Implementation Security In Cryptography

Lecture 05: Some Essential Concepts in Hardware Design

Recap

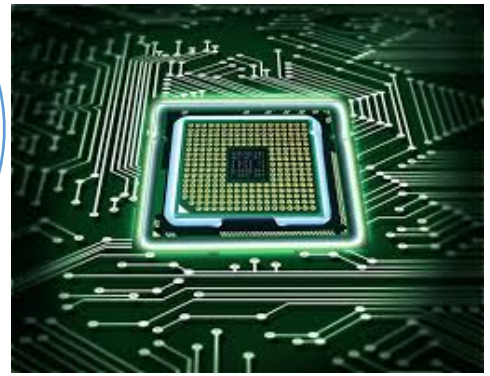
- In the last lecture
  - Introduction to block ciphers
  - Simple block cipher PRESENT
  - Introduction to Verilog

Today

- Some Essentials of Hardware Design

Hardware Design on ASICs and FPGAs

Performance:
Speed, Clock
Frequency, Latency
Fast Arithmetic



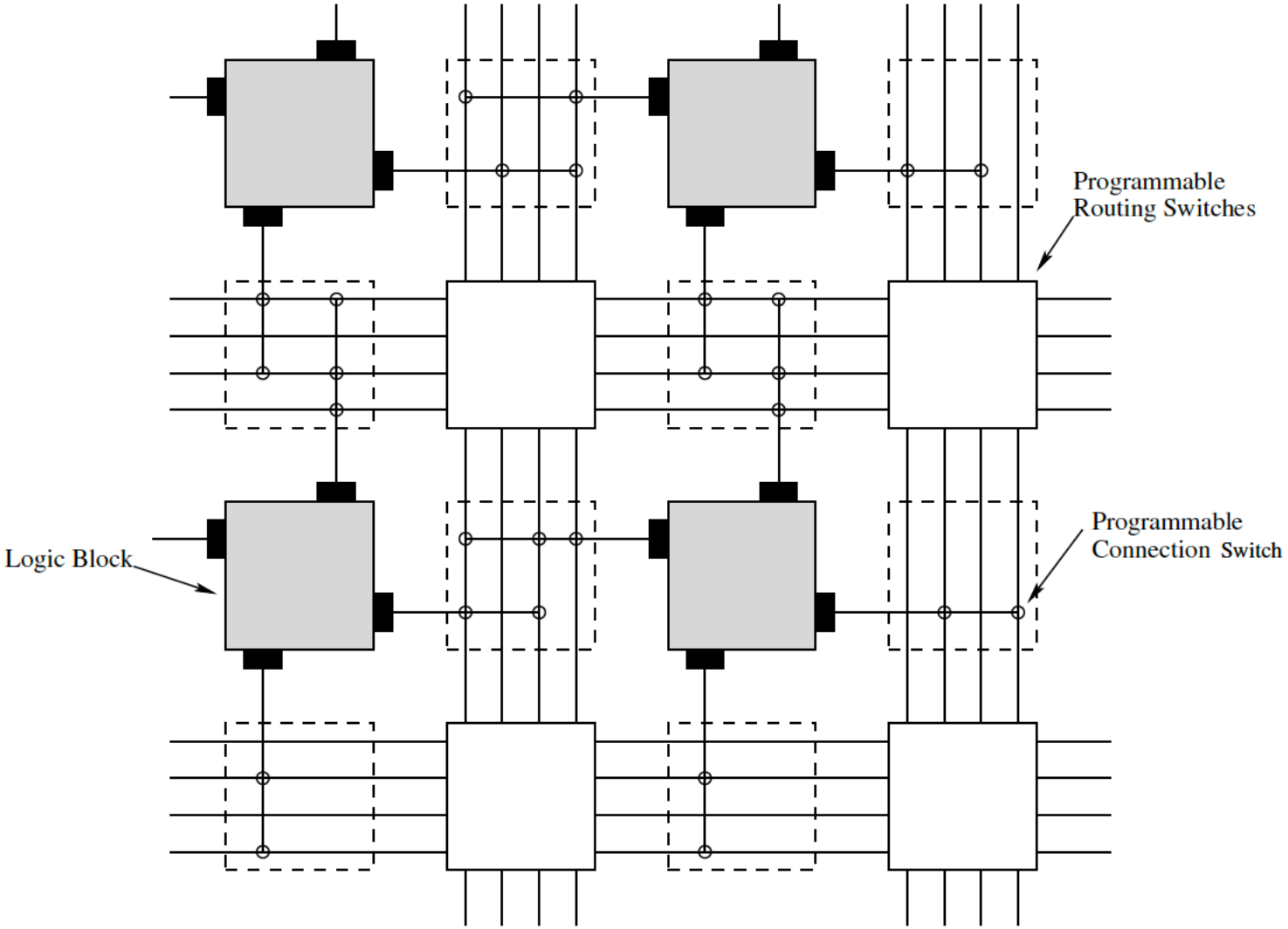
Security:
Side Channel Attacks,
Fault Attacks
Lightweight
Countermeasures

