A Number System Approach for Adder Topologies

Álvaro Vázquez

REPSOL-ITMATI

Technological Institute For Industrial Mathematics

Elisardo Antelo

University of Santiago de Compostela

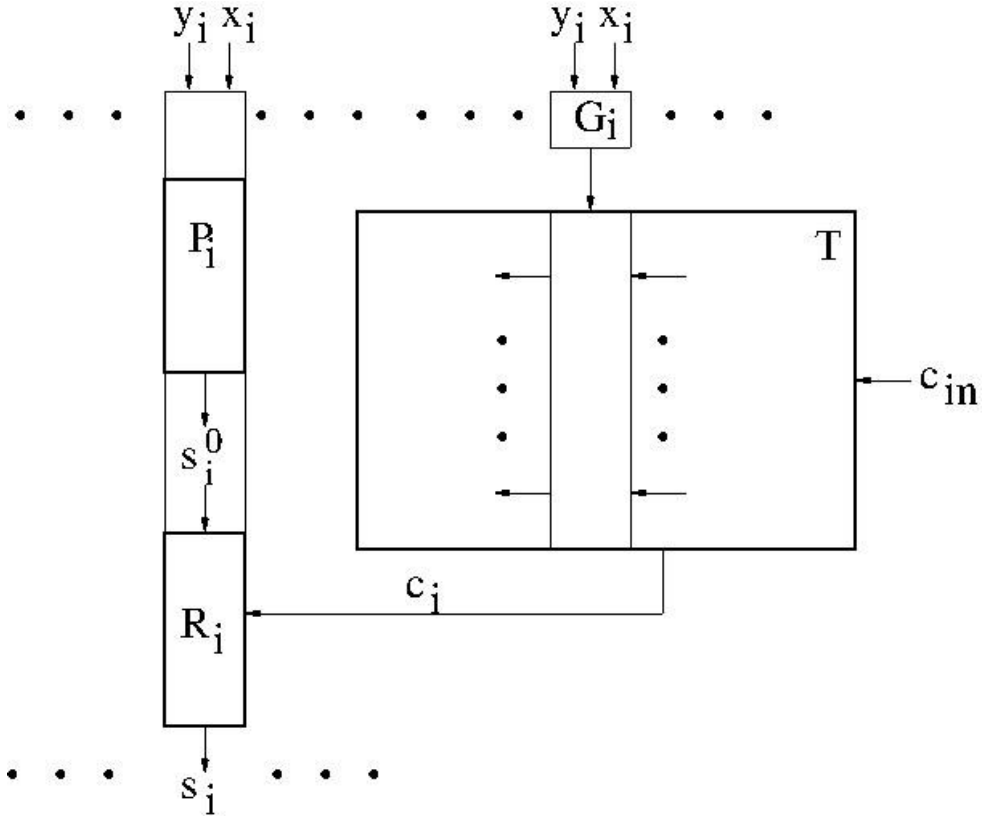
Spain

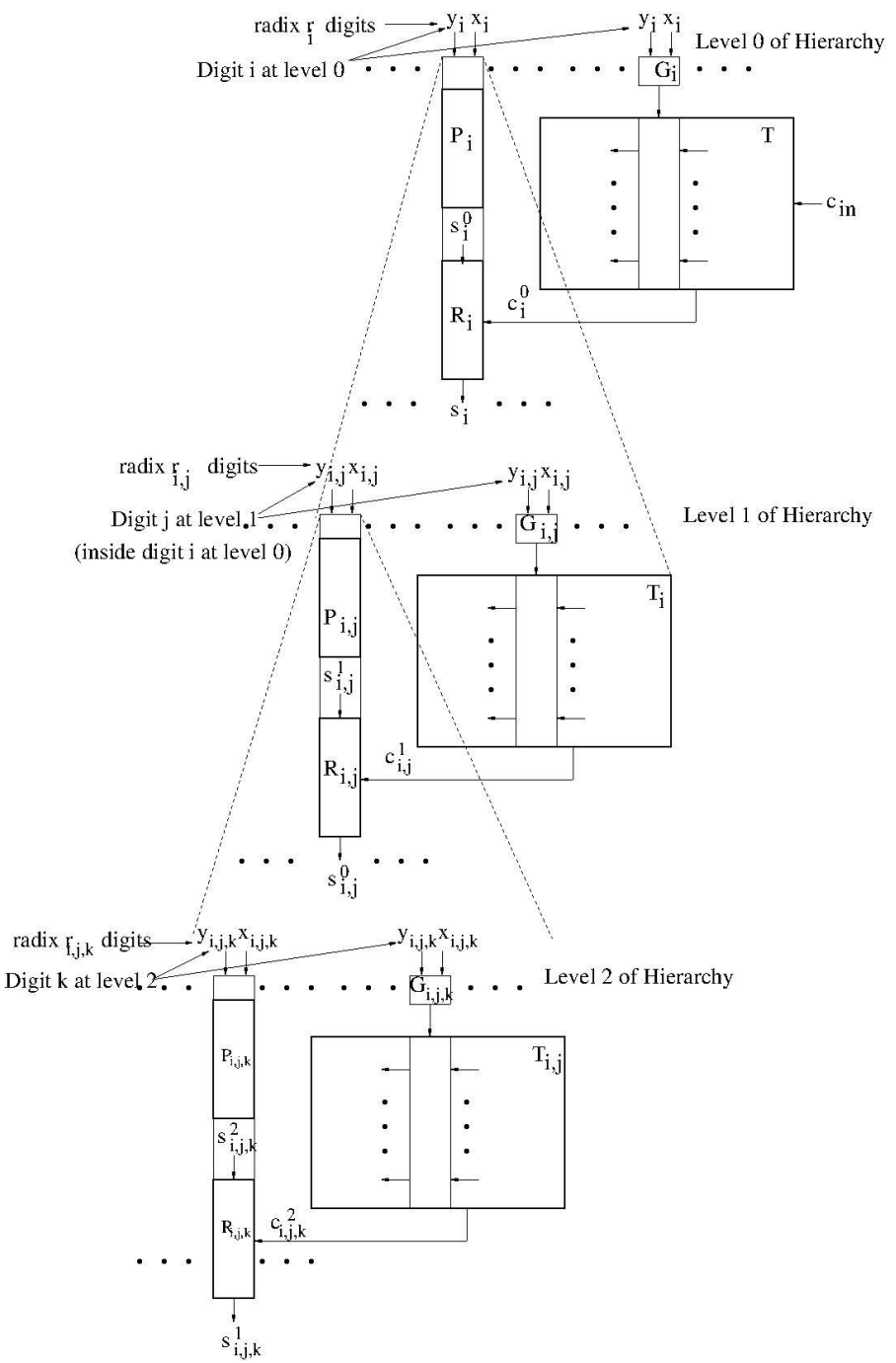
Introduction

- Mathematical Foundation behind the **design space of adders**.

- Significant role that **Number systems** play in the topology of adders.

Hierarchical Adder Model





Carry Computation

$g = xy$ $\quad a = x + y$ Generate and Alive signals

$$(g, a) \circ (g', a') = (g + ag', aa')$$

Group Generate Alive Signals

$$(g_{i,j}, a_{i,j}) = (g_i, a_i) \circ (g_{i-1}, a_{i-1}) \circ \ldots\ldots (g_{j+1}, a_{j+1}) \circ (g_j, a_j)$$

