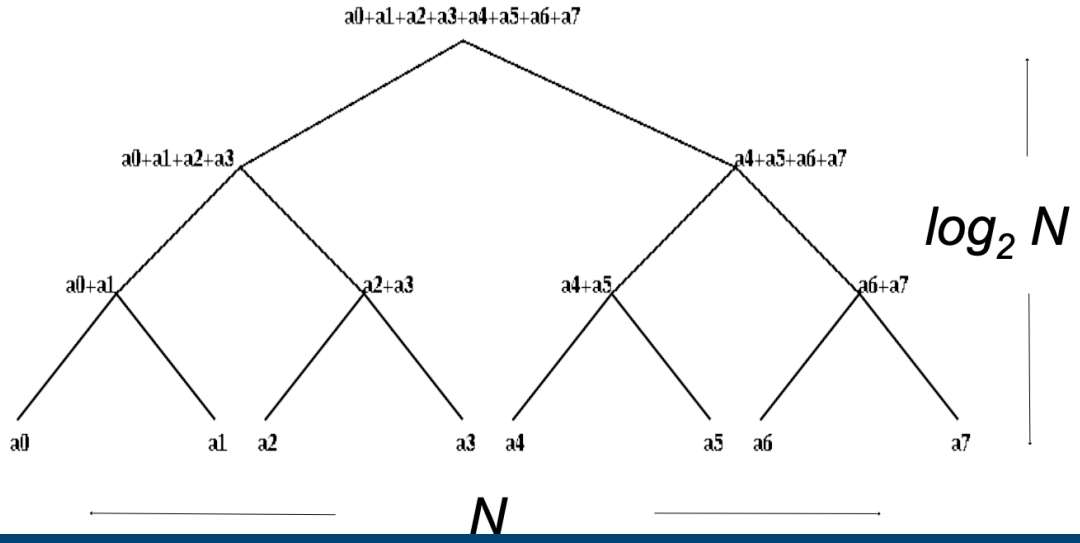
Parallelism

- Arguably the most important concept of hardware!
- Parallelism is currently how people get more performance. (think end of Moore's Law)
- Two main types of parallelism (often combined):
  - *Spatial* – Add more hardware and execute in parallel
  - *Temporal* – Exploit latency/time multiplex on the same amount of hardware

Parallelism is the act of ***doing more than one thing at a time****.*
Optimization in hardware design often involves using parallelism to trade between cost and performance.
Parallelism can often also be used to improve energy efficiency.

Parallelism – Spatial

- Trees are a common form of spatial parallelism
- In hardware reductions are perform on tree structure
  - Ex. adder reduction for dot product
- There are more fancy techniques to reduce total amount of hardware
- Maximize by allowing hardware to scale with task size



Parallelism – Temporal

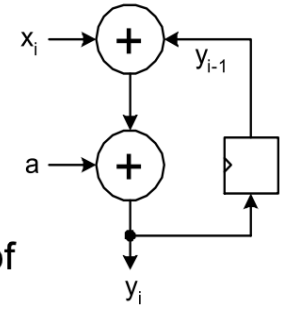
- **General concept**: exploit latency by allowing circuit to operate on a different inputs at different points of time
- Implementations
  - Pipelining
    - Each stage operates on a different input simultaneously
    - Pipelining is effective for streams of inputs
    - Examples: Multipliers, dividers, HW for image processing, processors, etc
  - Time multiplexing
    - A common circuit relies on some long latency input; switch operating on different chunks of work per period
    - Time multiplexing is useful for circuit with long latencies (ex. a circuit which must access DRAM)
    - Can be fixed, variable, event driven, etc
    - Examples: ALUs, processors, etc

Parallelism – Pipelining

- Pipelining Loops with Feedback
  - Pipelining modifies circuit timing. **Ensure correct functionality.**
  - Be careful pipelining circuits with feedback!

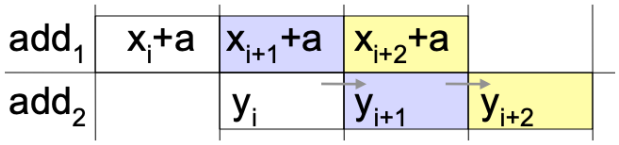
"Loop carry dependency"

However, we can overlap the "non-feedback" part of the iterations:

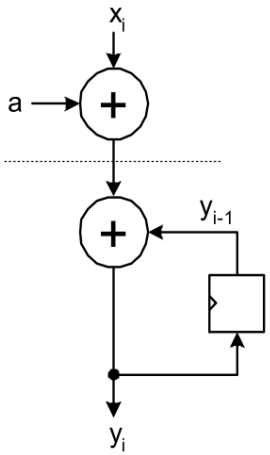


Add is associative and communitive. Therefore we can reorder the computation to shorten the delay of the feedback path:

$$y_i = (y_{i-1} + x_i) + a = (a + x_i) + y_{i-1}$$



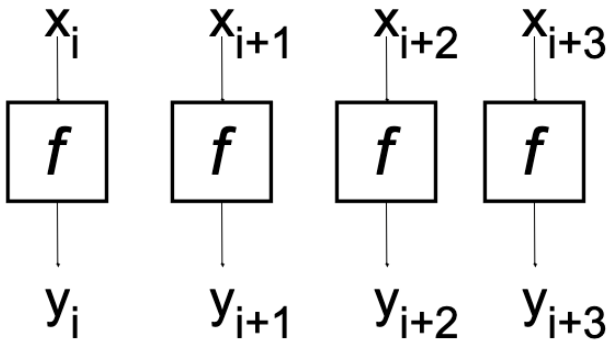
"Shorten" the feedback path.



- Pipelining is limited to 2 stages.

Parallelism – Loop Unrolling

- Works on multiple loop inputs at a time
- Combining Spatial and Temporal Parallelism
- Works best if there is no loop carry dependency
  - What happens if there is?



Parallelism – Summary

- **Key Idea:** Temporal vs spatial parallelism is a tradeoff
- Temporal parallelism saves hardware cost, but limit simultaneous operation if multiple inputs are available at the same time (i.e. decreased performance)
- Spatial parallelism increases the overall performance if multiple inputs are available at the same time, but require more hardware (i.e. more expensive)
- Most parallel hardware utilizes both to get the best of both worlds (i.e. a ML accelerator with hundreds of pipelined multipliers)
- A given target application naturally favors or the other

Question 1

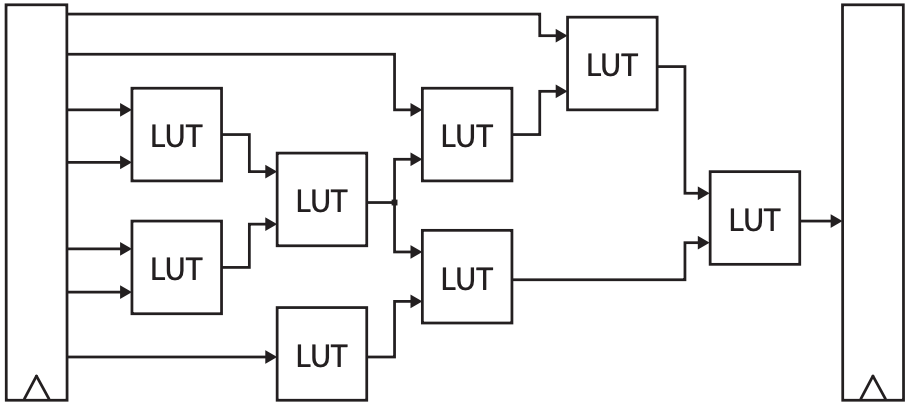
Problem 1: Pipelining for Speed [8 pts]

Given the circuit shown below with $\tau_{clk-q} = \tau_{setup} = 50ps$, and $\tau_{LUT} = 100ps$.

(a) Using pipelining, find the number of stages where the clock frequency is at least 2 times the original (unpipelined) clock frequency.

(b) Find the number of stages where the clock frequency is at least 3x the original.

Assume that LUTs are indivisible (cannot be internally pipelined). For each question above, draw dashed lines in the circuit diagram to indicate where to add registers.



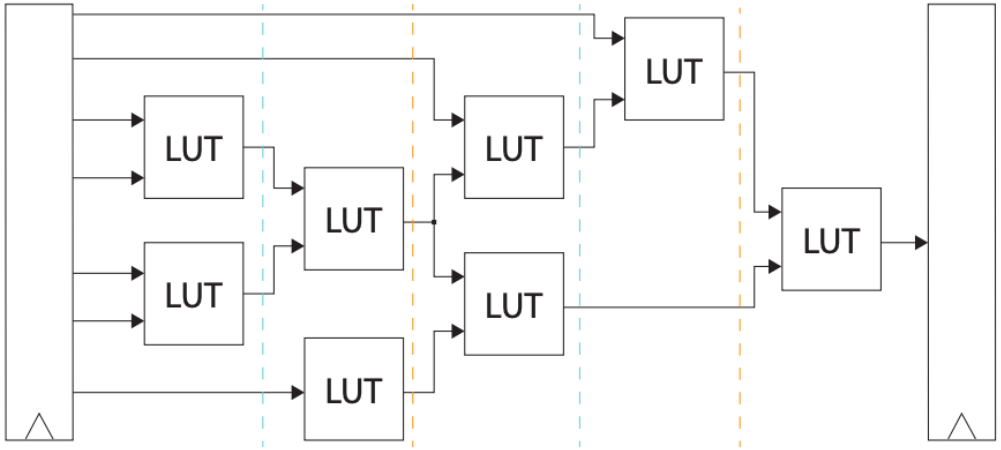
Solution

Solution:

The original critical path was $\tau_{clk-q} + 5\tau_{LUT} + \tau_{setup} = 50ps + 5(100ps) + 50ps = 600ps$.