Cost:
Area, Power,
Energy

Hardware Design on ASICs and FPGAs

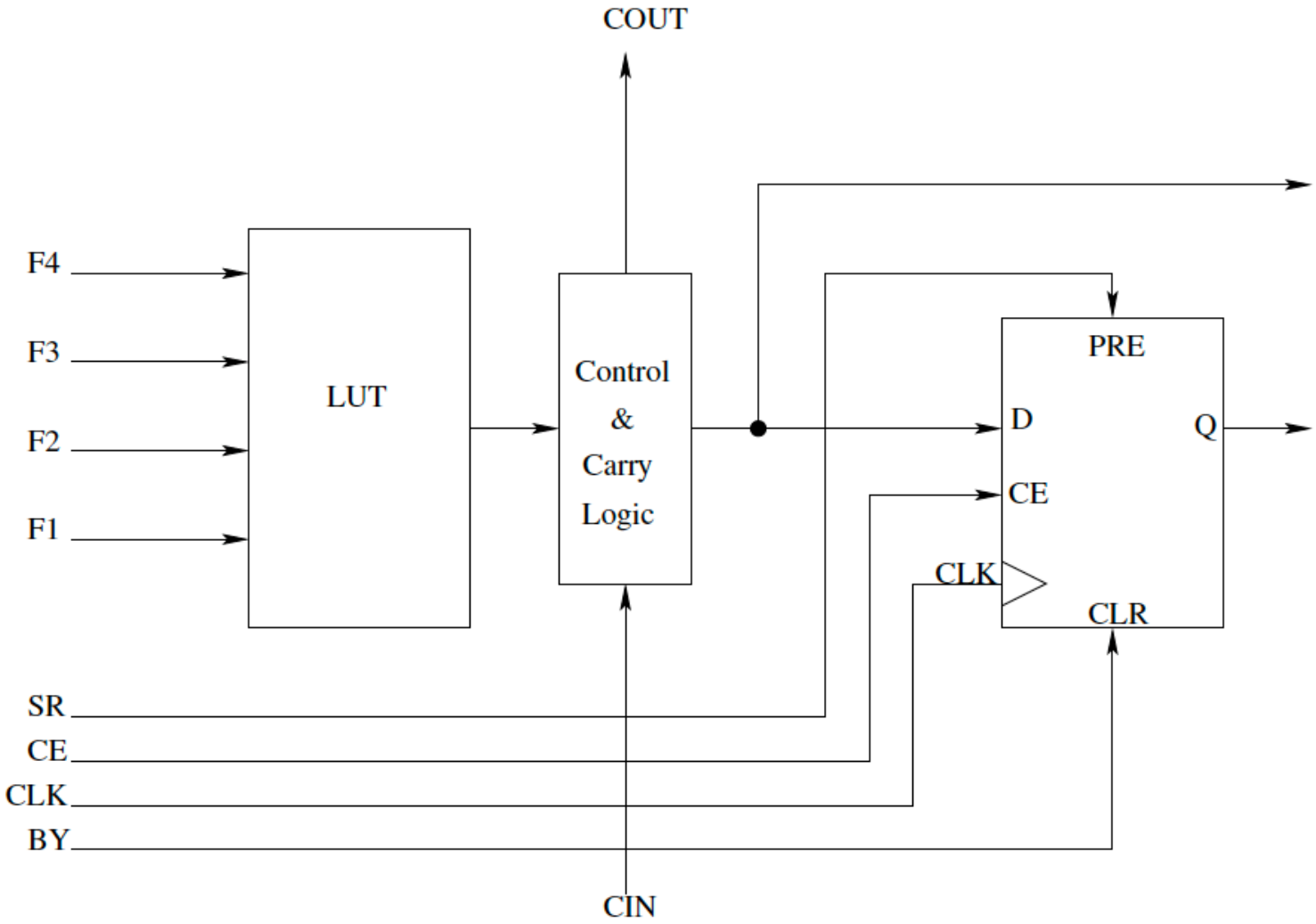
- Let is talk on the Area
  - For ASIC, a rough measurement is by number of gates/FFs.
  - For FPGA it is a bit tricky
    - Gate count indeed matters, but we have LUTs.

Hardware Design on ASICs and FPGAs



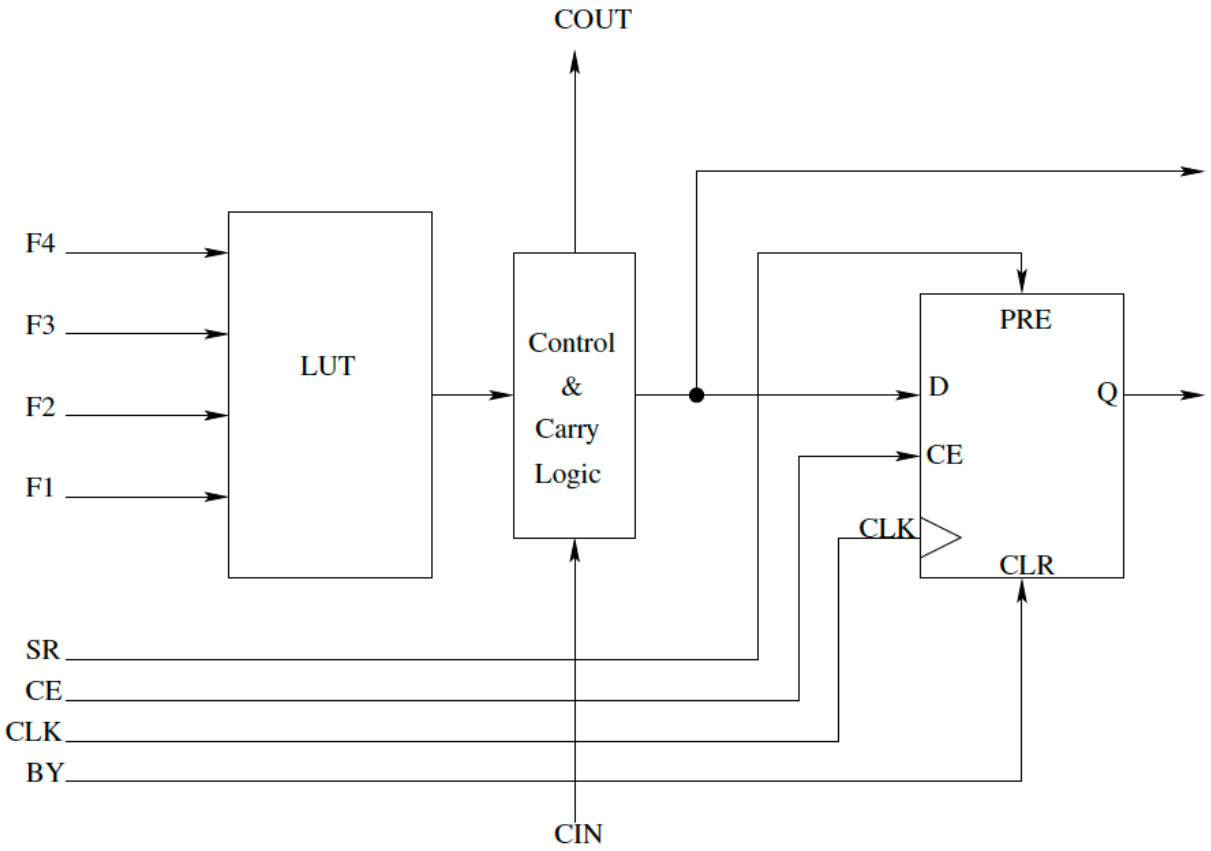
- Configurable Logic Blocks (CLBs): Main programming unit
- Routing switches and connection switches: Creates connection between the CLBs
- Each CLB contains a group of Basic Logic Elements (BLEs)
- Such BLEs are grouped into Slices