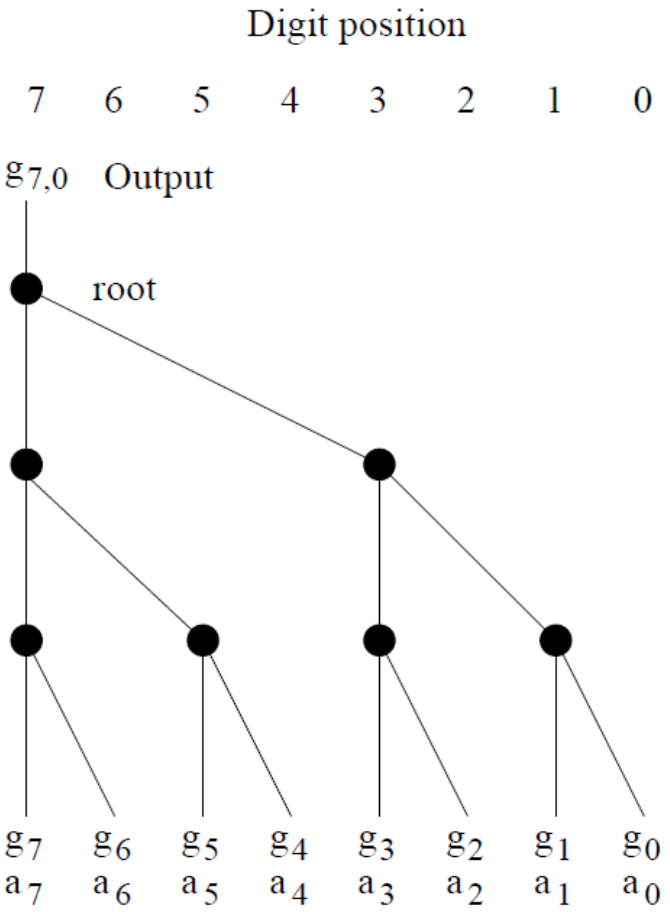
$$c_{i+1} = g_{i,j} + a_{i,j}\, c_j$$

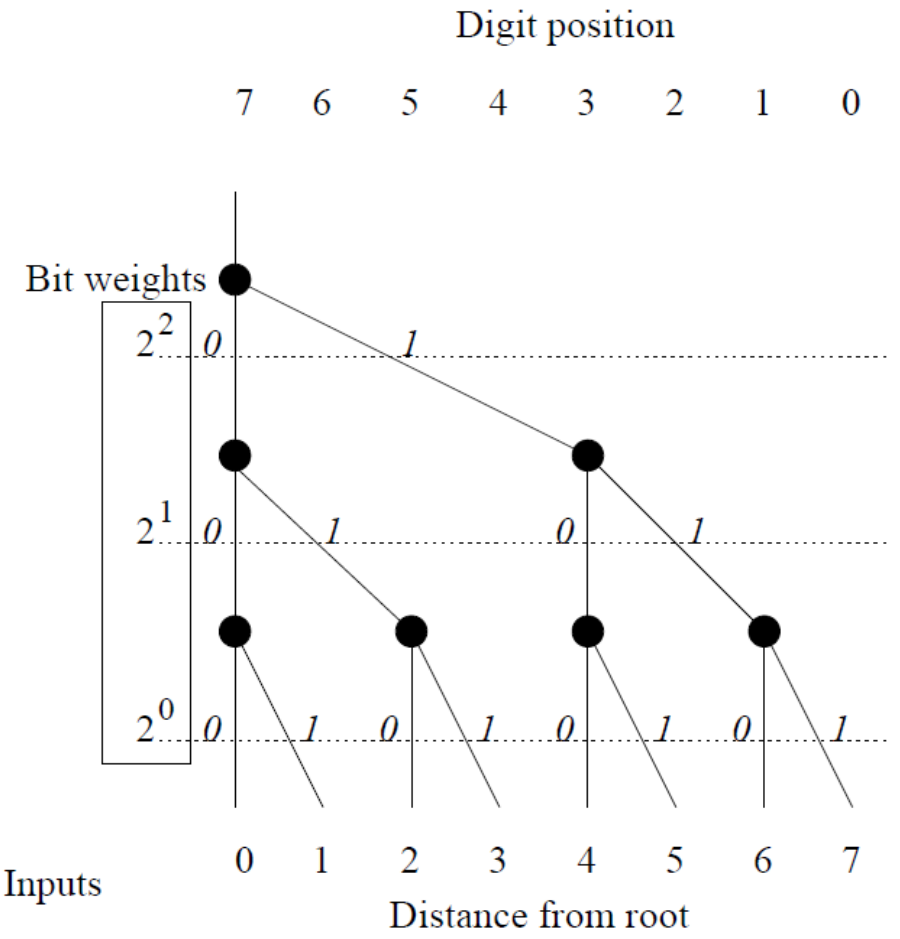
$$(c_{i+1}, a_{i,0}) = (g_i, a_i) \circ \ldots\ldots \circ (g_1, a_1) \circ (c_{inp}, 0)$$

Carry to postion i from input carry (gi,0) (cinp=g0)

$$(g_{i,0}, a_{i,0}) = (g_i, a_i) \circ \ldots\ldots \circ (g_1, a_1) \circ (g_0, a0)$$

Carry trees and Number Systems





digit2

digit1

digit0

Fundamental Question!

Necessary and sufficient conditions that must satisfy a **number system of distances** so that its corresponding tree performs a **correct carry computation**?

- Lets go with Number Systems:

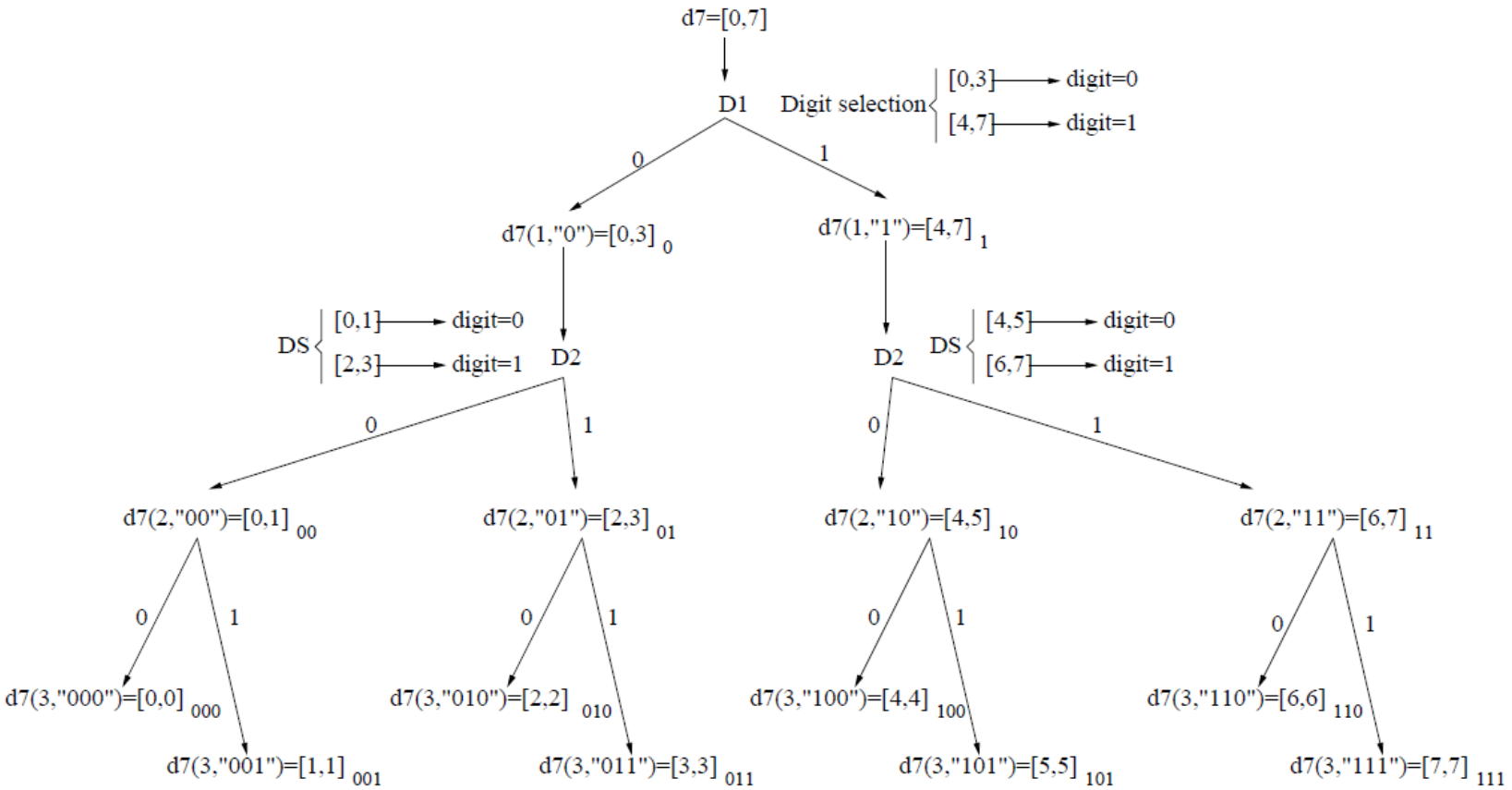
- **Important:** Interval of continous digit positions **[i,0] should map** to an interval of continous distances from the root **[0,i].**

- **Decimation**: Given a Number System (Digit set and Weights) → Obtain digits to represent each integer in **[0,i]**.

Example of Decimation [0,7]

