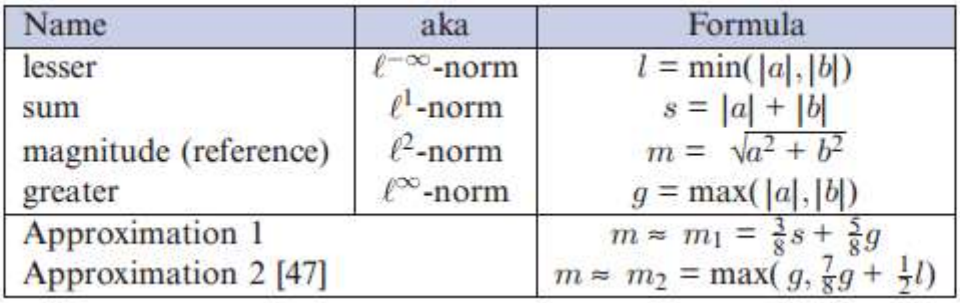
# Alexandria University
## Faculty of Engineering
***Electrical Engineering Department***

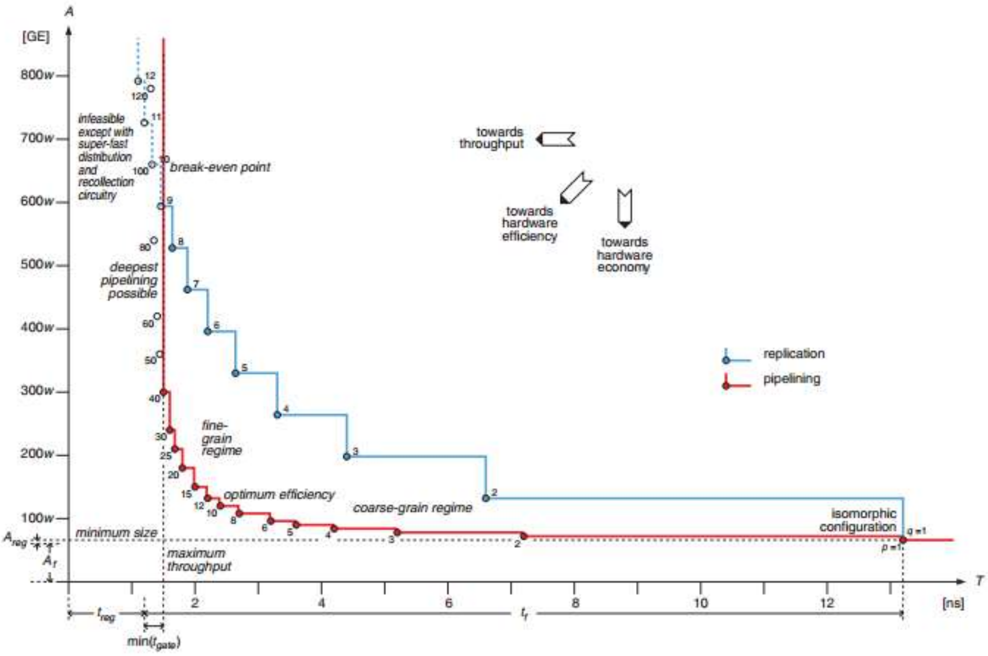
## EE432: VLSI Modeling and Design
## Sheet 4 : From Algorithm to Architecture

1. Computationally efficient approximations for the magnitude function $(a^2+b^2)^{0.5}$ have been presented in table 3.8.
   a. Show that approximation 2 remains within ±3% of the correct result for any values of a and b.
   b. Give three alternative architectures that implement the algorithm and compare them in terms of datapath resources, cycles per data item, longest path, and control overhead. Assume input data remain valid as long as you need them, but plan for a registered output. Begin by drawing the DDG.

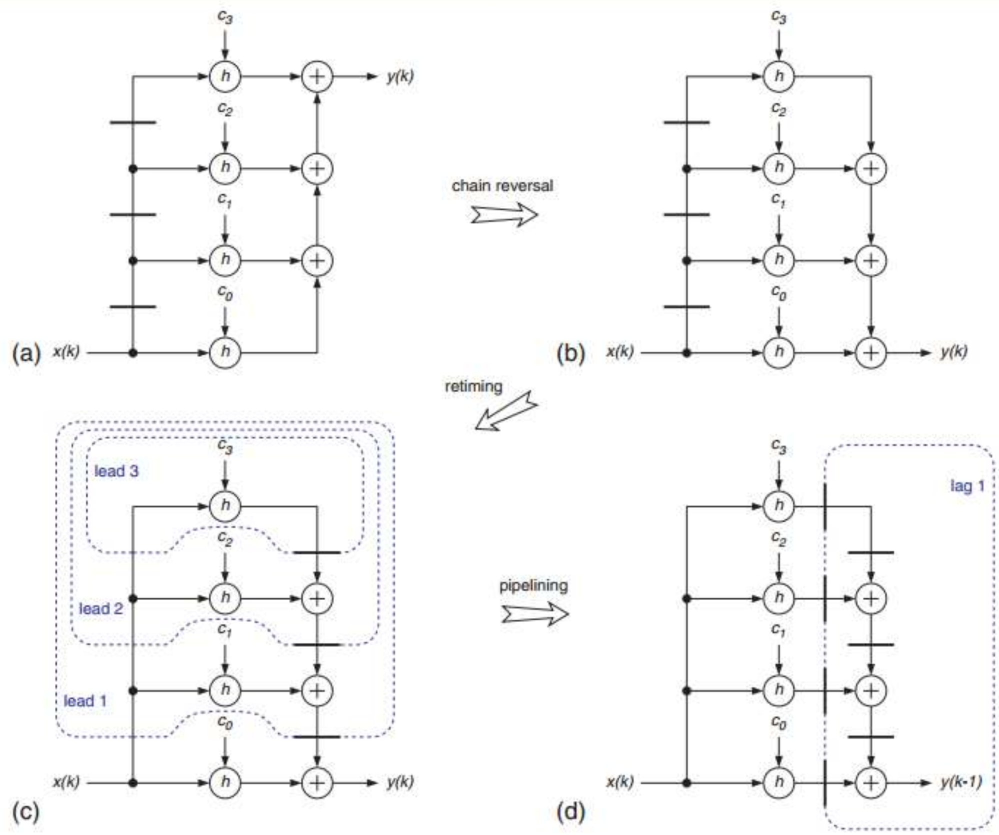


| Name | aka | Formula |
|---|---|---|
| lesser | $\ell^{-\infty}$-norm | $l = \min(\lvert a\rvert, \lvert b\rvert)$ |
| sum | $\ell^{1}$-norm | $s = \lvert a\rvert + \lvert b\rvert$ |
| magnitude (reference) | $\ell^{2}$-norm | $m = \sqrt{a^2 + b^2}$ |
| greater | $\ell^{\infty}$-norm | $g = \max(\lvert a\rvert, \lvert b\rvert)$ |
| Approximation 1 | | $m \approx m_1 = \frac{3}{8}s + \frac{5}{8}g$ |
| Approximation 2 [47] | | $m \approx m_2 = \max(g, \frac{7}{8}g + \frac{1}{2}l)$ |

2. Discuss the idea of combining replication with pipelining. Using fig.3.25 and the numbers that come along with it as a reference, take a pipelined datapath before duplicating it. Sketch the result in the AT-plane for various pipeline depths, e.g. for p=2, 3, 4, 5, 6, 8, 10. Compare the results with those of competing architectures that achieve similar performance figures (a)by replicating the isomorphic configuration and (b) by extending the pipeline approach beyond the most efficient depth. How realistic are the various throughput figures when data distribution/recollection is to be implemented using the same technology and cell library as the datapath?



3. Reconsider the third order correlator offig.3.32a.
    a. To boost performance, try to retime and pipeline the isomorphic architecture without prior reversal of the adder chain. How does the circuit so obtained compare with fig.3.32d. Give estimates for datapath resources, cycles per data item, longest path, latency, and control overhead?
    b. Next assume your prime concern is area occupation. What architectures qualify?



4. Fig.3.33 shows a viable architecture for a transversal filter. Before this architecture can be coded using an HDL, one must work out the missing details about clocking, register clear, register enable, and multiplexed control signals. Establish a schedule that lists clock cycle by clock cycle what data items the various computational units are supposed to work on, what data items or states that the various registers are supposed to hold, and what logic values the various control signals must assume to marshal the interplay of all those hardware items. Samples are to be processed as specified by fig.3.17a.

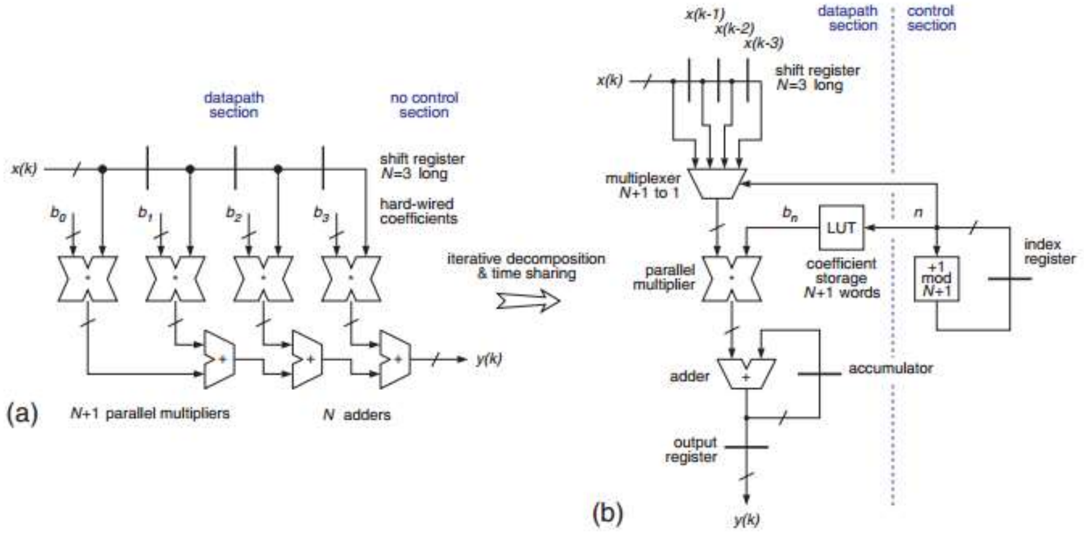


FIGURE 3.33

$$y(k) = \sum_{n=0}^{N=3} b_n\, x(k-n)$$

(a)

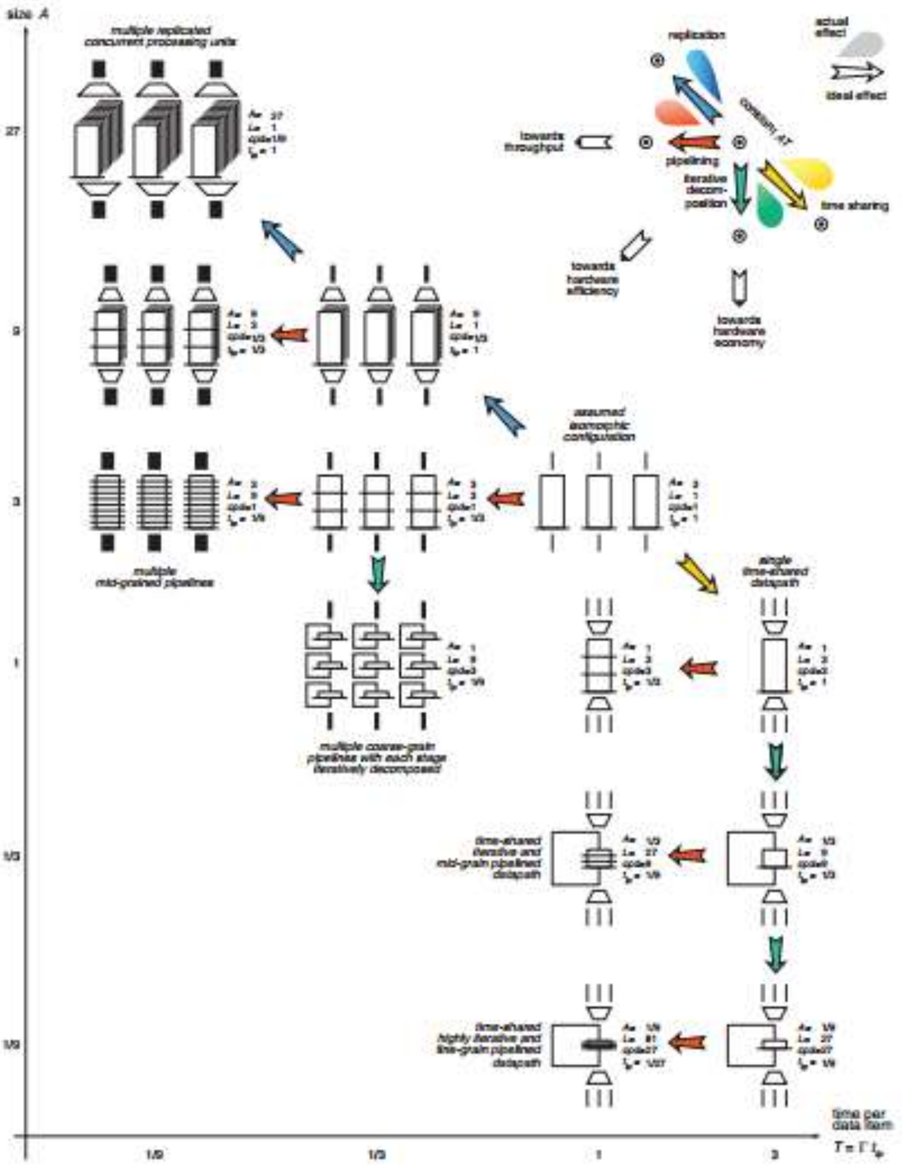
5. Arithmetic mean $\bar{x}$ and standard deviation σ are defined as

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad\qquad \sigma^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \bar{x})^2$$

Assume samples $x_n$ arrive sequentially one at a time. More specifically, each clock cycle sees a new w-bit data item appear. Find a dedicated architecture that computes $\bar{x}$ and $\sigma^2$ after N clock cycles and where N is some integer power of two, say 32. Definitions in (3.98) suggest one needs to store up toN−1 past values of *x*. Can you do with less? What mathematical properties do you call on? What is the impact on datapath word width? This is an old problem the solution of which has been made popular by early scientific pocket calculators such as the HP-45, for instance. Yet, it nicely shows the difference between a crude and a more elaborate way of organizing a computation.

6. Most locations in the map offig.3.28can be reached from the isomorphic configuration on more than one route. Consider the location where A=1/3 and T=1, for instance. Possible routes include
   a. (time share → decompose → pipeline) as shown on the map,
   b. (time share → pipeline → decompose),
   c. (pipeline → decompose → time share),
   d. (pipeline → time share →decompose), and
   e. (decompose → decompose).

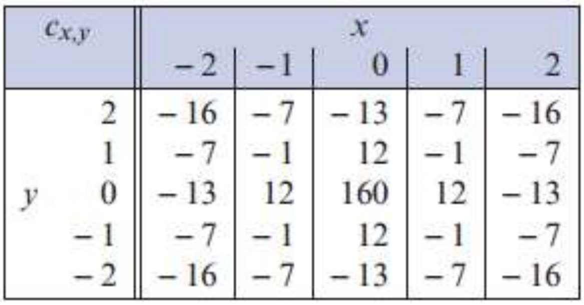
   Architectures obtained when following distinct routes typically differ. Fig.3.28indicates only one possible outcome per location and is, therefore, incomplete. Adding the missing routes and datapath configurations is left as a pastime to the reader. Purely out of academic interest, you may want to find out which transforms form commutative pairs.


size A
multiple replicated concurrent processing units
replication
actual effect
ideal effect
constant AT
towards throughput
pipelining
iterative decom-position
time sharing
towards hardware efficiency
towards hardware economy
assumed isomorphic configuration
multiple mid-grained pipelines
single time-shared datapath
multiple coarse-grain pipelines with each stage iteratively decomposed
time-shared iterative and mid-grain pipelined datapath
time-shared highly iterative and fine-grain pipelined datapath
27
9
3
1
1/3
1/9
1/9
1/3
1
3
time per data item
T = Γ t_lp

7. Fig.3.28 shows a kind of compass that expresses the respective impact of iterative decomposition, pipelining, replication, and time sharing. Include the impact of the associativity transform in a similar way.
8. Calculating the convolution of a two-dimensional array with a fixed two-dimensional operator is a frequent problem from image processing. The operator $c_{x,y}$ is moved over the entire original image *p(x,y)* and centered over one pixel after the other. For each position *X,Y* the pertaining pixel of the convoluted image *q(x,y)* is obtained from evaluating the inner product

$$q(X,Y) = \sum_{y=-w}^{+w} \sum_{x=-w}^{+w} c_{x,y}\, p(X+x, Y+y)$$

   Consider an application where w=2. All pixels that contribute to (3.99) are then confined to a 5-by-5 square with the current location in its center. An uninspired implementation with distributed arithmetics would thus call for a lookup table with 225 entries which is exorbitant. A case study by an FPGA manufacturer explains how it is possible to cut this requirement down to one lookup table of a mere 16 words. Clearly, this remarkable achievement requires a couple of extra adders and flip-flops and is dependent on the particular set of coefficients given below. It combines putting together multiple occurrences of identical weights, splitting of the lookup table, taking advantage of nonoverlapping1s across two coefficients, and clever usage of the carry input for the handling unit weights. Try to reconstruct the architecture. How close can you come to the manufacturer's result published in [85]?



| $c_{x,y}$ | | $x$ | | | | |
|---|---|---|---|---|---|---|
| | | − 2 | − 1 | 0 | 1 | 2 |
| | 2 | − 16 | − 7 | − 13 | − 7 | − 16 |
| | 1 | − 7 | − 1 | 12 | − 1 | − 7 |
| $y$ | 0 | − 13 | 12 | 160 | 12 | − 13 |
| | − 1 | − 7 | − 1 | 12 | − 1 | − 7 |
| | − 2 | − 16 | − 7 | − 13 | − 7 | − 16 |

9. It has been claimed in section 3.9.2 that dedicated architectures can carry out computations with a small fraction of the energy that a general purpose microprocessor would dissipate. However, in the iPhone example given there, Steve Jobs found the impact on battery run time to be just a factor of 2. Are these two statements in contradiction?