FPGA Architecture — Basic Logic Element



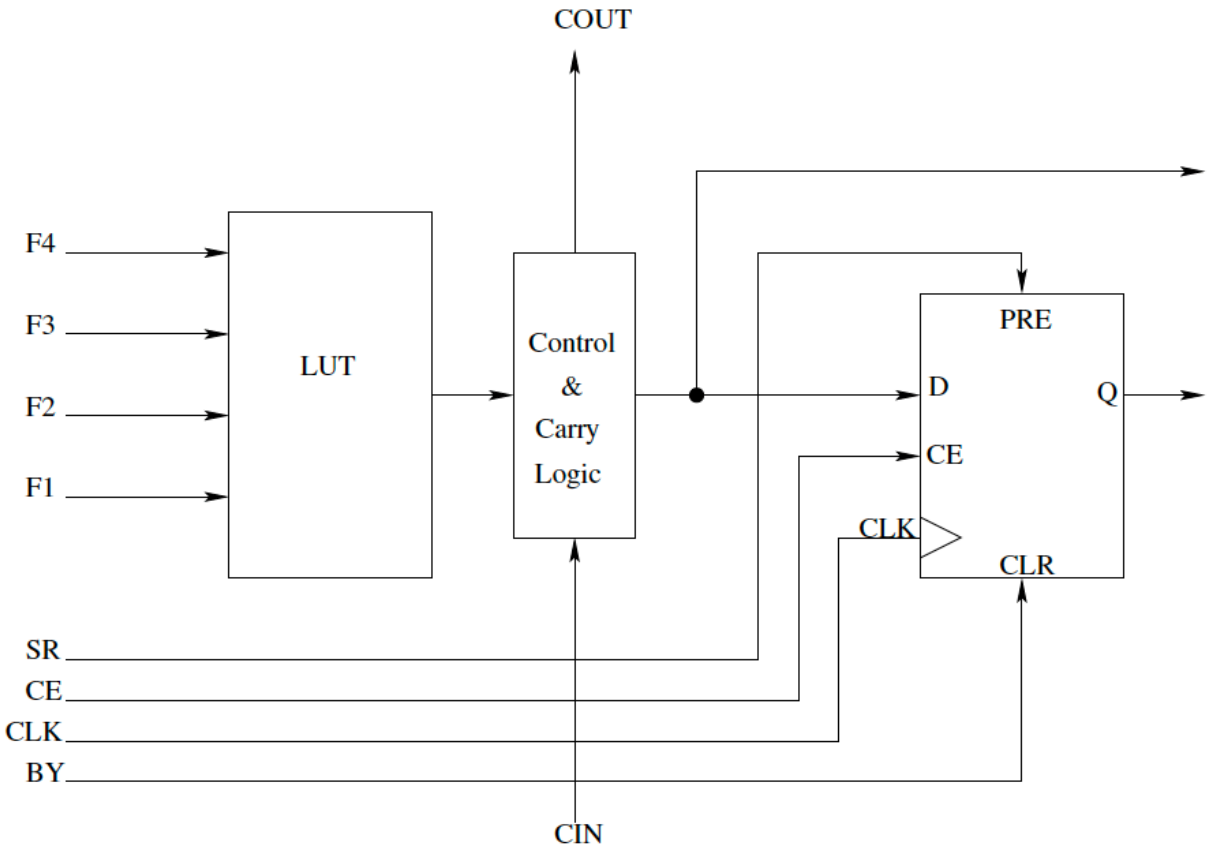
- Each slice contain multiple such BLEs
- Example: One CLB of Xilinx Virtex-5 FPGA contains two slices. Each slice contains 4 LUTs (lookup tables), 4 flip-flops, multiplexers and carry chains.

FPGA Architecture — Lookup Table



- <u>The “gate” of FPGA</u>
- LUTs store small truth tables
- m-input LUT can realise m-input Boolean function.
- These are “programmable”..
- It has some memory for storing truth tables and multiplexors for input/ output selection
- What is the best size for an LUT?

FPGA Architecture — Lookup Table



- Very large LUT — more logic can fit in, so lesser path delay,
- But larger memory for storing a large truth table and larger MUXes. — more area
- Study shows, 4 to 6 input LUTs give the best area delay product.
- Given an FPGA, it has either 4 or 6 input LUTs only

Some Important Concepts

- **Register to Register Delay**: Propagation delay of a signal through the combinational path between two registers
- **Critical Path**: Longest delay path between two registers.
- Length of critical path decides the clock frequency