2^2 a+2b+c

g7,0



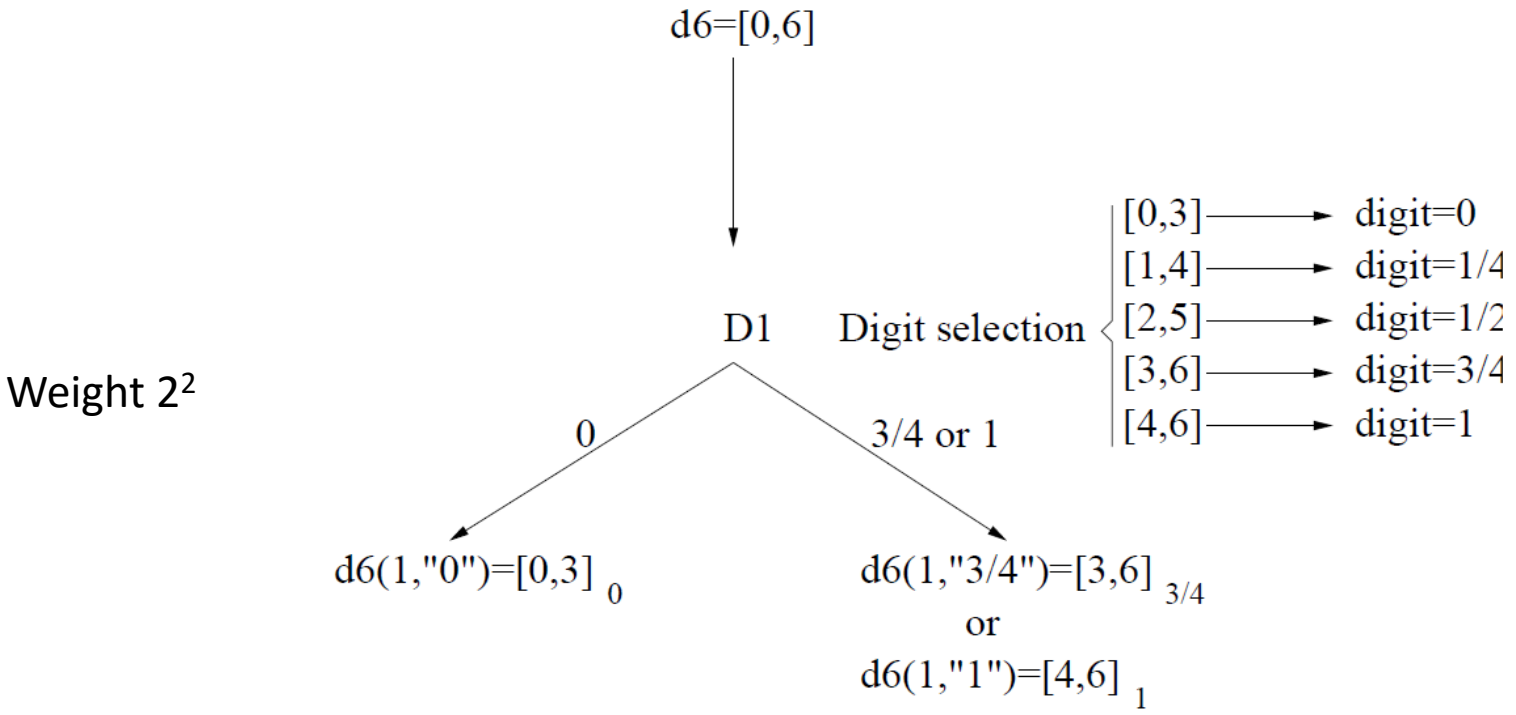
g7 *g6* *g5* *g4* *g3* *g2* *g1* *g0*

Tree for computing g7,0 can be obtained directly from this graph

Interesting Number System

- Weights= 2^j
- Digit Set: $0, 1/2^j, 2/2^j, \ldots, (2^j - 1)/2^j, 1$.
- Value of Digit j: $0, 1, 2, \ldots, (2^j - 1), 2^j$.
- Example: number of digits b=3
  - Digit 0: weight=1 digit set: 0,1.
  - Digit 1: weight=2 digit set: 0, 1/2, 1.
  - Digit 2: weight=4 digit set: 0, 1/4, 1/2, 3/4,1.
  - Digit 0: value ->0,1.
  - Digit 1: value ->0,1,2.
  - Digit 2: value ->0,1,2,3,4.

Example for First Step of Decimation



Digit 3/4 is possible since it is multiplied by weight 4 (2 = int(levels=log(3))

Number system (as a polynomial): x2 2^2 + x1 2^1 + x0 2^0, Digits for x2 (0,3/4) or (0,1).

In this case there may be overlap of intervals of dinstances from the root [0,3] and [3,6]

Necessary and Sufficient Conditions

- The number sytem should allow the computation of

$$(c_{i+1}, a_{i+1,0}) = (g_i, a_i) \circ \ldots\ldots \circ (g_{1,} a_1) \circ (c_{inp}, 0)$$

- This requires every term (g,a) to be present, and preserve the order of evaluation.

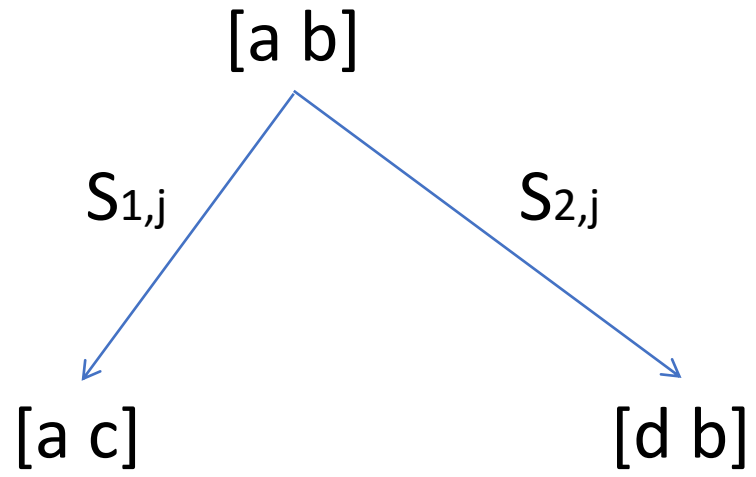
- Then the number system should allow the representation of [0,i]. with no interval.

- Conditions for decimation:
  - decimation of [a,b]
  - resulting intervals [a,c] and [d,b].
  - previous conditions are verified if during decimation c>=d-1.

