VLSI 1 - Notes Week 11

Ruben Schenk, ruben.schenk@inf.ethz.ch

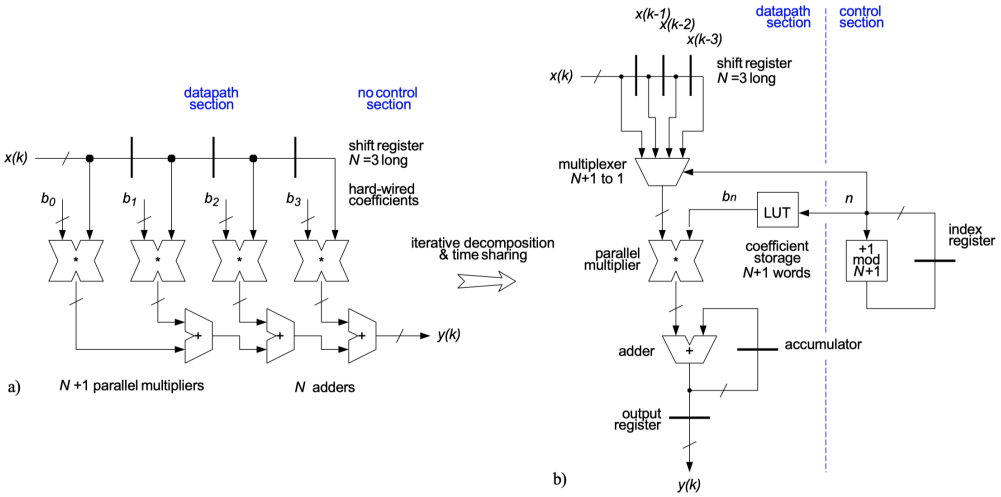
January 12, 2022

6.5.3 Iterative Decomposition & Timesharing Revisited

Decomposing and timesharing sequential computation is straightforward and can significantly reduce datapath hardware. Functional memory requirements remain the same as in the isomorphic architecture. There is somewhat of a mixed blessing energy-wise:

- – More uniform combinational depth reduces glitching activity
- – Extra multiplexers necessary to route, recycle, collect, and/or redistribute

Example: Consider the following third order traversal filter:



6.5.4 Digest

Retiming can help to optimize datapath architectures for sequential computations without affecting functionality nor latency:

1. Retiming, pipelining and combinations of the two can improve throughput of arbitrary feedforward computations
2. The associative law allows one to take full advantage of the above transforms by having a DDG rearranged beforehand.
3. Iterative decomposition and timesharing are the two options available for reducing circuit size.
4. Highly time-multiplexed architectures dissipate energy on ancillary activities that do not directly contribute to data computation.

6.6 Equivalence Transforms For Recursive Computations

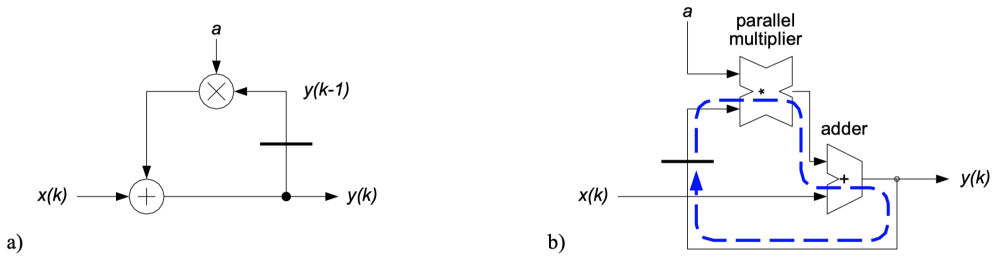
A computation is termed (*sequential and*) **recursive** if:

1. Result is dependent on earlier outcomes of the computation itself.
2. Edges with weights greater than zero are present in the DDG.
3. Circular paths (of non-zero weight) exist in the DDG.

6.6.1 The Feedback Bottleneck

Recursions such as $y(k) = ay(k-1) + x(k)$ which in the z domain corresponds to the transfer function $H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1-az^{-1}}$ have many technical applications, such as IIR filters, differential pulse code modulation encoders, or servo loops.

They impose a *stiff timing constraint,* however. The following figure shows a DDG (a) and an isomorphic architecture (b):



This has an iteration bound of $\sum_{loop} t = t_{reg} + t_* + t_+ = t_{lp} \leq T_{cp}$:

- This is no problem as long as the long path constraint can be met with available and affordable technology
- However, there is no obvious solution otherwise, recursion is a real *bottleneck*

6.6.2 Unfolding of First-Order Blocks

The key idea to solve the above-mentioned problem is to real the timing constraint by inserting additional latency registers into the feedback loop. A tentative solution must look like:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{N(z)}{1 - a^p z^{-p}},$$

where $N(z)$ is here to compensate for the changes due to the new denominator. Recalling the sum of geometric series we easily establish $N(z)$ as:

$$N(z) = \frac{1 - a^p z^{-p}}{1 - az^{-1}} = \sum_{n=0}^{p-1} a^n z^{-n}.$$

The new transfer function then becomes:

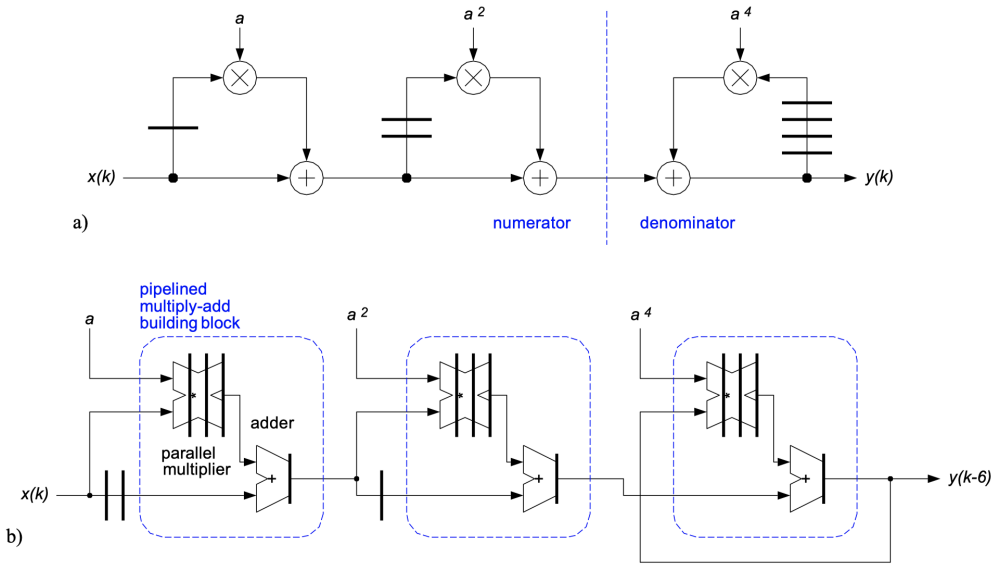
$$H(z) = \frac{\sum_{n=0}^{p-1} a^n z^{-n}}{1 - a^p z^{-p}}$$

and the new recursion in the time domain follows as $y(k) = a^p y(k-p) + \sum_{n=0}^{p-1} a^n x(k-n)$.

After unfolding by a factor of $p = 4$, the original recursion takes on the form

$$y(k) = a^4 y(k-4) + a^3 x(k-3) + a^2 x(k-2) + ax(k-1) + x(k).$$

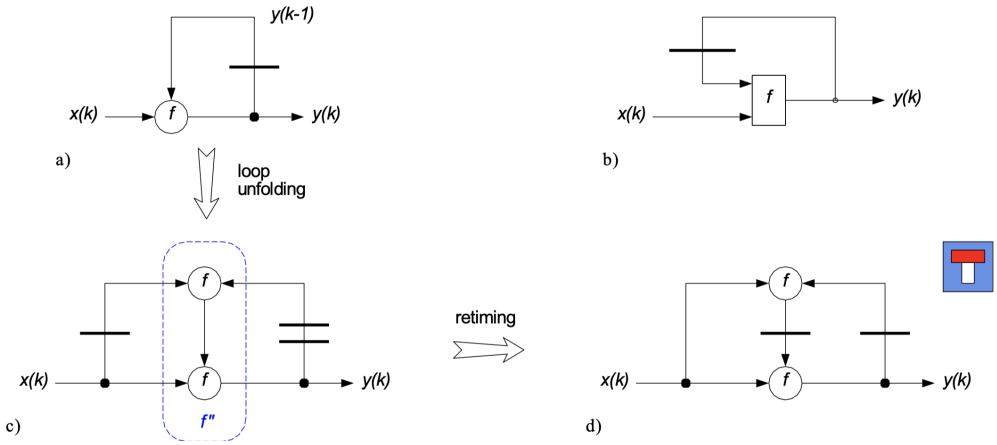
The *net result* is that the denominator now includes p unit delays rather than one! The following figure shows a DDG (a) unfolded by $p = 4$ and a high-performance architecture (b):



6.6.3 Non-Linear or General Loops

The most general case of a first-order recursion goes like $y(k) = f(y(k-1), x(k))$ and can be unfolded an arbitrary number of times, e.g. with $p = 2$ it becomes $y(k) = f(f(y(k-2), x(k-1)), x(k))$.

The figure below shows an original DDG (a) and an isomorphic architecture (b), as well as a DDG after unfolding by a factor of $p = 2$ (c), and the same DDG with retiming added on top (d):



All successful architectural transforms for recursive computations take advantage of algorithmic properties such as linearity, fixed coefficients, associativity, limited word width or of a very limited set of registers states. When the state size is large and the recurrence is not a closed-form function of specific classes, our methods for generating a high degree of concurrency cannot be applied.

6.6.4 Digest

Architectural transform	*Decomposition*	*Pipelining*	*Replication*	*Timesharing*	*Associativity etc.*	*Retiming*	*Loop unfolding*
Nature	universal	universal	universal	universal	algebraic	universal	algebraic
Applicable to	combinational	combinational	combinational	combinational	combinational	sequential non-recursive	sequential recursive

6.7 Generalization Of The Transform Approach

Architectural transform	Decom-position	Pipe-lining	Repli-cation	Time sharing	Associa-tivity	Retiming	Loop unfolding
Nature	universal				algebraic	universal	algebraic
Applicable to	combinational computations					sequential computations	
Impact on						nonrecurs.	recursive
A	−...=	=...+	+	−...=	=	=	+
Γ	+	=	−	+	=	=	=
t_{lp}	−	−	=, mux −	=	−...+	−	−
$T = \Gamma \cdot t_{lp}$	=	−	−	+	−...+	−	−
AT	−...=	−...=	=	=...+	−...+	−	+
L	+	+	=, mux +	+	=	=	+
E	−...+	−...+	=	=...+	−...+	=	+
Extra circuit overhead	recy. and cntl.	none	distrib., recoll., and cntl.	collect., redist., and cntl.	none	none	extra word width
Helpful for indirect energy saving	no	coarse grain yes	possibly yes	no	yes	yes	possibly yes
Compatible storage type	any	register	register	any	any	register	register

Chapter 7: Introduction to Embedded System Design

Left out for now, may will be added later if considered important for the exam.