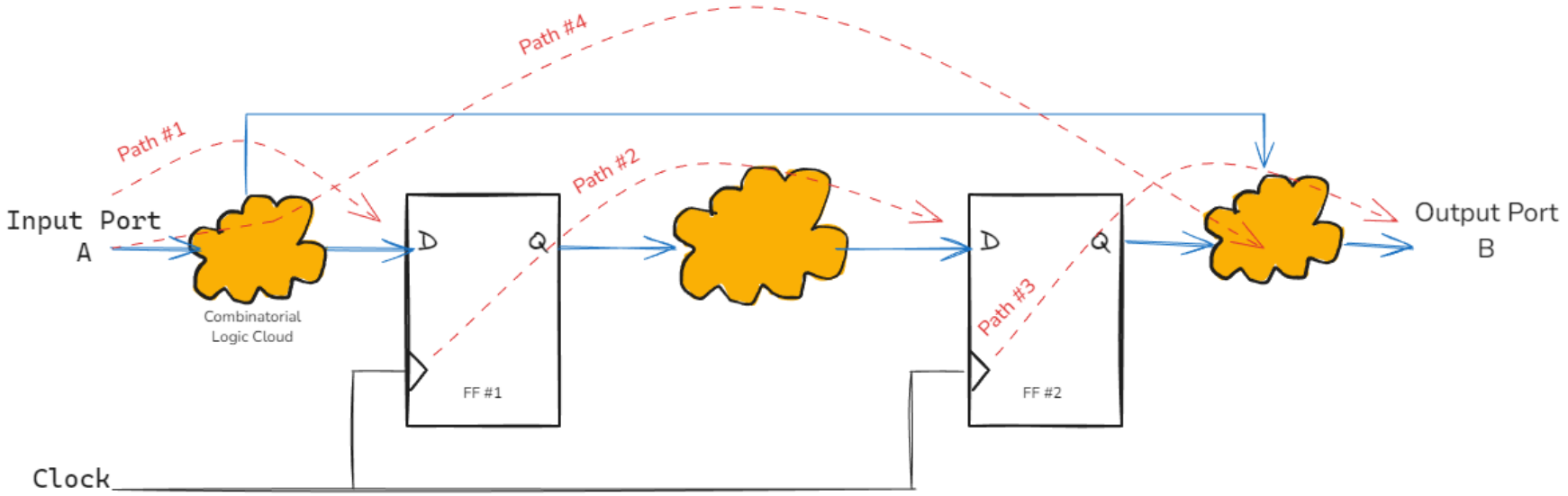


Image source: internet

Some Important Concepts

- **Setup-time**: It is the minimum time that the synchronous data must arrive before the active clock edge in a sequential circuit.
- **Hold-time**: It is the minimum time that the synchronous data should be stable after the active clock edge.

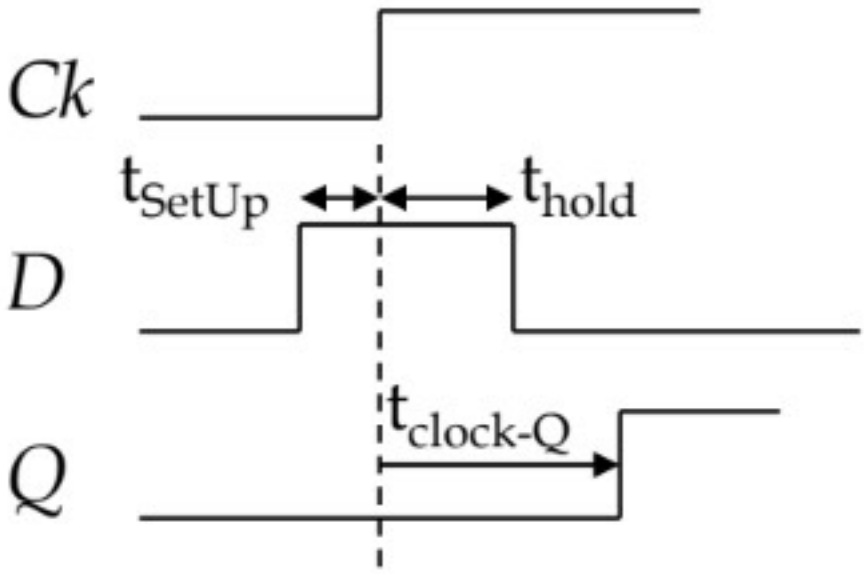


Image source: internet

How to Understand the Resource Utilization on FPGAs

- LUT count, number of slices, number of FFs etc.
- Let us assume that we have 4-input LUTs
  - A function may have only 2 inputs (or say 3)
  - Then, if it maps to an LUT, the LUT will be *under utilized*
- *The synthesis tools try to minimize the number of under-utilized LUTs*
- But it is always good to have a rough idea of what the tool can do..

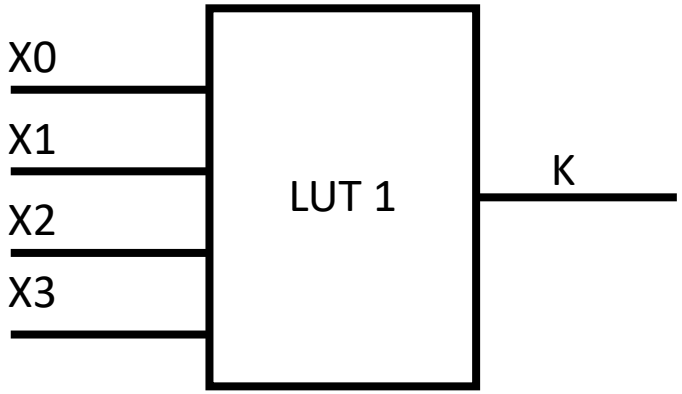
How to Understand the Resource Utilization on FPGAs

