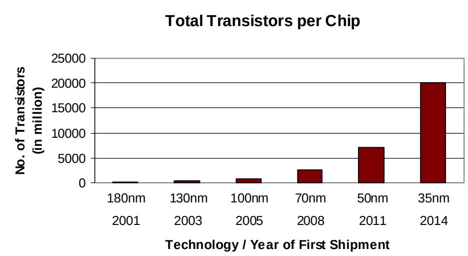
EE 486 : lecture 1, the integers

M. J. Flynn

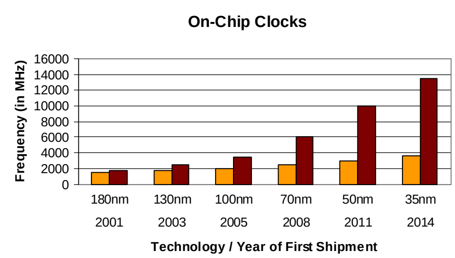
The role of arithmetic

- With increasing circuit density available with sub micron feature sizes, there's a corresponding broader spectrum of arithmetic implementations,
- Signal processors, controllers, wireless dsp, crypto, etc.

SIA Roadmap - 1



SIA Roadmap - 2



Semiconductor Industry Roadmap

Semiconductor Technology Roadmap (1999)

Year	2001	2005	2008	2014
Technology generation (nm)	180	100	70	35
Wafer size (mm)	300	300	300	450
Defect density (per m^2)	1742	1262	1101	837
µP die size (mm^2)	450	622	713	937
Chip Frequency (MHz)	1767	3500	6000	13500
MTx per Chip (Microprocessor)	220	882	2494	19949
Max Power (W)	115	160	170	183

Some term used in number representation

- The integers: weighted positional number system: *wpns*, residue number system: *rns* and log number system: *lns*
- Floating point(*fpns*): IEEE and specialized formats
- Redundant number representation *rnr*. This can be used in any number system.
- Optimized representations: log, exponential, continued fraction, etc

The integers

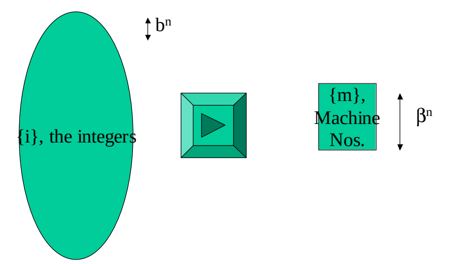
- Weighted positional number system (*wpns*)
  - Non redundant and redundant forms
  - $X = d_0\beta^0 + d_1\beta^1 + \ldots + d_{n-1}\beta^{n-1}$ where β is the radix and $\{d_i\}$ is the digit set
  - If number of symbols in digit set $\{d_i\} = \beta$ then we have non – redundant system
  - If number of symbols in digit set $\{d_i\} > \beta$ then we have redundant system

The integers

- In general redundant numbers can offer some advantages, such as carry free addition. The Roman Numeral system is a redundant system if one allows for the use of improper forms.
- The only redundant system of interest to us is the signed digit system (*sds*) which we'll consider later.

WPNS and *RNS* (non-redundant)

- The residue number system uses n relatively prime moduli and defines each digit independently as $d_i = X \bmod m_i$
- Two types :optimal and binary based
  - Optimal : ***rns*** system whose largest modulus (m_n) is the smallest possible to provide a required representation capacity
  - Binary : largest modulus of the form 2^n and all others of the form $2^{n-i}-1$

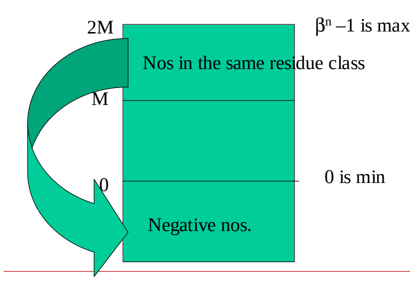


Mapping the integers onto the machine numbers: $m = i \bmod \beta^n$

The machine numbers

- The integers, I, map onto the machine numbers, M. I: $i \rightarrow m \in M$, $i \bmod \beta^n \rightarrow m$, o
- The residue, m, is the least positive remainder, o is overflow
- Modular operations:
  - (m+n) mod M =(m mod M+n mod M) mod M
  - (m-n) mod M =(m mod M - n mod M) mod M
  - (mxn) mod M =(m mod M x n mod M) mod M

The negative numbers

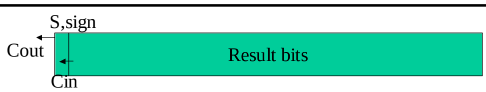


Representing negative numbers

- Sign and magnitude (s+m)
- Radix complement (*rc*), diminished radix complement (*drc*). (2M - x) mod 2M= -x
- Where 2M = β^n for *rc* and "2M" = β^n –1 for *drc*; in binary n is the number of bits in a word, the representational capacity:
  max positive = 2^{n-1}; min = 0 (*rc*) and
  max positive = 2^{n-1} – 1; min = 0 (*drc*)

The negative numbers: complements

- The complement of x is "2M"-x where 2M is β^n for *rc* or β^n –1 for *drc*.
- (x-y) mod 2M= (x +(2M –y)) mod 2M = (x-y) mod 2M; result is a valid machine number if "x" and "-y" have opposite signs.
- Otherwise overflow, o, is possible and must be detected.



Signs
+ = 0
- = 1

x	y	Σ signs	Cin	Cout	Over-flow
0	0	0	0	0	no
0	0	0	1	0	yes
1	1	0	1	1	no
1	1	0	0	1	yes
1	0	1	0	0	no, x>y
1	0	1	1	1	no, y>x

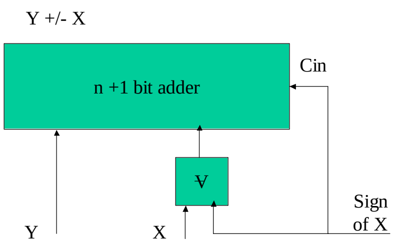
Overflow detection

- Overflow, o is detected when
- o = Cin ∀ Cout

Finding the radix (2's)complement

- Suppose that i is the first non zero bit in X, then for all x_j, i > j > or = 0; *rc* (x_j) = x_j = 0. For bit i *rc*(x_i) = (β – x_i) = x_i = 1
- For bits j > i, *rc* (x_j) = β –1 – x_j (inversion).

Finding the radix (2's)complement



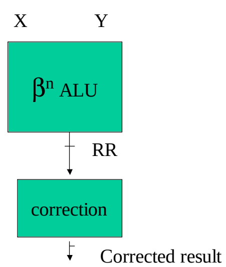
Finding the diminished radix (1's) complement

- For all bits i; *drc* $(x_i) = \beta - 1 - x_i$
- So the *drc* is just the bit wise complement of X.
- But since a β^n ALU is used we must correct the result so that it is mod $(\beta^n - 1)$. I.e. we want to stay in the *drc* number system but our ALUs are in a radix based system.

Fixing up a radix based result so that it remains in the drc.

- If radix result (RR) RR< β^n-1 then DRR=RR nothing need be done.
- If radix result RR= β^n-1 then DRR=0
- If RR> β^n-1 then
- DR= RR + [RR/ (β^n-1)]

drc radix result fix up



Integer multiply

- n bits x n bits = 2n bits unsigned
- In s + m product is 2n-1 bits
- In 2's complement -2^n is representable in n bits but the product -2^n x -2^n is not representable in 2n-1 bits

Integer divide: a/b=q+r/b

- In division result q has same sign as a, the dividend, but the result is a (q,r) pair and thus not unique. While (a) can be 2n bits, (b,q and r) are n bits.
  - If magnitude q is the same regardless of the signs of a,b result is ***signed*** division
  - If r is always the lpr (least positive remainder, including 0) then the (q,r) result is ***modular*** division

Division

- x/y= q + r/y; any (q,r) satisfies this, so the division result has many correct results.
- $\div_s$ signed division: select q so that the quotient is the same regardless of the signs of x,y.
- $\div_m$ modular division: select q so that the remainder is always the least positive remainder.
- Many other forms:such as floor division, q closest integer to 0 and r is a signed remainder.

shifts

- Logical shifts: all bits shift (left or right).
- Arithmetic shifts: sign is fixed, other bits shift left or right.
  - Left shift by p multiplies by 2^p; shift 0's into the lsb.
  - Right shift by p divides by 2^p; shift sign bit into the msb BUT be careful, the result depends on the complement coding used.

Integer divide

- On arithmetic shift division results depend on the type of integer complement coding that's used.
  - If magnitude q is the same regardless of the signs of a,b result is ***signed*** division
  - If r is always the lpr (least positive-incl 0 - remainder) then (q,r) result is ***modular*** division
  - 1's complement produces a ***signed*** (q,r)
  - 2's complement produces a ***modular*** (q,r)

Redundant number representations (*rnr*)

- Applicable to any number system.
- The signed digit number system offers carry free addition / subtraction
- SD numbers represent a number with radix $\beta > 2$ using digits $\{-\alpha, \ldots, -1, 0, 1, 2 \ldots, \alpha\}$ where $\beta/2 < \alpha < \beta$.
- Summing 2 digits $p_i = x_i + y_i$. If p_i exceeds α then it is recoded as $w_i = p_i - \beta$ with a carry of 1

Redundant number representations (*rnr*)

- Summing 2 digits $p_i = x_i + y_i$. If p_i exceeds α then it is recoded as $w_i = p_i - \beta$ with a carry of 1
- Then the sum is $s_i = w_i + c_{i-1}$
- The redundant condition assures that no carry will propagate more than a single digit
- As $-\alpha + 1 < w_i < \alpha - 1$
- Extendable to binary, $\beta = 2$; because of conversion not much used directly at least.