Cairo University
Electronics and Communications Engineering

Computer Arithmetic, Lecture 4: Are there any limits?

Hossam A. H. Fahmy

Limits on what and why look for them?

Time indicates how quickly the operation executes.

Area (gate count) indicates how large the circuit is. This translates into initial cost in design, managing the complexity, testing, and fabrication.

Power represents the running cost to operate the circuit *and to cool it.*

The three are linked and the figure of merit is

$$\text{merit} = T^a A^b P^c$$

Why are there limits?

- Fundamental issues:

  Time: Carry propagation, Leading One Detection, Sticky bit, Steps of FP add,. . . (sequential $\rightarrow$ parallel)

  Power: Is there really any *fundamental* minimum amount of energy for computation? (The answer is 'no', the *practical* answer is 'yes'.)

- Practical issues:

  Time: Switching speed of the transistors, capacitance and resistance of the wires, maximum fan-in for gates, . . .

  Area: How large are the chips that we can currently fabricate? (Price, yield, mechanical issues, . . .)

  Power: Can the package dissipate that much heat? Can the source supply that much power?

The Residue Number System

RNS uses relatively prime positional bases, for example: $(2, 3, 5)$ and $(4, 5, 7, 9)$.

Any number is represented by its residues after dividing the number by the base.

Example 1 To convert the decimal number 29 to a residue number with the bases $5, 3, 2$, we compute:

$$\begin{aligned} R_5 &= 29\mathbf{mod_5} = 4 \\ R_3 &= 29\mathbf{mod_3} = 2 \\ R_2 &= 29\mathbf{mod_2} = 1 \end{aligned}$$

and say that the decimal number 29 is represented by $[4, 2, 1]$.

The *Chinese Remainder Theorem* ensures that each number less than $(5 \times 3 \times 2)$ has a unique representation.

Why RNS?

There are *no carries between columns* in addition and multiplication.
Remember the properties of modular arithmetic: if $N' = N\mathbf{mod}_\mu$ and $M' = M\mathbf{mod}_\mu$, then

$$\begin{aligned}(N+M)\mathbf{mod}_\mu &= (N'+M')\mathbf{mod}_\mu\\ (N-M)\mathbf{mod}_\mu &= (N'-M')\mathbf{mod}_\mu\\ (N\times M)\mathbf{mod}_\mu &= (N'\times M')\mathbf{mod}_\mu\end{aligned}$$

Arithmetic is closed (done completely) within each residue position.

For subtraction we use complement coding so that $X^c = [x_i^c]$.

As simple as that

Example 2 In the 5,3,2 residue system, $M = 30$, integer representations 0 through 14 are positive, and 15 through 29 are negative (i.e., represent numbers -15 through -1). Calculate $(8)^c$ and $(9)^c$ as well as $8-9$.
Solution: The representations of 8 and 9 are

$$\begin{aligned}8 &= [3,2,0],\\ 9 &= [4,0,1]\end{aligned}$$

So, $(8)^c = [2,1,0]$ i.e. $5-3, 3-2$, and $(2-0)\mathbf{mod_2}$ while $(9)^c = [1,0,1]$.

$$\begin{array}{rcccl} 8 & = & 8 & = & [3,2,0] \\ \underline{-9} & = & (9)^c & = & \underline{+[1,0,1]} \\ -1 & & & & [4,2,1] \;\; = 29 \text{ or } -1 \end{array}$$

Selection of the Moduli

- Considerations:
  - Relatively prime
  - Minimise the largest modulus
  - Efficient in their binary representation: using n bits, are we representing almost 2^n numbers? $(5,3,2)$ and $(8,7)$
  - Compatibility with binary ALUs.
- Two systems:

  Optimal: minimizes the largest modulus

  "Binary": moduli of the form 2^{k_1}, $2^{k_1}-1$, $2^{k_2}-1$, $\ldots 2^{k_n}-1$ $(k_1, k_2, \ldots, k_n$ are integers)

Binary (or Merrill) RNS

Merrill suggests the largest be of the form 2^{k_1} and the second largest of the form $2^{k_1}-1$, k_1 the same.
The remaining moduli should avoid common factors.

Moduli	Prime Factors
3	—
7	—
15	3,5
31	—
63	3,7
127	—
255	3,5
511	7,73
1023	3,11,31
2047	23,89
4095	3,5,7,13
8191	—
2^k $(k = 1,2,3,4\ldots)$	2

Why those Merrill moduli?

- "Compatible" with binary ALUs.
- Almost the capacity of 2^n where $n = k_1 + \sum k_i$.