Formal Definition of Binary Decimation

mv -> máximum value of digit
Assume positive digits

$$L_j = \sum_{v=0}^{j-1} m_v 2^v$$



Decimation of an interval

Digit selection: Interval $[a+2^j\ s\ ,\ a+2^j\ s + Lj\ ]$ -> digit =s

$a+2^j\ s \le a$ ⇒ $S_{1,j}=0$

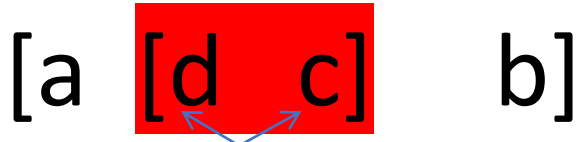
$a+2^j\ s + Lj \ge b$

$2^j\ s_{2,j} + Lj \ge b - a$

Continuity of intervals $Lj \ge 2^j\ s_{2,j} - 1$ ← Conditions for $s_{2,j}$

Formal Definition of Binary Decimation

Overlap of Intervals [a c] and [d b]



$[a,\ a+\min\{Lj\ , 2^{j} - 1\}]$ $[\max\{a+ 2^{j}\ s_{2,j}\ ,\ b-2^{j}+1\},\ b]$

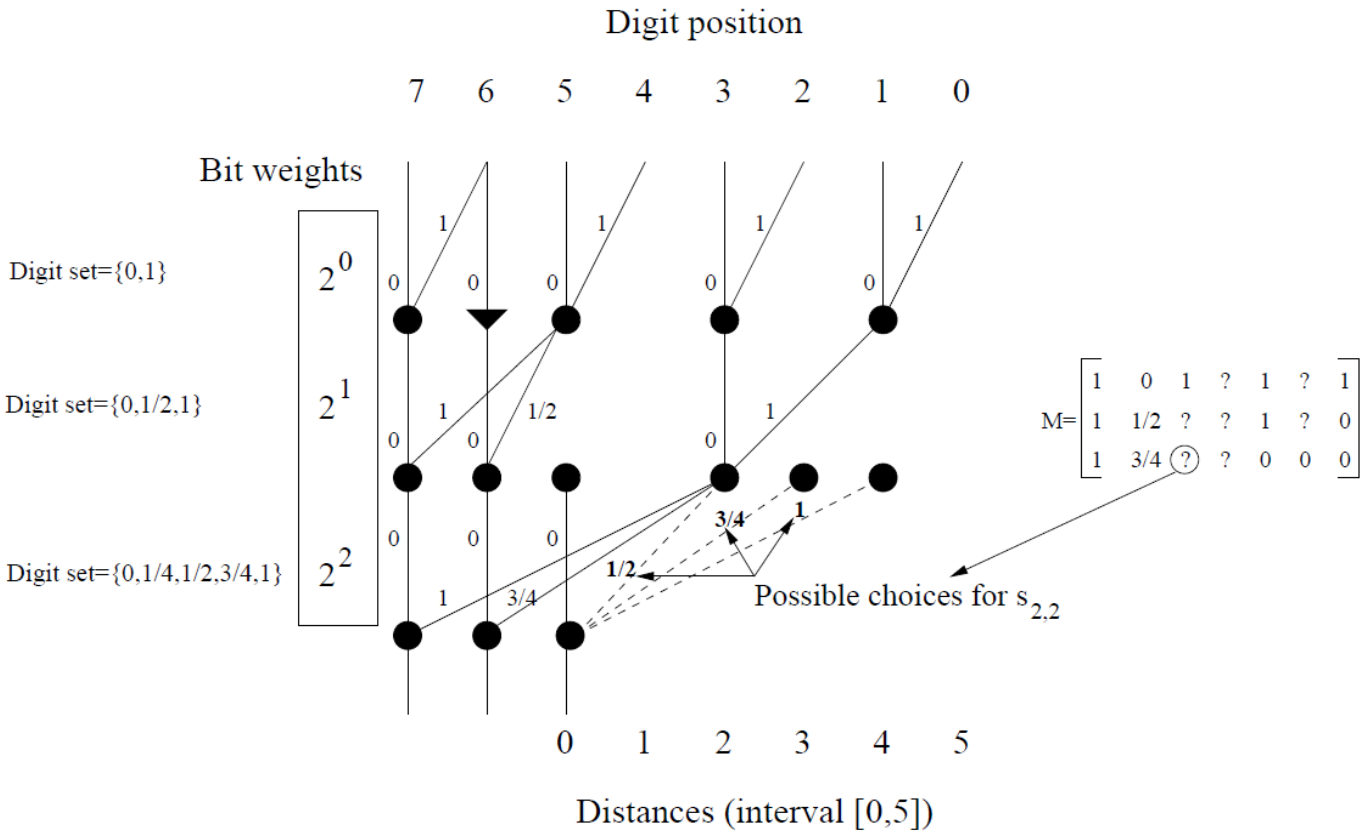
$\rho = \min\{Lj, 2j\ -1\} - \max\{2^{j}\ s_{2,j}\ ,\ b-a-2^{j}+1\}$

Overlap

Design of Module T

```
Define Number System: S.
for(i=0 to N-1) { /* For each tree */
  Nd=1; /* Number of intervals to be decimated at each level */
  for(j=⌈log2(N − i)⌉ − 1 to 0) { /* for each level */
    Next_Nd=0;
     for(k=0 to Nd-1) { /* for every interval at level j */
          Basic_Decimation_Step {
            Input_interval: [a, b] (remainder interval [0, h = b − a])
            Def._Sel_Remainder_Intervals: [2^j s, 2^j s + L_j] → digit=s
            Determine_Valid_Digits(); /* Possible values of s_{2,j} */
            s_{2,j}=M_{j,i+a} if M_{j,i+a} ≠ 0 else Select_Digit();
            /* Select taking into account previous decimations
            /* and other design criteria */
            M_{j,i+a} = s_{2,j}; /* matrix update */
            Obtain_Two_Remainder_Intervals();
           /* Intervals [0, min{L_j, 2^j − 1}] and
            [0, max{h − 2^j s_{2,j}, 2^j − 1}] */
           /* Adjust the intervals to the desired redundancy */
            Reduce_Redundancy(ρ,f)
            Check_fanout(flag); /* if flag=0, everything is ok */
            if (flag=1) { Restore();
                 /* Restore old M_{j,i+a} and set s_{2,j} not valid */
                 Go to Determine_Valid_Digits();
                 /* repeat selection process */ }
            Next_Nd=Next_Nd+ no_of_new_intervals;
              }
     }
     Nd=Next_Nd;
}}}
```

Example of a Process of Making a Desing Using Our Method



Some Cases of Interest for Number System Sr

$S_{2,j} = 1 - \lambda / 2^j \quad (3)$

λ is the bit complement of the k least significant bits of N-(i+a) at level j (for gi,0, input interval [a b]).

Adders proosed in [4] $\rightarrow$ Mask for selection using (3)
$[1,1,1,1] \rightarrow [0,0,0,0]$; $[2,1,1,1] \rightarrow [1,0,0,0]$; $[2,2,1,1] \rightarrow [1,1,0,0]$; $[2,2,2,1] \rightarrow [1,1,1,1]$; $[4,1,1,1] \rightarrow [2,0,0,0]$; $[4,2,1,1] \rightarrow [2,1,0,0]$; $[4,2,2,1] \rightarrow [2,1,1,1]$; $[4,4,1,1] \rightarrow [2,2,0,0]$; $[4,4,2,1] \rightarrow [2,2,1,1]$; $[8,1,1,1] \rightarrow [3,0,0,0]$; $[8,2,1,1] \rightarrow [3,1,0,0]$; $[8,2,2,1] \rightarrow [3,1,1,1]$; $[8,4,1,1] \rightarrow [3,2,0,0]$; $[8,4,2,1] \rightarrow [3,2,1,1]$

[4] Knowles Adders
and N=16

Relation with Burgess Adders [21]

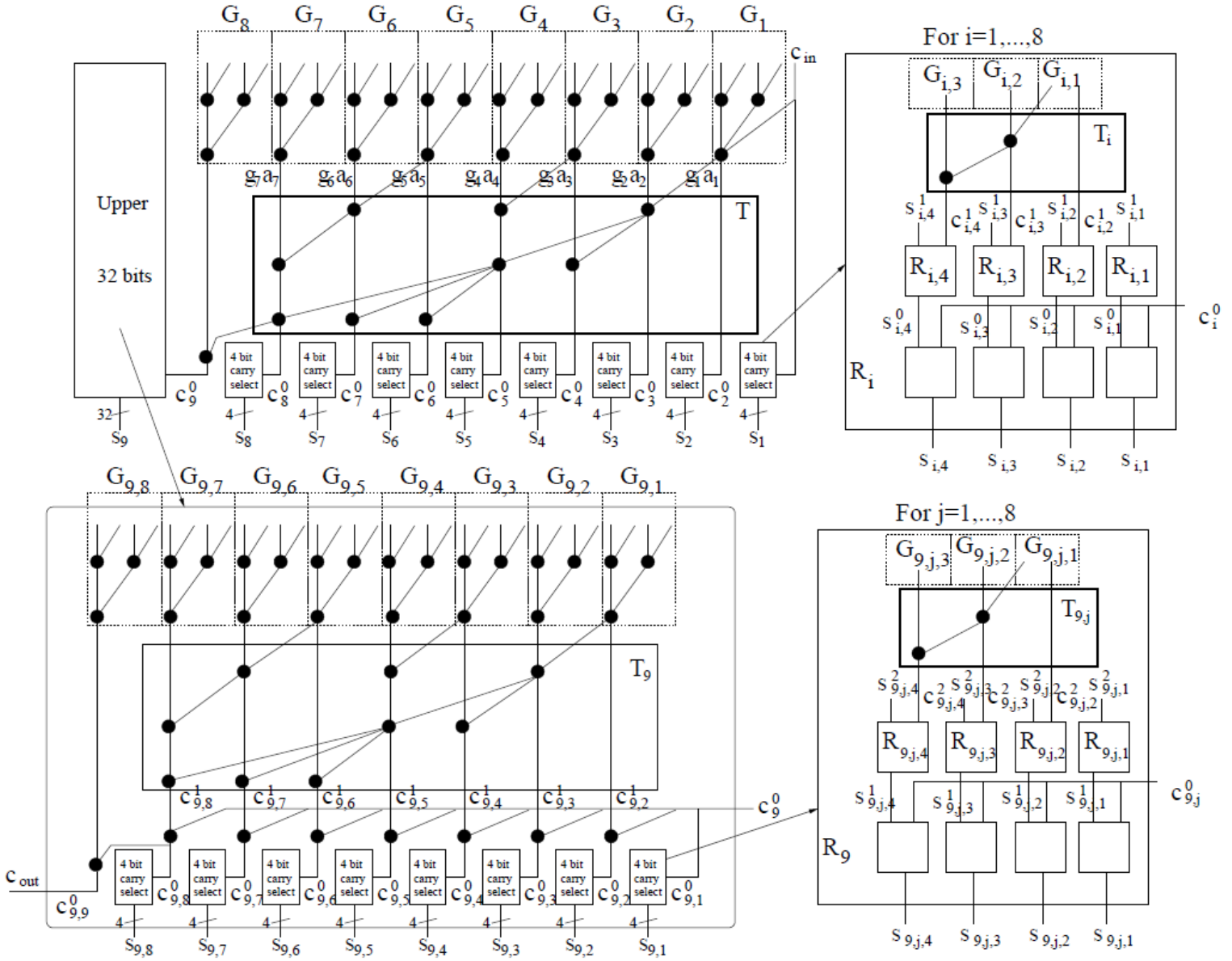
Span(j): introduced to determine whether idempotency is present in a prefix graph.

Relation to our work:

$$span(j) = \min\{s_{2,j}\} \times 2^j$$

$$s_{2,j} \neq 0$$

Number System Parameters of Intel Adder Architecture [20]



Number System Parameters of Intel Adder Architecture [20]

Digits and hierarchy levels	Level 0: $N = 9$; $r_i = 2^4$ $(1 \leq i \leq 8)$; $r_9 = 2^{32}$ Level 1: $r_{i,j} = 2$ $(1 \leq j \leq 4)$; $r_{9,j} = 2^4$ $(1 \leq j \leq 8)$ Level 2: $r_{9,j,k} = 2$ $(1 \leq k \leq 4)$;
T Modules Number System: S_r.	$MT = MT_9 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1/2 & 0 & 0 & 1 & 1/2 & 0 \\ 3/4 & 2/4 & 1/4 & 0 & 0 & 0 \end{bmatrix}$; $MT_i = MT_{9,j} = \begin{bmatrix} 0 & 1 \\ 1/2 & 0 \end{bmatrix}$
G Modules*	G_i: inputs $\rightarrow (g_{i,j}, a_{i,j})$ (bit level signals); Function $\rightarrow$ as a T module with matrix $\begin{bmatrix} 1 & 1 \\ 10 & \end{bmatrix}$ $G_{9,j}$: same as G_i; $G_{i,j}$: $(g_{i,j}, a_{i,j})$ pairs (bit level signals) $G_{9,j,k}$: $(g_{9,j,k}, a_{9,j,k})$ pairs (bit level signals)
Coding of carries	$c^1_{i,j} \rightarrow$ conditional carries; $c^0_i \rightarrow$ explicit carry; $c^2_{9,j,k} \rightarrow$ conditional carries; $c^0_{9,j} \rightarrow$ explicit carry
Coding of pseudosums	$s^1_{i,j} \rightarrow$ propagate signal; $s^0_{i,j} \rightarrow$ conditional pseudosums; $s^2_{9,j,k} \rightarrow$ propagate signal; $s^1_{i,j,k} \rightarrow$ conditional pseudosums

* $G1$ has a different design than the other G_i, using a 3–ary tree to accommodate the c_{in}.

Conclusions

Presented and adder model that allows design and specification on adders based on Number systems.

Explored the mathematical foundations behind the trees for carry computations.

We propose a method to design adders based on finding integer representations on a given number system.

We showed how our model is applied to many existing adder designs.

Our work is a step forward to the design of adders even at a higher algorithmic level than it was done up to this time.