

Bit Serial Architecture - Outline

CS294-7, Spring 1997, Day 11

1. Introduction
2. Why Interesting
3. Denyer - Design methodology
4. Adders
5. Multipliers
6. FPGA implementations

Introduction: Bit-Serial Architecture

- bit-serial communication - data/control transmitted bit sequentially on a single wire (rather than a parallel bus)
- bit-serial computing operations on operands bit sequentially

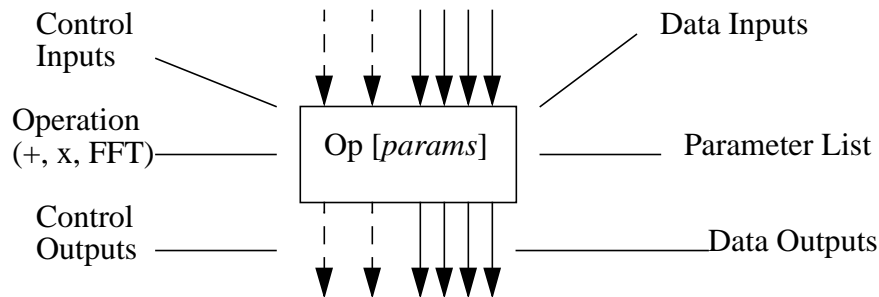
Why Interesting?

- wires are an expensive area resource - “find a way to do more with less” (positive)
- bit-serial components demonstrate very good Area-Time trade-off.(positive)
- bit-serial components are small (positive)
- potential for runtime binding of operand width (positive)
- long latency operations (negative)
- At times delay is more important than efficiency - high throughput, realtime computation (negative)
- interesting end of Area-Time trade-off spectrum
- assumptions: 1. globally synchronous bit clock; 2. LSB 1st

Note: 1. clock skew will cut into bit time; 2. Arithmetic operation calls for sending LSB first.

Denyer & Renshaw

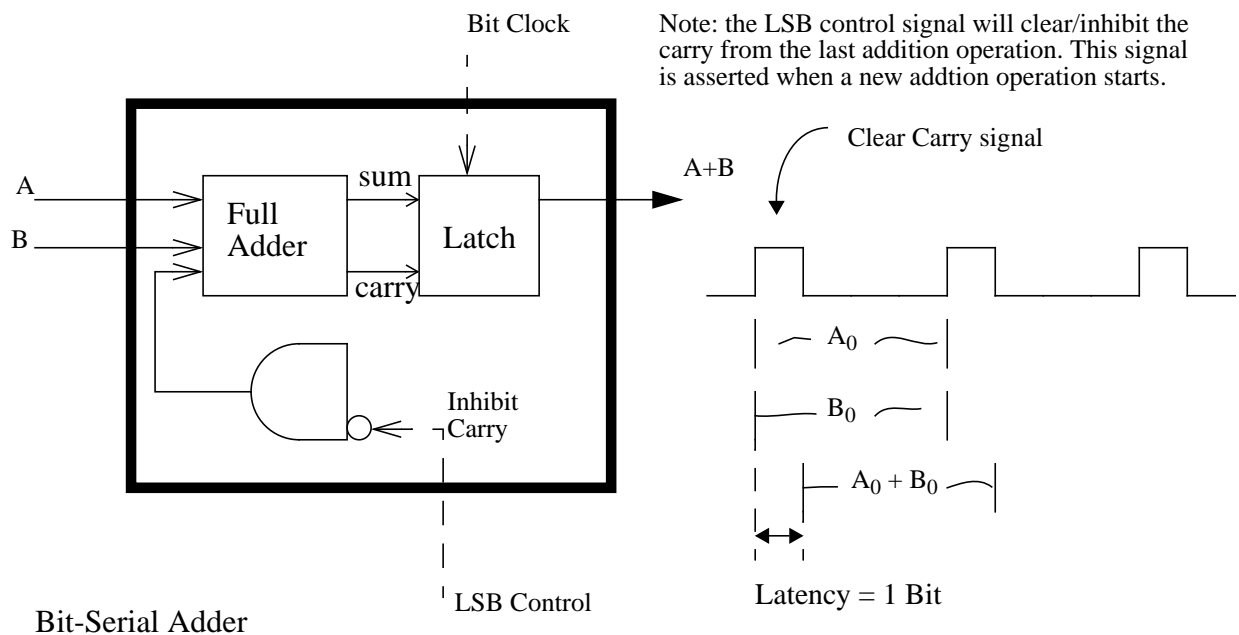
Generic operator definition



- parameters - values bounded at creation (compile?) time
- operators have a fixed known latency expressed in “bits” (& words)

Notes: the parameter list could be configurable bits for CLB, values bounded at compile time, or shift amount, for example.

Example operator: Adder

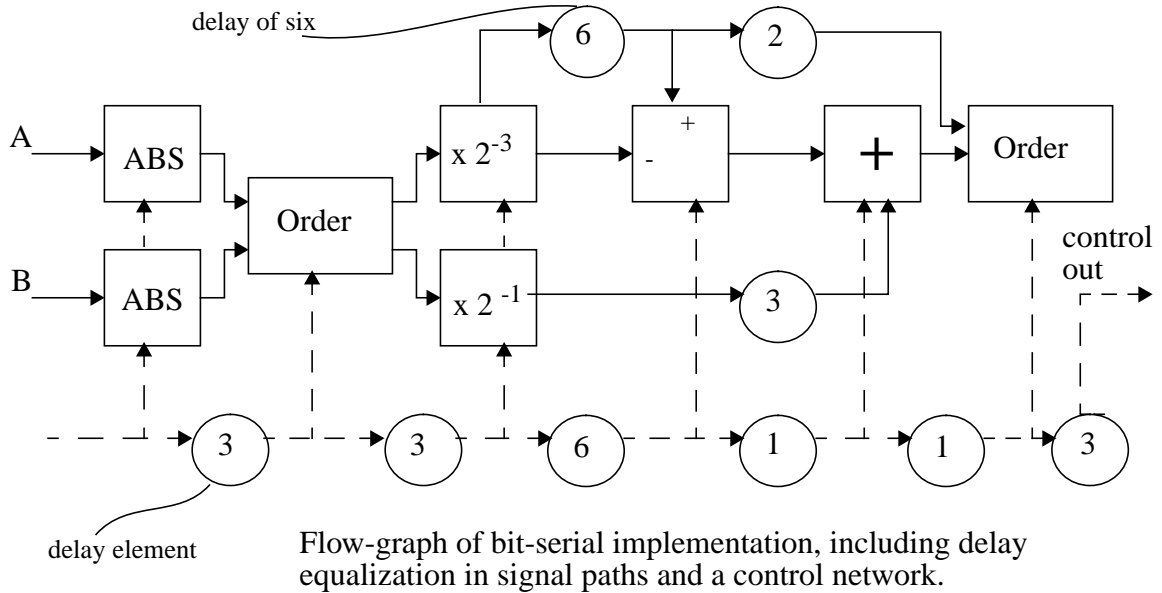
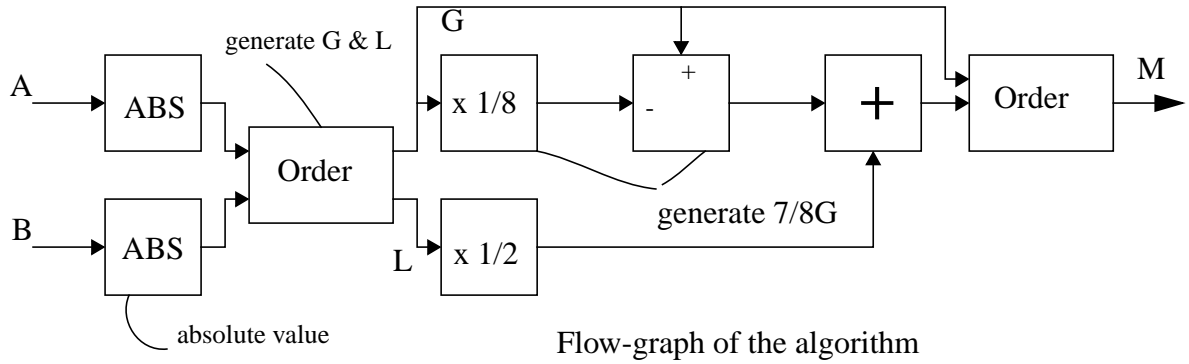


Functional Design (intended for VLSI silicon compilation from high level language into gates)

- Imagine a library of operators of generic type: adder, multiplier, delay, filter, FFT...
- Computing structures are built from a functional specification using library operators and automatically translated to silicon layout. Bit serial simplifies the translation.
- Example Design: $M = \text{magnitude of a complex number } I + jQ$

$$M = \max \{G, 7/8G + 1/2L\}$$

G = greater (|I|,|Q|)
 L = lessor (|I|,|Q|)



Notes: Because different block has different delay, delay is added to make sure that LSB of input arrives at the same time. ABS stands for the absolute value operation, not anti-lock brake.

Table 1: Latencies

block	delay
ABS	n+3
Order	n+3
DSHIFT	p+3
ADD	1

Table 1: Latencies

block	delay
SUB	1

Note: n is the number of bits in a word and p is the shift amount.

- control globally generated
- layout created
- hierarchy works. Note: this block can be used by another by applying a wrapper.
- design style good match to reconfigurable computing (spatial computation).

Adder Efficiency

1. Parallel adder

with sequential carry propagate:

Area: $O(n)$

Time: $O(n)$

Area-Time: $O(n^2)$

fast adder (ex: Carry Look Ahead)

Area: $O(n)$

Time: $O(\log n)$

Area-Time: $O(n \log n)$

Note: FPGA fast carry chain adders are still $O(n^2)$

2. Bit Serial (see page 2 for schematic)

Area: $O(1)$

Time: $O(n)$

Area-Time: $O(n)$

3. Hiding latency:

example: add K n-bit numbers

Table 2: Bit Serial vs Bit Parellel

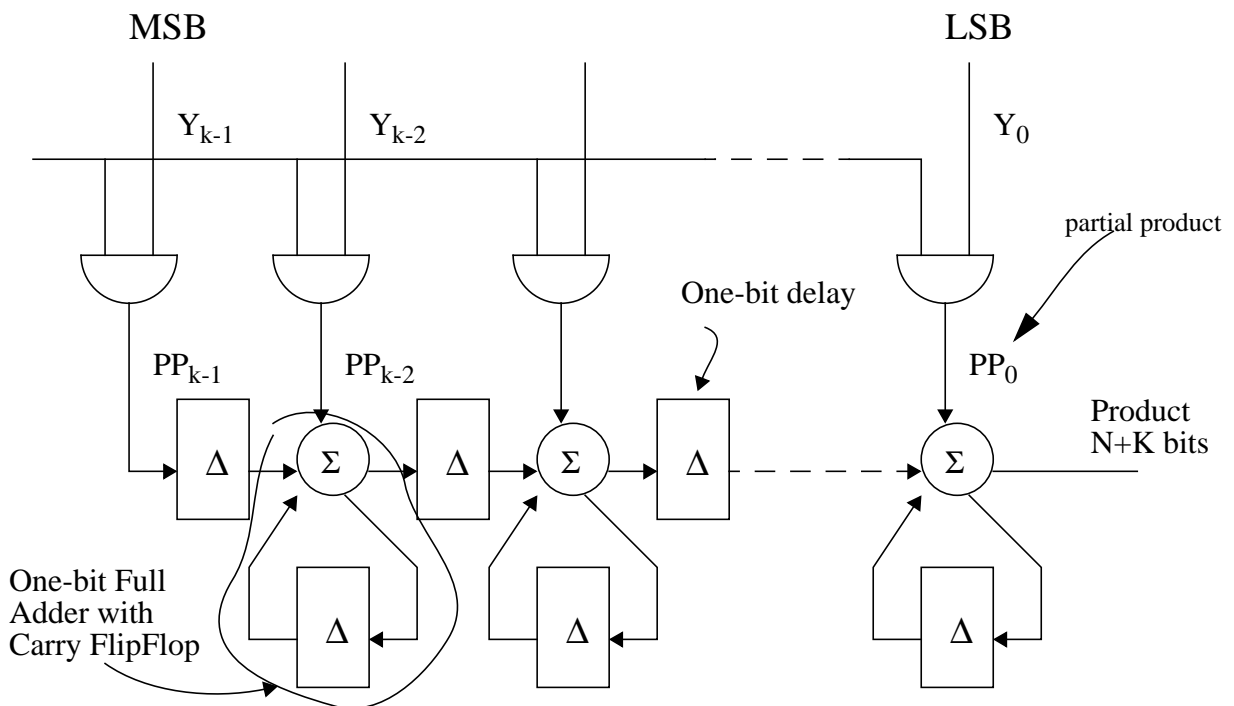
	Bit-Serial	Bit Parallel
Area	$K - 1$	$n(k - 1)$
Time	$n + \log K$	$\log K$

Note: As soon as the first bit is done, it is passed on to the next operator.
 Note: There exists equally efficient parallel version.

Bit-Serial Multipliers

				x3	x2	x1	x0
			X	y3	y2	y1	y0
				x3y0	x2y0	x1y0	x0y0
			x3y1	x2y1	x1y1	x0y1	
		x3y2	x2y2	x1y2	x0y2		
	x3y3	x2y3	x1y3	x0y3			
z7	z6	z5	z4	z3	z2	z1	z0

Attempt #1

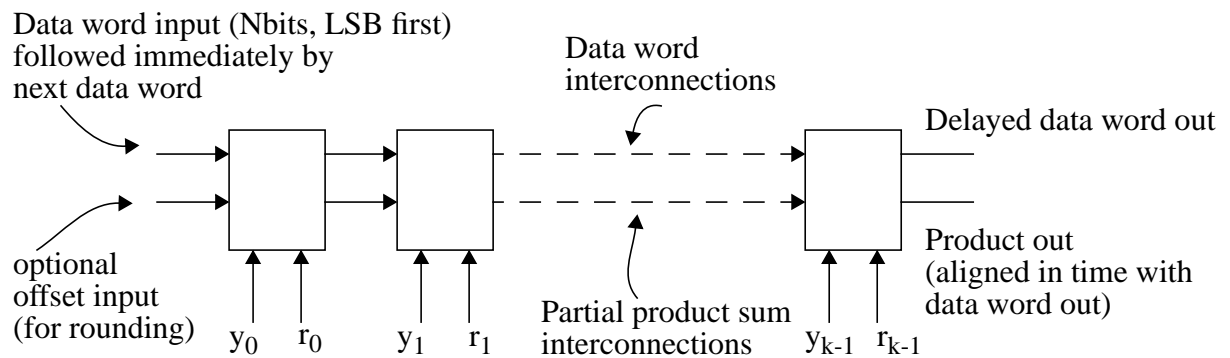


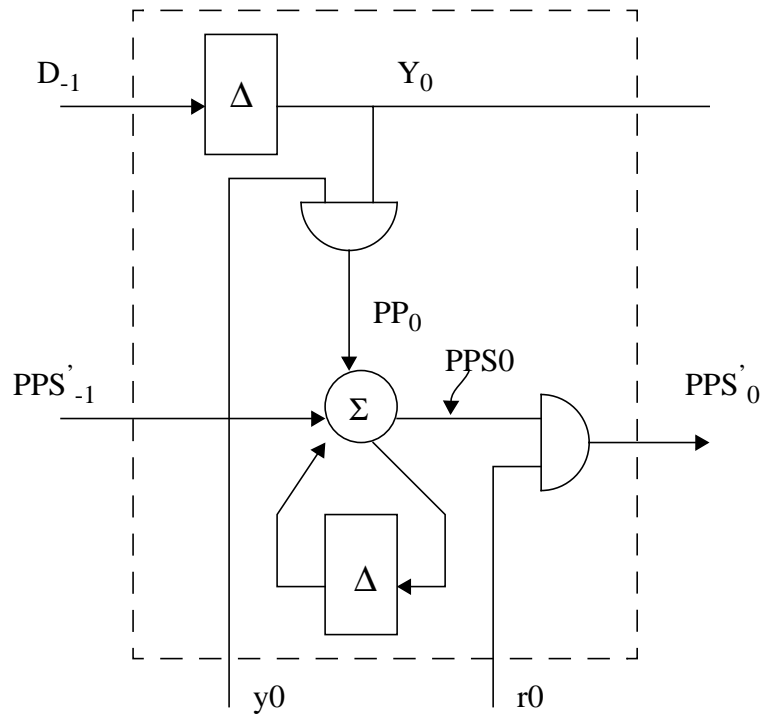
- $k + n$ bit product takes $k+n$ cycles
- cannot overlay next multiply (pipeline)
- observation: if we do not need to keep all the bits, we can pipeline the multiplier.
- question: which bits to keep? In DSP or graphic operation, only the higher bits are needed.

Note: In computer graphics and DSP operation, data is often in fixed point (fraction) representation. For example, a 4 bits fixed-point data of 1.3 representation means that MSB is before the binary point and the others three bits are after the binary point. Therefore 1001 in 1.3 binary

fixed-point representation means $1 \frac{1}{8}$ in decimal representation. 1001 in 2.2 binary fixed-point representation means $2 \frac{1}{4}$ in decimal representation. 1.15, 0.16 8.8 are the commonly used 16-bit fixed-point data representation in DSP and computer graphics.

Attempt #2



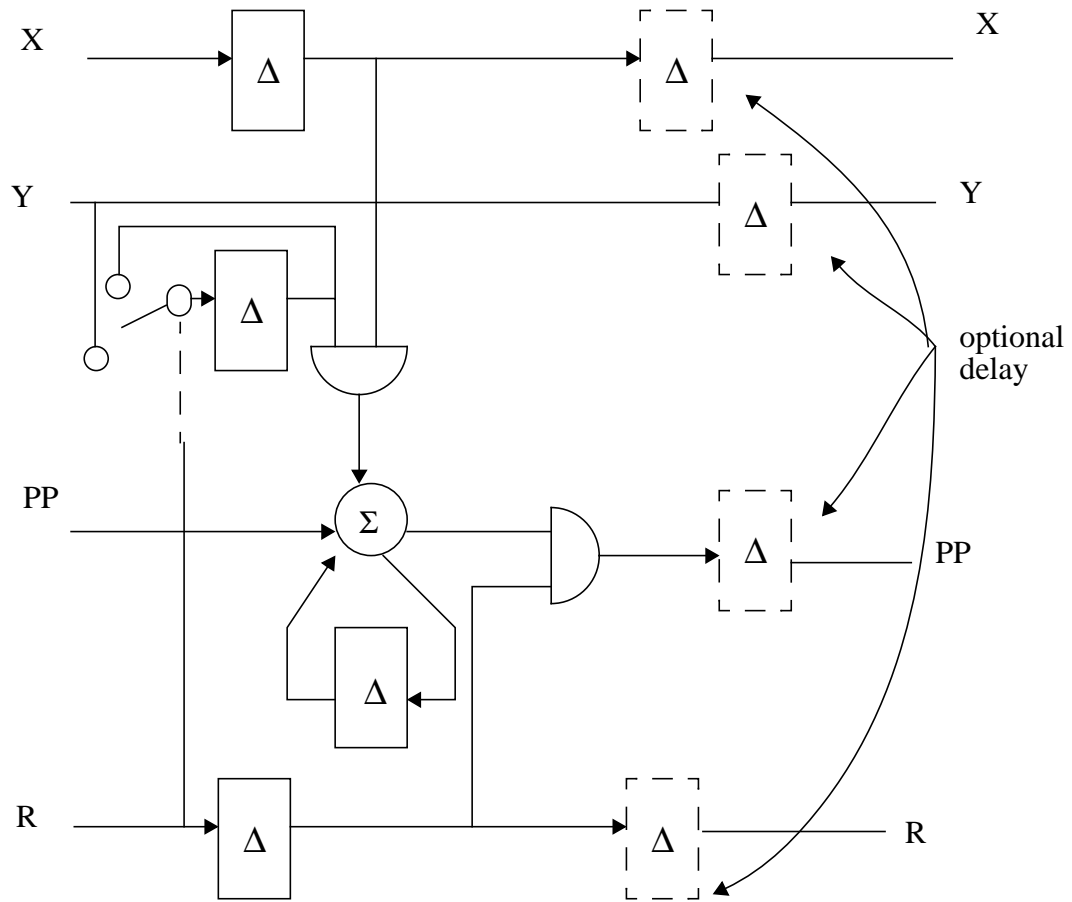


- calculates one column of partial product at a time
 - low product bits are dropped (but carries are preserved) to enable previous multiply to finish.
- Note: the basic idea of the above method is to truncate the partial products, but keep carry in the cell. This will enable the previous multiply to finish correctly.

Modifications for:

- serialization of y, r
- pipelined sum propagate
- 2's compliment
- fixed coefficient
- both high and low outputs

Attempt #3



Multiplier Efficiency

	Parallel array	Bit Serial	Pipelined array
Area	$O(n^2)$	$O(n)$	$O(n^2)$
Time	$O(\log n)$	$O(n)$	$O(1)$
Area-Time	$O(n^2 \log n)$	$O(n^2)$	$O(n^2)$