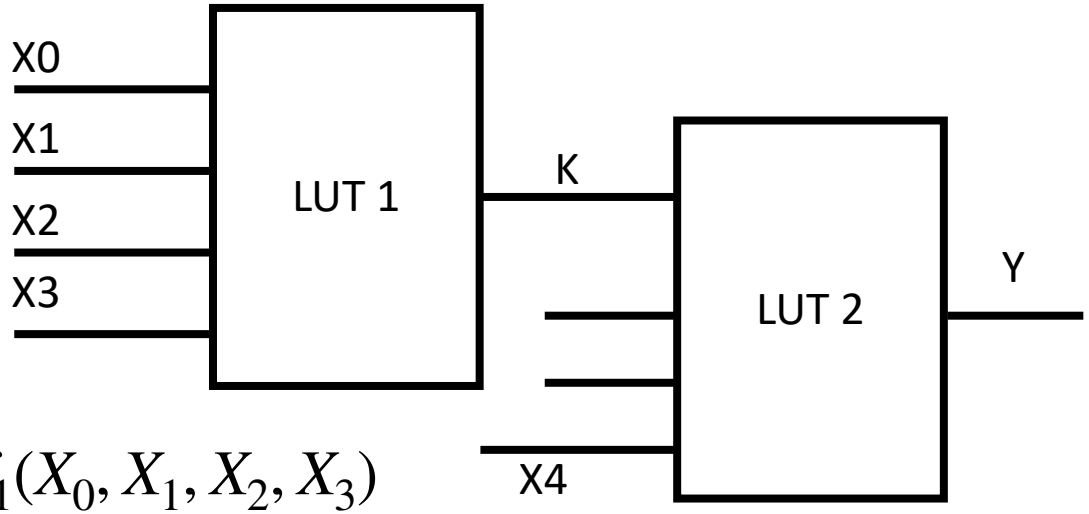
- A good hardware implementation requires that the number of LUTs are minimized and the LUTs are fully utilized.
- We can actually estimate how good is our LUT utilization for a given function.
  - Let's see how..
- Suppose, I have a *q* input function:
  - If $q = 1$, I need 0 LUT
  - If $q \leq 4$, I need 1 LUT.
  - If $q > 4, \; and \; q \; mod \; 3 = 2$, we need $\lceil q/3 \rceil$ LUTs
  - If $q > 4, \; and \; q \; mod \; 3 \neq 2$, we need $\lfloor q/3 \rfloor$ LUTs

How to Understand the Resource Utilization on FPGAs

- Suppose, I have a *q* input function:
  - If $q = 1$, I need 0 LUT
  - If $q \leq 4$, I need 1 LUT.
  - If $q > 4, \; and \; q \; mod \; 3 = 2$, we need $\lceil q/3 \rceil$ LUTs
  - If $q > 4, \; and \; q \; mod \; 3 \neq 2$, we need $\lfloor q/3 \rfloor$ LUTs
- This is (roughly) the minimum number of LUTs — not a perfect estimate.
- But it works for an early assessment of a design.

How to Understand the Resource Utilization on FPGAs





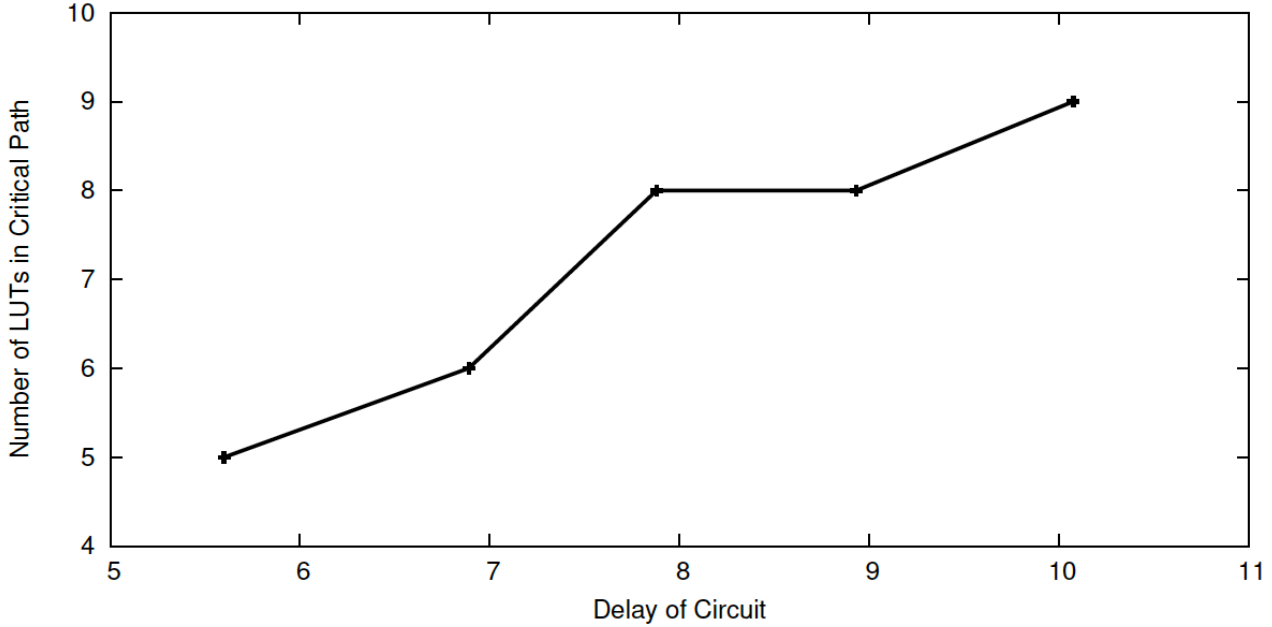
$$K = f_1(X_0, X_1, X_2, X_3)$$

$$Y = f_2(K, X_4)$$

- Key Idea: We can assign 4 inputs to the first LUT. Then for each kth LUT we can assign at most 3 new inputs and one coming from the (k-1) th LUT. That is why we divide with 3.

LUTs in Critical Path to Measure Delay

Delay in FPGAs comprise of both LUT and routing delay, and thus is more complex to analyze.
Through experiments one can however see that for combinational circuits, the delay varies linearly with the number of LUTs in the critical path.



Linear relationship between number of LUTs in Critical Path and Delay of a Combinational Multiplier of increasing dimensions.

References: Hardware Security: Design, Threats, and Safeguards, CRC Press.