Bits to represent	Moduli set
17	32, 31, 15, 7
25	128, 127, 63, 31
28	256, 255, 127, 31

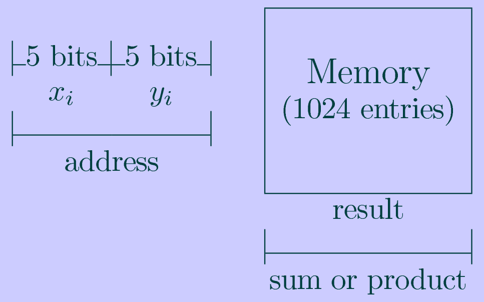
In the 17-bit case, instead of 2^{17} code points, we have

$$2^5(2^5-1)(2^4-1)(2^3-1) = 2^{17} - \mathcal{O}(2^{14}).$$

That is less than 1 bit of representational capability.

Operations with General Moduli

Example 3 A table of 1024 or 2^{10} entries is used for moduli up to 32, or 2^5; i.e., if x_i and y_i are 5-bit arguments, then their concatenated 10-bit value forms an address into a table of results.



In this case, addition, subtraction, and multiplication are accomplished in one access time to the table.

Conversion to RNS with general moduli

Let us compute the residue $\mathbf{mod_7}$ of the radix 10 integer 1826.
We begin by decomposing the number

$$\begin{aligned} 1826 &= 1 \times 1000 + 8 \times 100 + 2 \times 10 + 6 \\ &= a_3 \times 10^3 + a_2 \times 10^2 + a_1 \times 10 + a_0 \end{aligned}$$

and note that

$$\begin{aligned} 10\mathbf{mod_7} &= 3 \\ 100\mathbf{mod_7} &= (10\mathbf{mod_7} \times 10\mathbf{mod_7})\mathbf{mod_7} = 2 \\ 1000\mathbf{mod_7} &= (100\mathbf{mod_7} \times 10\mathbf{mod_7})\mathbf{mod_7} = 6. \end{aligned}$$

Thus, we have the following table and get

$$1826\mathbf{mod_7} = (6+2+6+6)\mathbf{mod_7} = 6.$$

a_3	x_{j3}	a_2	x_{j2}	a_1	x_{j1}	a_0	x_{j0}
0	0	0	0	0	0	0	0
1	6	1	2	1	3	1	1
2	5	2	4	2	6	2	2
3	4	3	6	3	2	3	3
4	3	4	1	4	5	4	4
5	2	5	3	5	1	5	5
6	1	6	5	6	4	6	6
7	0	7	0	7	0	7	0
8	6	8	2	8	3	8	1
9	5	9	4	9	6	9	2

Conversion to RNS with “Binary” moduli

A binary number $X\mathbf{mod_{2^n}}$ with the value

$$X_{base2} = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0,$$

where x_i has value 0 or 1, is rewritten as:

$$X_{base2^k} = X_{m-1}\left(2^k\right)^{m-1} + X_{m-2}\left(2^k\right)^{m-2} + \cdots + X_0,$$

where X_i has values $\{0, 1, \ldots 2^k - 1\}$.

Then, $X\mathbf{mod_{2^k}} = X_0$ and

$$X\mathbf{mod_{2^k-1}} = \left(\sum_{i=0}^{m-1} X_i (2^k)^i \mathbf{mod_{2^k-1}}\right) \mathbf{mod_{2^k-1}}.$$

From RNS back to conventional binary

1. Find the *weight* of modulus m_j: the residue representation that has a “1” in the j^{th} residue position and zero for all other residues.
2. To recover the integer X from its residue representation, we sum the weighted residue modulo M:

$$X\mathbf{mod_M} = \left(\sum (x_j w_j)\right) \mathbf{mod_M}.$$

Can we always use RNS?

The difficulties in using a residue number system are:

1. the long conversion times,
2. the complexity of number comparisons,
3. the difficulty of overflow detection, and
4. the indirect division process.

However, in an algorithm that relies heavily on addition and multiplication and does not need to convert or compare often (as in signal processing and cryptography) residues have their place!

Other uses: Error checking

If, in an n-bit binary system:

$$\begin{array}{r} a\mathbf{mod_{2^n}} \\ +b\mathbf{mod_{2^n}} \\ \hline c\mathbf{mod_{2^n}} \end{array}$$

then it also follows that:

$$\begin{array}{r} a\mathbf{mod_{2^k-1}} \\ +b\mathbf{mod_{2^k-1}} \\ \hline c\mathbf{mod_{2^k-1}} \end{array}$$

Since 2^n and $2^k - 1$ are relatively prime, it is possible to use a small k-bit adder ($n \gg k$) to check the operation of the n-bit adder.

Coming soon

- Minimizing the largest modulus leads to a shorter carry chain and a shorter delay $\rightarrow$ “optimal” moduli selection.
- Bounds on the time to add and multiply.
- Modeling memories, multiplexers, and shifters.