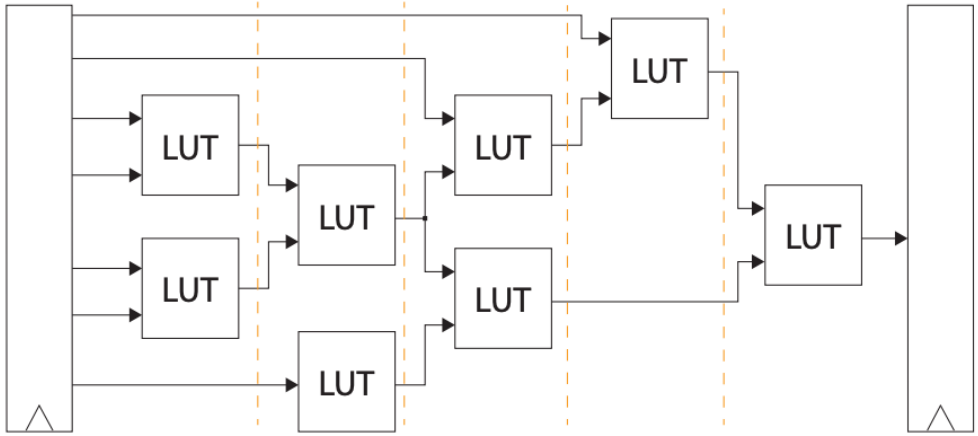
For pipelining to increase the clock frequency by at least 2x, the critical path must be reduced to be $\leq 300ps$. This can only be accomplished with a critical path of a maximum of 2 LUTs in series per between pipeline registers. *Remember that when pipeline, you need to make sure you match the delay on parallel paths. Otherwise, signals will not be aligned when arriving at the combinational logic blocks..*

Possible pipeline insertion points are shown below. Each color represents a possible solution.



Solution

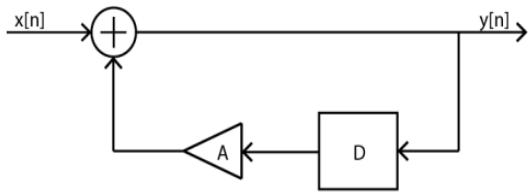
To increase the clock frequency by a factor of 3, the critical path must be reduce to be $\leq 200ps$. This can only be accomplished with a critical path of a single LUT between registers. In this case, there is only one possible pipelining solution which is shown below.



Question 2

Problem 2: Microarchitecture

In the circuit below, block D is a delay element (i.e. register) and block A performs multiplication by A.



(a) Write down the expression for $y[n]$.

(b) Unroll the loop such that each iteration covers two iterations of the original case. Substitute for $y[n-1]$ using the expression from (a) and then draw the resulting block diagram.

(c) Try to pipeline your design from part b.

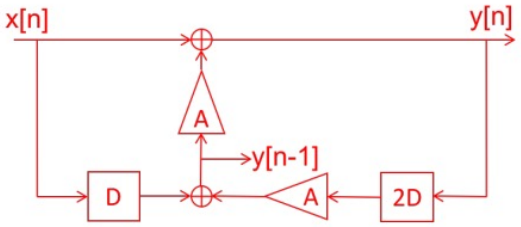
Solution

(a) Write down the expression for $y[n]$.

$y[n] = x[n] + A \cdot y[n-1]$ $\qquad$ $y[n-1] = x[n-1] + A \cdot y[n-2]$

(b) Unroll the loop such that each iteration covers two iterations of the original case. Substitute for $y[n-1]$ using the expression from (a) and then draw the resulting block diagram.

$y[n] = x[n] + A(x[n-1] + Ay[n-2])$

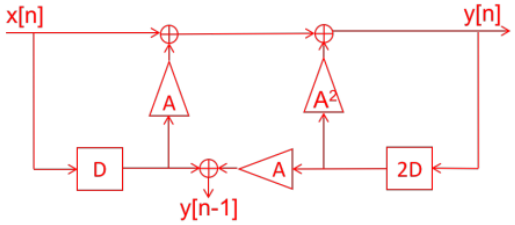


Solution

(c) Try to pipeline your design from part b.

Using distributivity and associativity:

$y[n] = x[n] + Ax[n-1] + A \cdot A \cdot y[n-2] = (x[n] + Ax[n-1]) + (A^2 y[n-2])$



Now that we have two separate loops, a pipeline register can be added in between the two adders: