Efficient digital systems for computer arithmetics

Loofmann

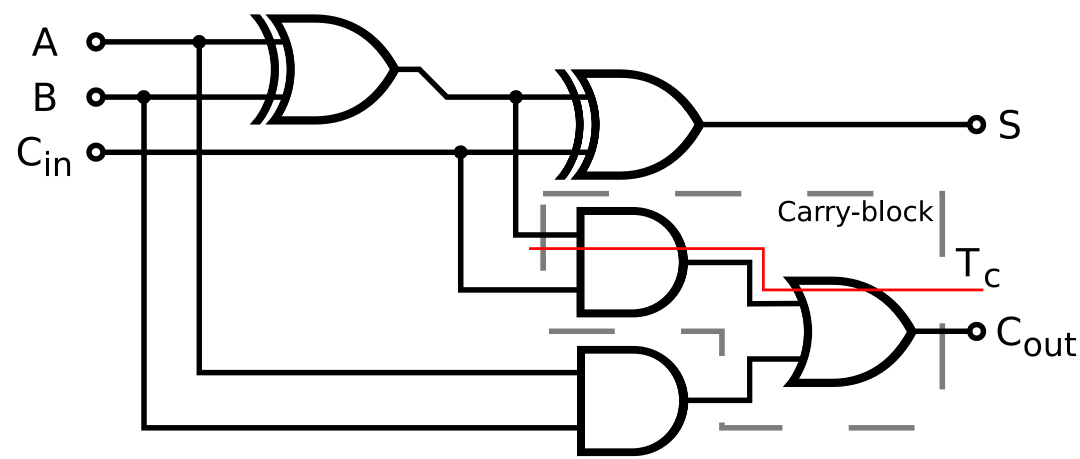
2016-05-27 @ AFRA

Introduction

- Who am I?
  - Stefan Laufmann (Loofmann)
  - `loofmann@afra-berlin.de`
- What do I do?
  - study of Computer Engineering (Msc.) @ TU Berlin
- Why do I know this stuff?
  - interest in design of digital systems
  - took a course in the last semester: "Computer Arithmetics: Circuit Perspective" by Dr. Ahmed Elhossini

What is this talk about?

- digital systems (circuits) for arithmetic functions
- arithmetic functions as in:
  - addition, multiplication, division
- numbers as in:
  - unsigned integers
- digital circuits as in:



Outline

1. Define our measure
2. Addition – the straightforward way
3. Addition – the clever way
4. Multiplication – done precisely
5. Multiplication – done not so precisely

Ex: Design Process of Digital Systems

How are digital systems designed? (from my experience)

1. design circuit on whiteboard
2. describe circuit in HDL (Verilog, VHDL)
3. throw powerfull and awful software tools at it
4. get a circuit out

Define our Measure

What makes a "good" circuit?

- **area** used on final chip
- speed of the circuit: critical path **delay**

How do you measure this?

- e. g. for CMOS technology
  - smallest area is that of 1 inverter (α)
  - shortest delay is that of 1 inverter (Δ)

critical path

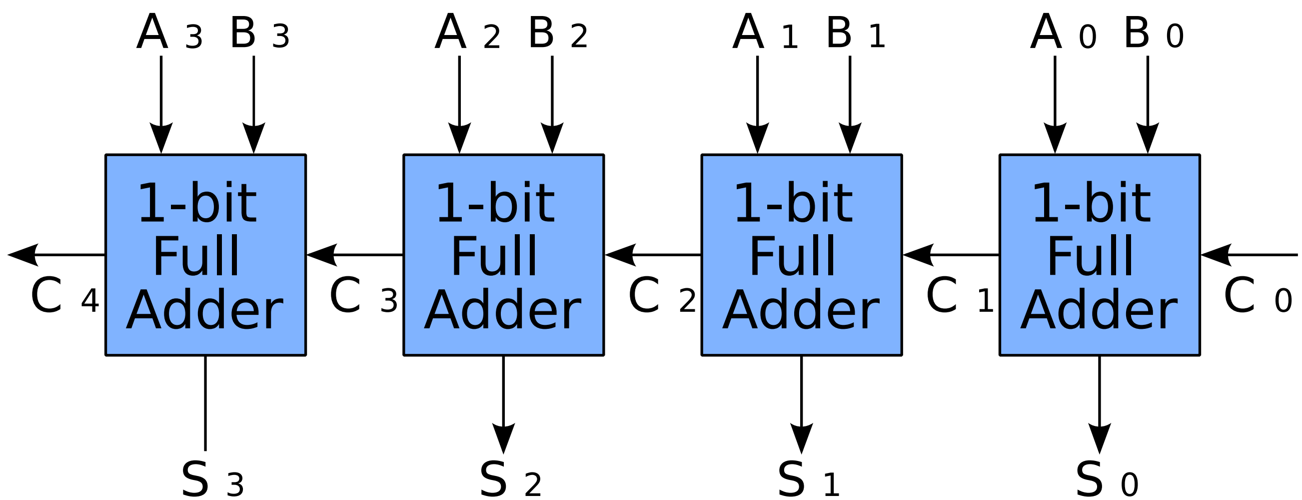
longest logical path through a circuit

Addition – the straightforward way

- adding two bits and carry bit is easy (seen above)

  How to add two numbers (e. g. 4 bit)?

  – add each bit position one after the other



source: `https://en.wikipedia.org/wiki/Adder_%28electronics%29#/media/File:4-bit_ripple_carry_adder.svg`

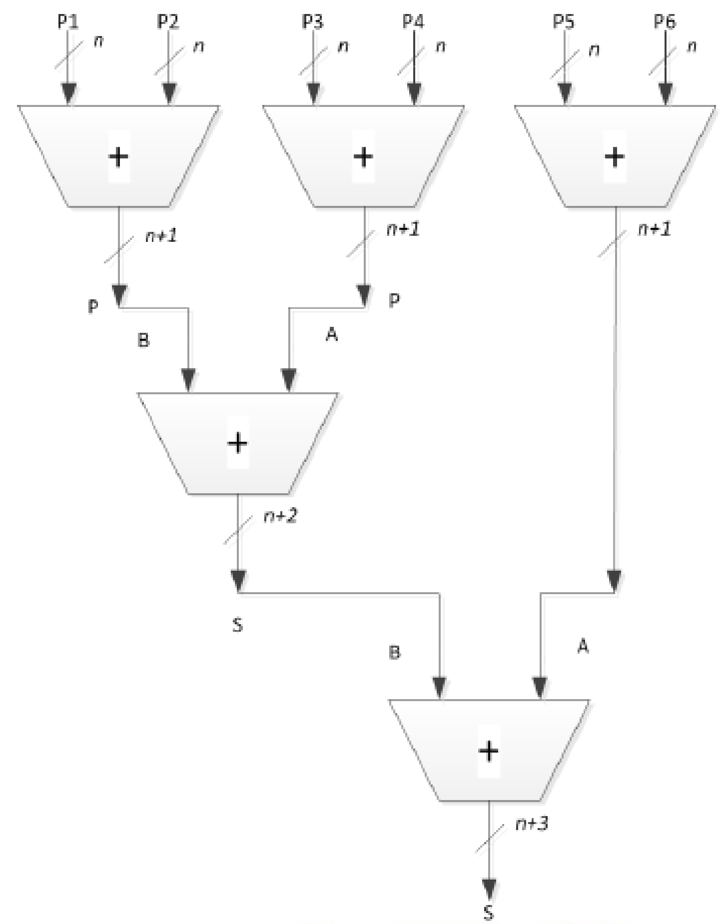
Addition – the straightforward way (2)

How to add multiple numbers?

- add each operand one after the other

What are the downsides of this?

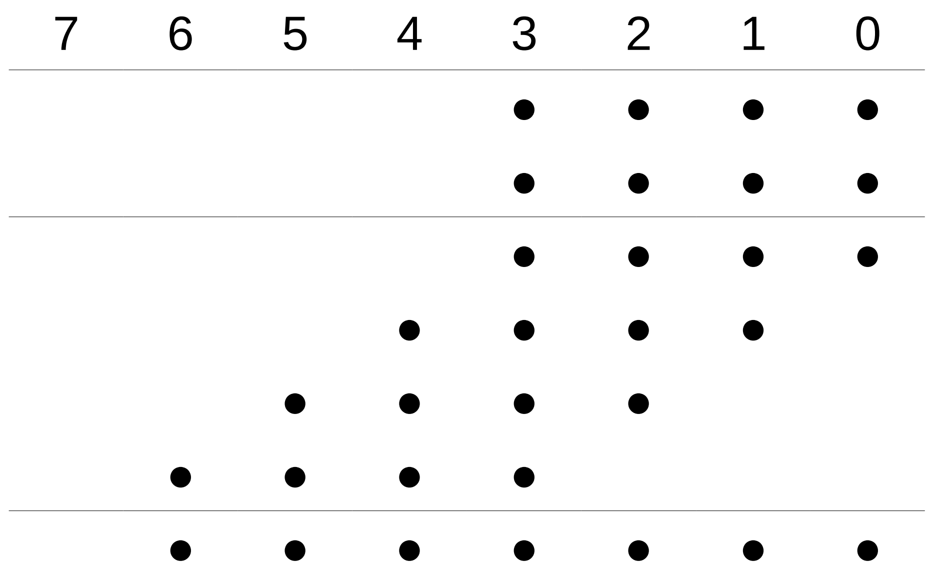
- consumes large area on the chip
- high delay due to long critical path



source: TU Berlin, *Computer Arithmetics: Circuit Perspective*; lecture 3, slide 21

Ex: Dot Diagrams aka dotagrams

- technique to visualize the structure or arithmetic operations
- also used for computer arithmetic
- dot represents binary digit



example for multiplication of two 4-bit numbers

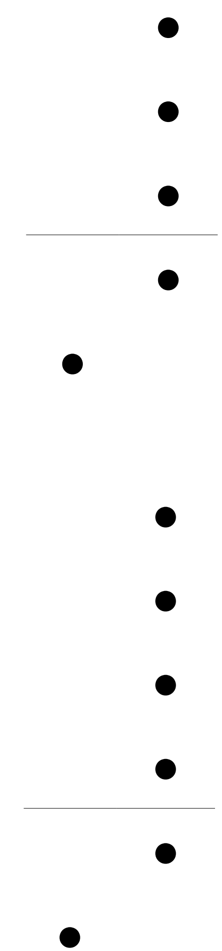
Addition – the clever way

- ways to speed up addition of two operands:
  - carry-lookahead adder
  - conditional sum adder
  - carry-select adder

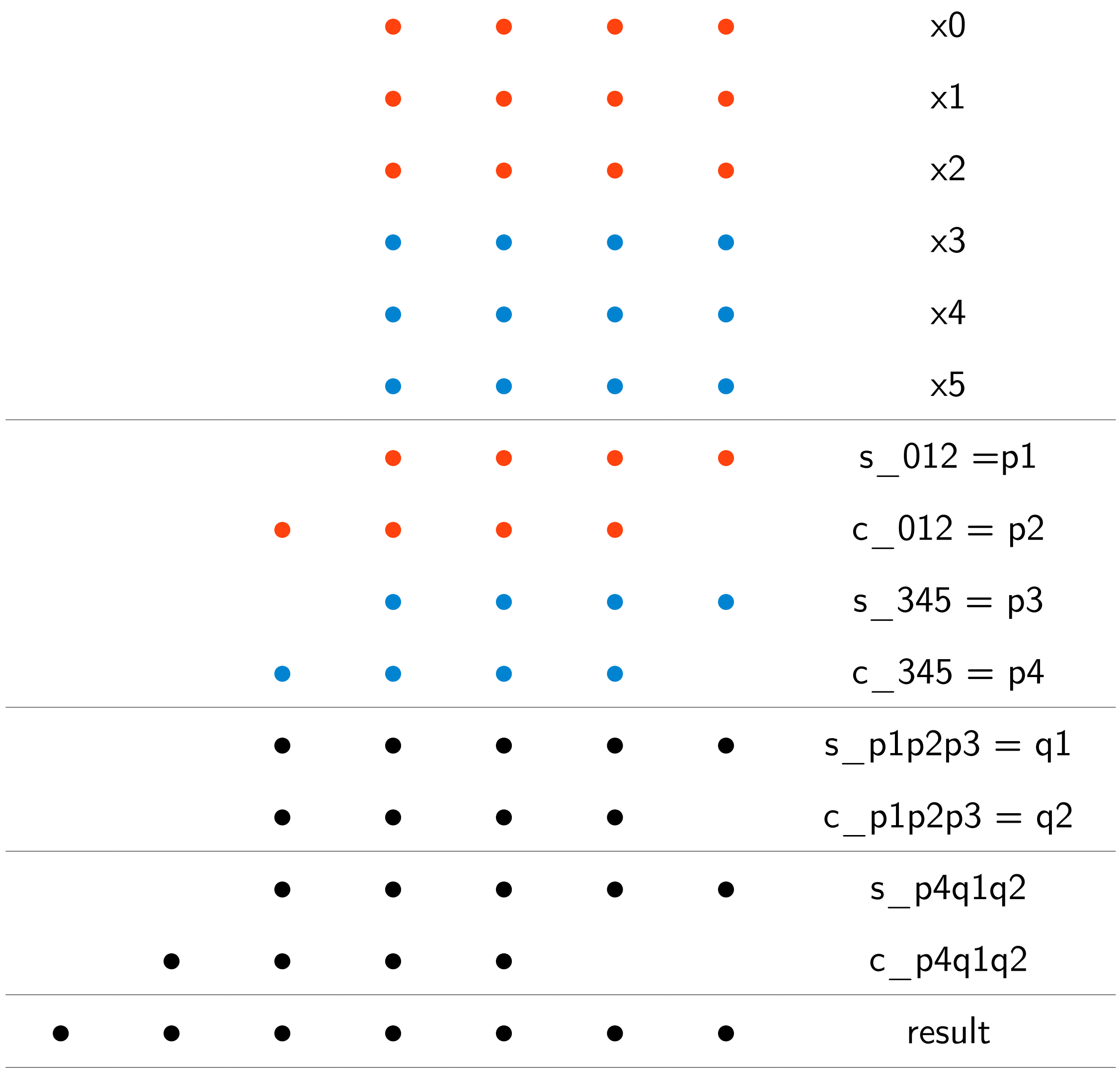
  How to speed up the addition of multiple operands?
  - use so called "counters":
    - 3:2
    - 4:2
    - m:n

Addition – the clever way (2)

- a full-adder is called 3:2-counter
- addition of two bits can be seen as counting the 1's in the two operands
- with only little overhead there can also be a 4:2-counter
- these counters can be combined to easily add multiple operands



Addition – the clever way (3)



- addition of six 4-bit numbers in a so called "carry save adder"
- 4 rows of 3:2-counters
- 1 final full adder
- total area: 15FA + 1HA + 6bit adder
- delay: 4 stages of FA

Note:

Different counter types can be mixed but CMOS technology processes benefit from regular structures.

Multiplication – done precisely

- in fact multiplication is just addition
- different multiples of one factor get added according to the bits in the other factor

  e. g.:

  - $1101_2 * 1001_2 = 1*1101_2 + 8*1101_2 = 1110101_2$

    $(13 * 9 = 117)$
- the multiples – called partial products – get shifted according to the bit position they originated from

Multiplication – done precisely (2)

- 3:2-counters or 4:2-counters are used to add partial products
- this is – again – the straightforward way

  Is there a better way?

  Yes!

  - For example: multipliers using booth-encoding
  - use the fact that partial products occur more than ones in the multiplication
  - this reduces the number of partial products to add

Multiplication – done precisely (3)

- Radix-4 Booth Multiplier:
  1) take 3 digits from the multiplicand (start from the right)
  2) generate partial product according to the digits
  3) take next 3 digits (2 steps to the left from last ones)
  4) repeat 2 – 3 until no digits left
  5) add all partial products

digits from multiplicand			factor for partial product	digits from multiplicand			factor for partial product
0	0	0	0	1	0	0	-2
0	0	1	1	1	0	1	-1
0	1	0	1	1	1	0	-1
0	1	1	2	1	1	1	0

Note: Here we deal with signed integers (two's complement). This influences the addition circuit!

Multiplication – done precisely (4)

Give me numbers!

Radix r	Group Size (bits)	Reduction Ration in SD *(n/k)*	Radix Set Size	Encoder Control Signals	Selector AND-OR (input)	HARD Multiples	FPA's	Critical Path (Gates)
2	2	1	3	2	1 AND	0	0	3
4	3	0.5	5	3	4	0	0	6
8	4	0.333	9	5	8	1	1	$7+t_f$
16	5	0.25	17	9	16	4	3	$8+t_f$
32	6	0.2	33	17	32	11	7	$9+t_f$
64	7	0.166	65	33	64	26	15	$10+t_f$
256	9	0.125	257	129	256	120	63	$11+t_f$

source: TU Berlin, *Computer Arithmetics: Circuit Perspective*; lecture 6, slide 37

Multiplication – done not so precisely

- precise results not always required

  Maybe we can figure out a way to do do multiplication different and faster but with a small error?

Computer Multiplication and Division Using Binary Logarithms*

JOHN N. MITCHELL, JR.,† ASSOCIATE, IRE

John N. Mitchell. "Computer Multiplication and Division Using Binary Logarithms". In: Electronic Computers, IRE Transactions on EC-11.4 (Aug. 1962), pp. 512–517.

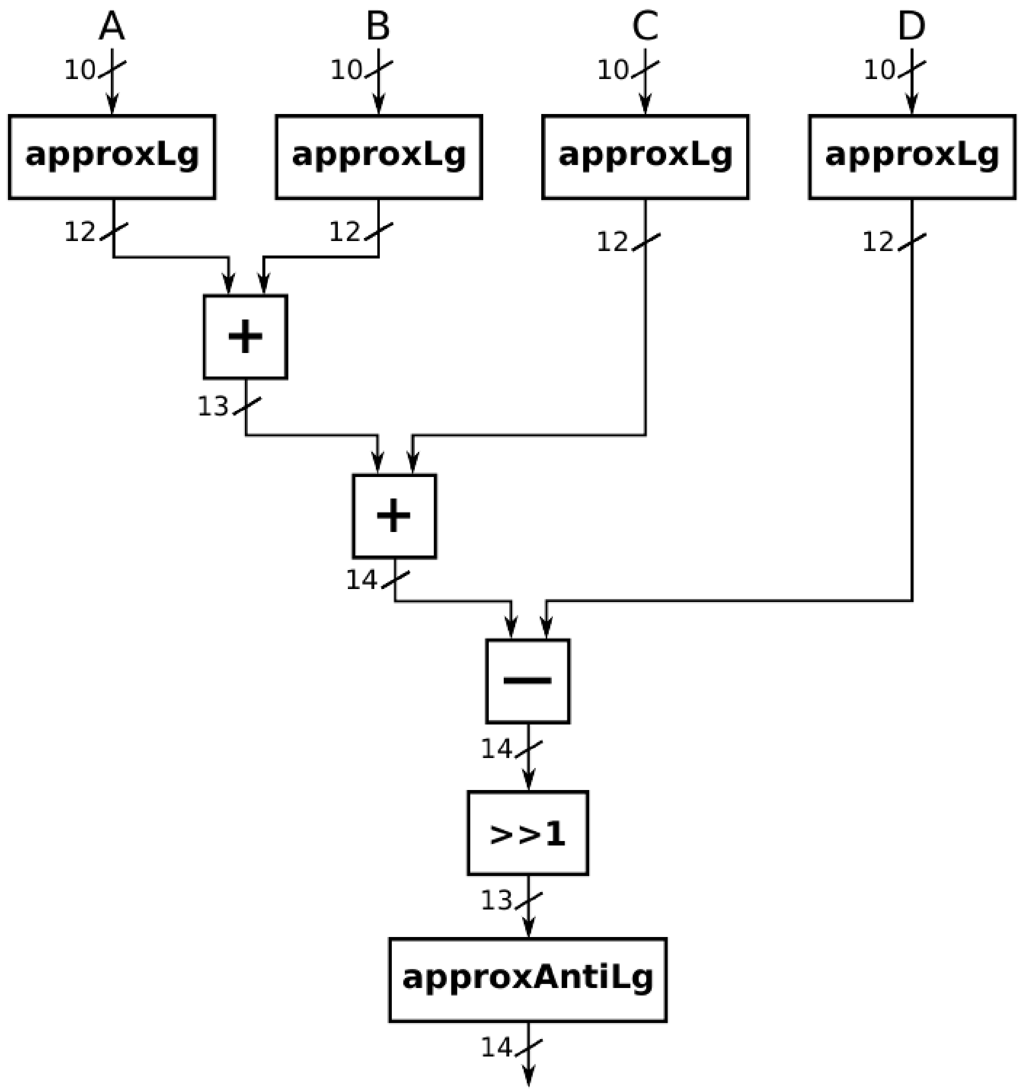
Multiplication – done not so precisely (2)

- it is proven that:

  $a*b = \log^{-1}(\log(a*b)) = \log^{-1}(\log(a)+\log(b))$
- if we have an efficient (anti-)logarithm for binary numbers we can save the multiplication
- approximate binary logarithm:
  - find position of most significant 1, treat it as integer part of result
  - rest of number becomes fraction part of result
  - e. g.: $\log(1101) \approx 11.101$ $\log(13) \approx 3.625$
- anti-logarithm is the reversion of this process

Multiplication – done not so precisely (3)

- the multiplication is then performed as:
  1. compute approx. log
  2. add
  3. compute approx. anti-log
- easy solution for functions with multiplication and division
- picture shows circuit for:
  - $f = \sqrt{\frac{A \cdot B \cdot C}{D}}$



The last slide, yeah!

Questions?