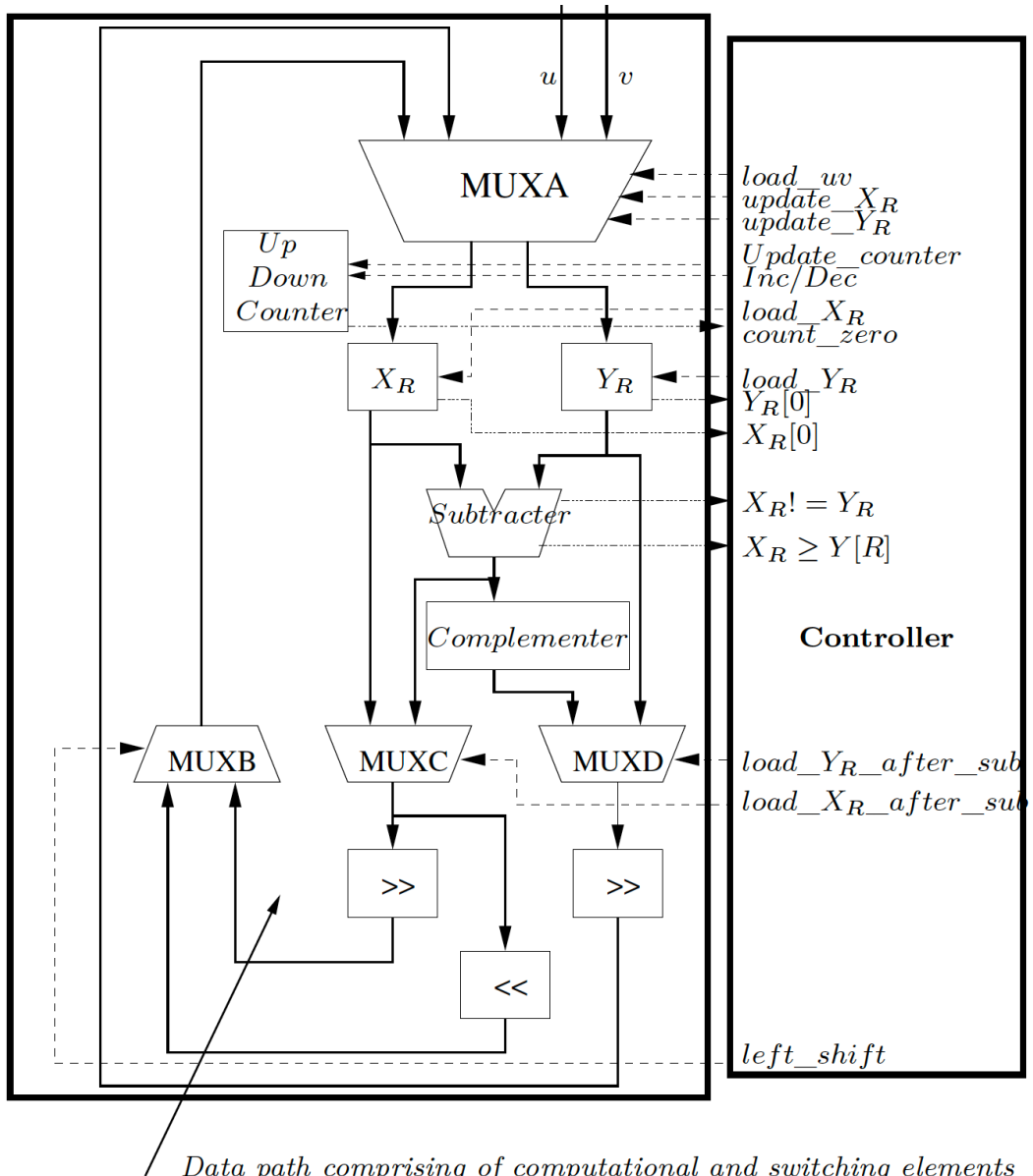
Delay in FPGAs

- Delay of a q input function is basically the number of LUTs in the critical path.
- The minimum possible estimate is:

$$Delay(q) = \lceil \log_4(q) \rceil \times D_{LUT}$$

- Observe that this does not exactly match with the minimum possible estimate of LUT count.
  - Delay calculation assumed logarithmic number of LUTs on the critical path
  - LUT count estimate is linear
  - But both are basically minimum estimates — so actual values will be somewhat in between

Modeling the Components of the gcd processor



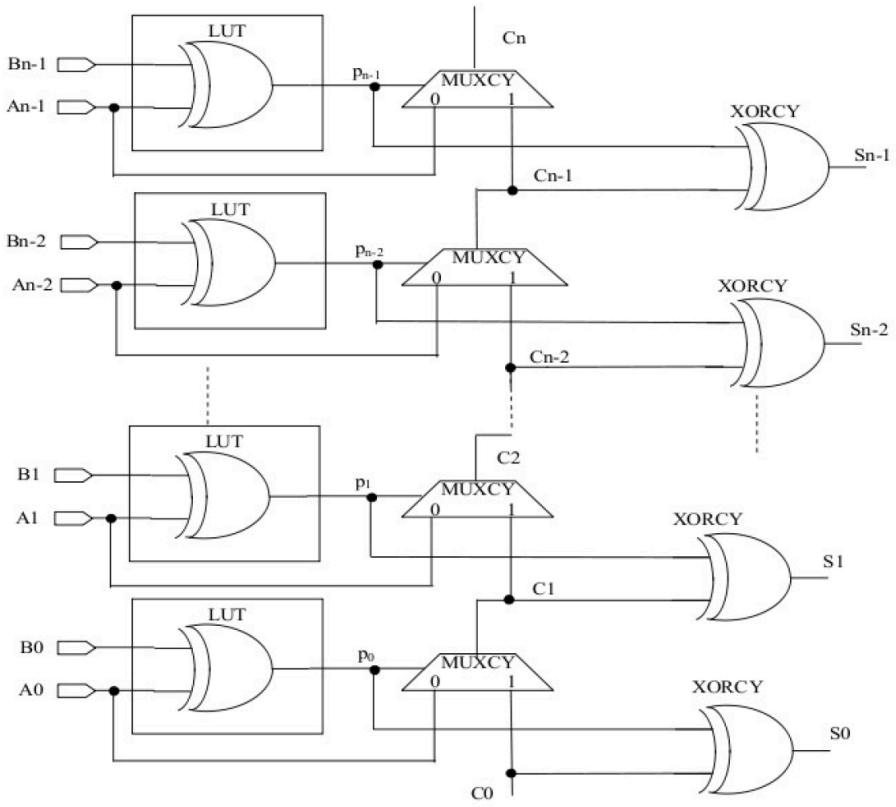
- Let us consider the datapath of a GCD processor.
- Details on how to design this processor can be found at: https://www.youtube.com/watch?v=sACVot8QFWY&list=PLQBbcgo55TX-7vygatpMHOOtgflvYtkeZ&index=3
- We are only interested in the delay estimation of the datapath
- Let us assume that the gcd circuit handles m -bit variables
- So each of the components process m bit values

Modeling the Components of the gcd processor

Most important component in the data-path of the gcd processor is subtractor, which can be realized by an adder.

On FPGA platforms, carry chain based adder are specially optimized and are fast.

For Xilinx Virtex IV FPGAs, s is almost 17.



The carry chains use m-MUXCY for m-bit adder.
They are much faster compared to the LUTs which are used for other parts of the circuit.
So, in order to compare we scale the delay, and state:

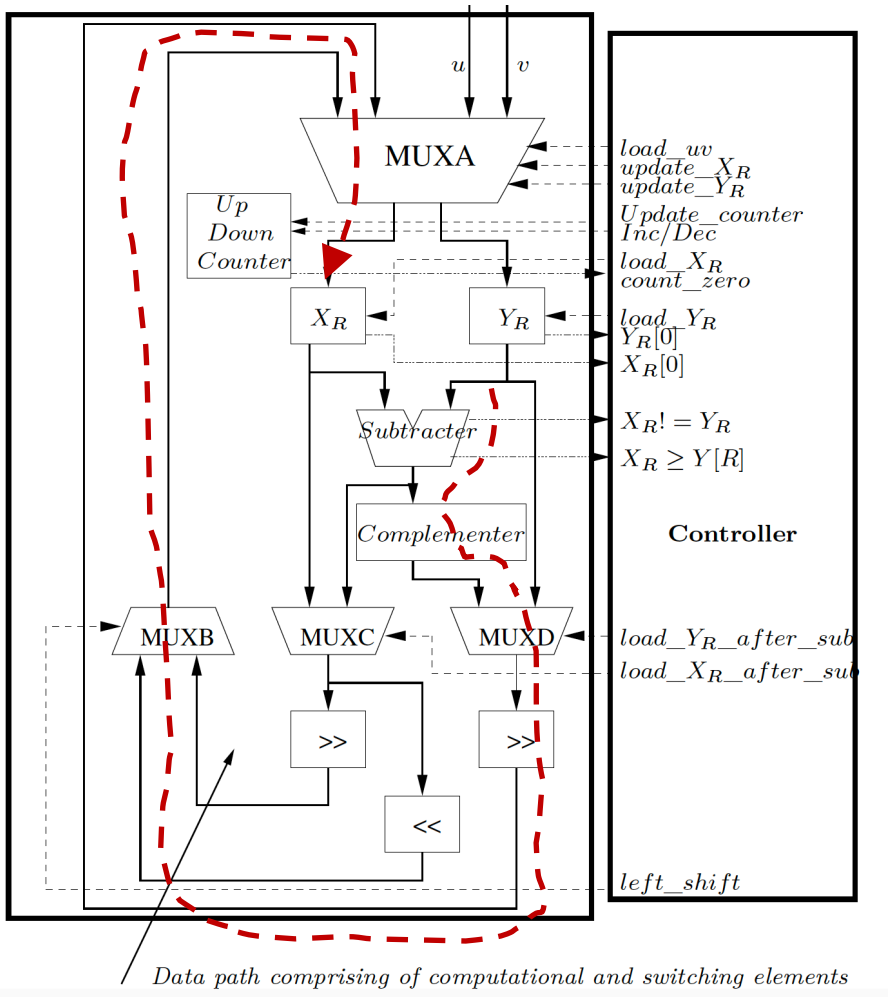
$$D_{add}(or\ D_{sub}) = \lceil m/s \rceil$$

References: Hardware Security: Design, Threats, and Safeguards, CRC Press.

Modeling the Multiplexer

- 2^t:1 Multiplexer: Output is a Boolean function of (2^t+t) variables.
- So, for each bit output, it requires to implement a (2^t+t) bit function. Say that needs lut(2^t+t) LUTs.
- For, an m-bit gcd multiplier, which requires to pass m bit values in the datapath, hence no. of LUTs is m* lut(2^t+t)
- Delay: DMUX= $\lceil \log_4(2^t + t) \rceil$
- It has been experimentally verified that these rough estimates are close to reality.
- For example, if t=2, we need 2 LUTs. You may check it out..

Total Delay Estimate of the gcd processor



- Critical path of the design:

subtractor→complementer→MUXD→MUXB →MUXA.

$$D_{\text{gcd}} = 2D_{sub} + D_{MUXD} + D_{MUXB} + D_{MUXA}$$
$$= 2\lceil m/s \rceil + 1 + 1 + 1 = 3 + 2\lceil m/s \rceil$$

Note that the delay of MUXA comes from the fact that the multiplexer is made of 2 smaller 2-input multiplexers in parallel: one input writing to XR, while the other writing to YR.

Total LUT Estimate of the gcd processor

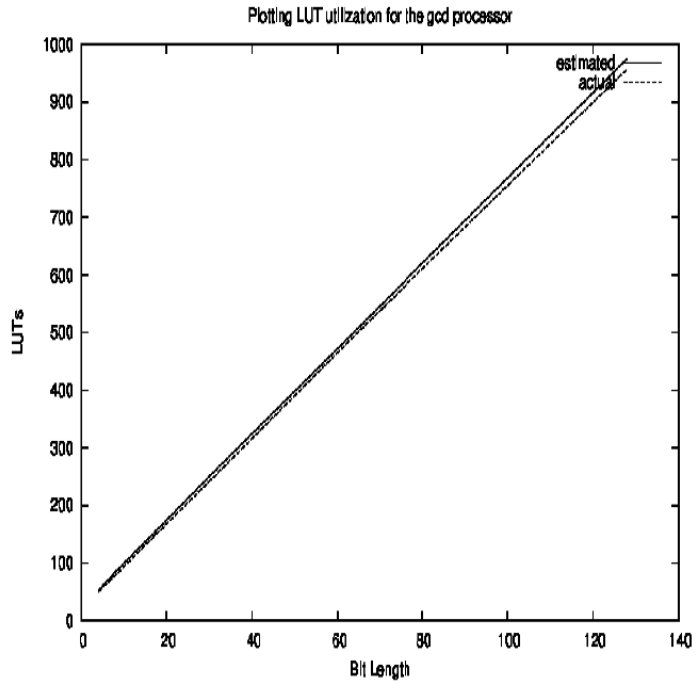
- Sum of the LUTs for MUXA, MUXB, MUXC, MUXD, and the subtractor along with the complementer (which is another subtractor).
- The state machine is assumed to consume very less LUTs and is ignored.
- Total number of 4-LUTs in the entire circuit:

$$\#\,LUT_{\text{gcd}} = 2\,LUT_{subtractor} + LUT_{MUXA} + LUT_{MUXB} + LUT_{MUXC} + LUT_{MUXD}$$

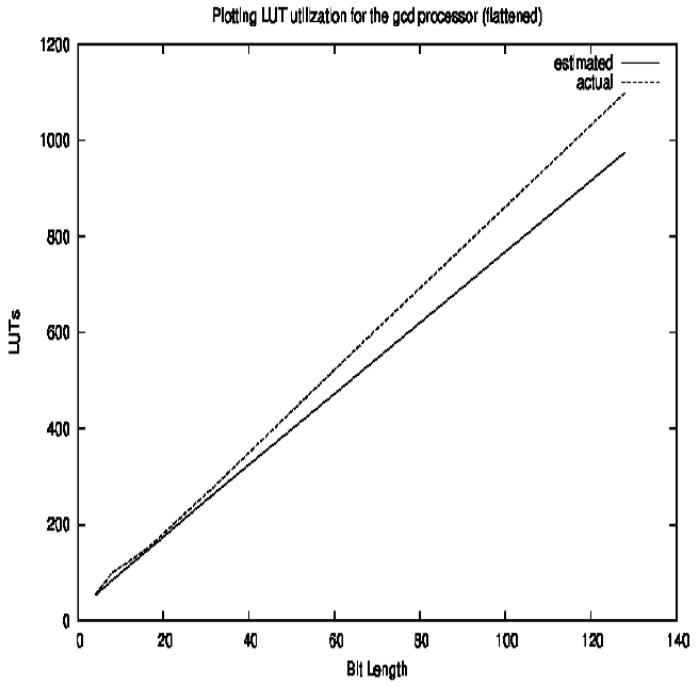
Experimental Exploration

- The above design was synthesized using Xilinx ISE tools and targeted on a Virtex-4 FPGA.
- Objective of the experiment was to study the above dependence on the LUT utilization and to estimate the critical path delay for the circuit.
- The objective of the exploration is to see the trend, and not the exact figures.
- We vary the bit length of the gcd processor, and repeat the experiments to study the scalability of the design.
- This helps in design exploration, as we are able to understand the dependence and tweak the design early in the design cycle.

LUT utilization

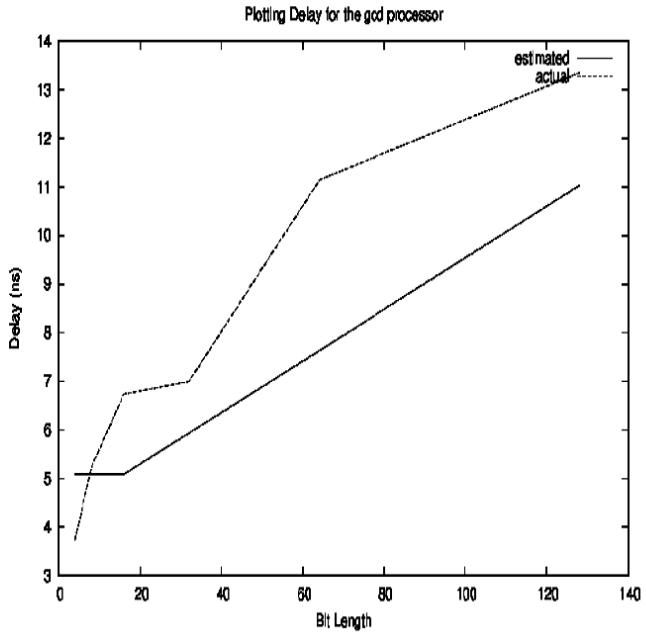


(a) LUT utilization of the gcd processor (hierarchy on)

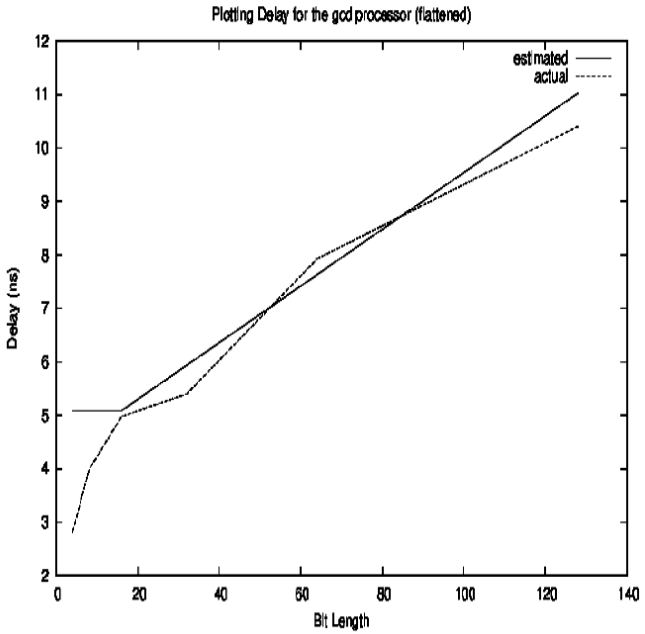


(b) LUT utilization of the gcd processor (hierarchy flattened)

Delay Estimation



(a) Critical Path Delay of the gcd processor (hierarchy on)



(b) Critical Path Delay of the gcd processor (hierarchy flattened)