

BEYOND BOOLE ALGEBRA WITH TI TMS320 DSP MULTIPROCESSING

G rard PINSON

Abstract

From information compression study, we assumed the following outlines : i) Transmitted informational state is a quantized, serial, local data. ii) Non-transmitted informational state is a continuous, concurrent, distributed pattern. iii) It is possible to define properties of a "Super-Boole" computer matching this two kinds of information processing, combining bits and forms, i.e. arithmetic calculus and Boole algebra principles with harmonic analysis and Lie algebra.

In fact, this principles point out quantum calculus. But in the face of hardware difficulties making quantum computer, and software difficulties (calculation time) simulating with boolean computer, we suggest, to study this systems, a circuit diagram with DSP processors. Feasibility evaluation is proposed from fixed-point TMS320Cxx.

Independance scale principle may be applied. This principle of calculus does not modify the shape of a form, only its amplitude. So normalized FFT F_1 is seen as NOT operator, convolution as OR, product as AND, normalized Dirac distribution $\delta_1(x)$ and $\mathbf{1}(x)$ as boolean \emptyset and 1 respectively.

Hardware applications like timing, computational speed, wiring are examined. For instance, boolean operation XOR here means a new DSP operation, where signal and spectrum are multiplied. It is a new device for programming.

From separability of M-dimension FT property, it is possible to construct arithmetic acting on M-dimension numbers whose "bits" are themselves signals of N-samples, with a very important new feature : there are two "zero", the real number "0" and the "super-boole" zero normalized distribution δ_1 . Other extensions of boolean calculus, as non-commutability and Lie algebra, modulation and modal logic, etc, are to be considered. The main advantage in relation to quantum computer is here it is extremely easy keeping coherence, with SYNC inputs.

Key words

DSP processors, Fourier Transform, Harmonic Analysis, Boole algebra, Lie Algebra, Modal Logic

INTRODUCTION

Compressing information embeded in a message is possible because there is a code which is previously set up between transmitter and receiver. A significant message includes an information content (selective and measurable content) that is really transmitted, and an identification content (structural content of the code, context, system of reference) that is not transmitted [1][2][3][4].

Transmitted information

Transmitted information measure is evaluated from discernable states of a source. So extending concepts from discrete to continuous models encounters difficulties :

$$H = -\sum_{i=1}^n p_i \log p_i \rightarrow H = -\int_a^b p(x) \log p(x) dx \quad (1)$$

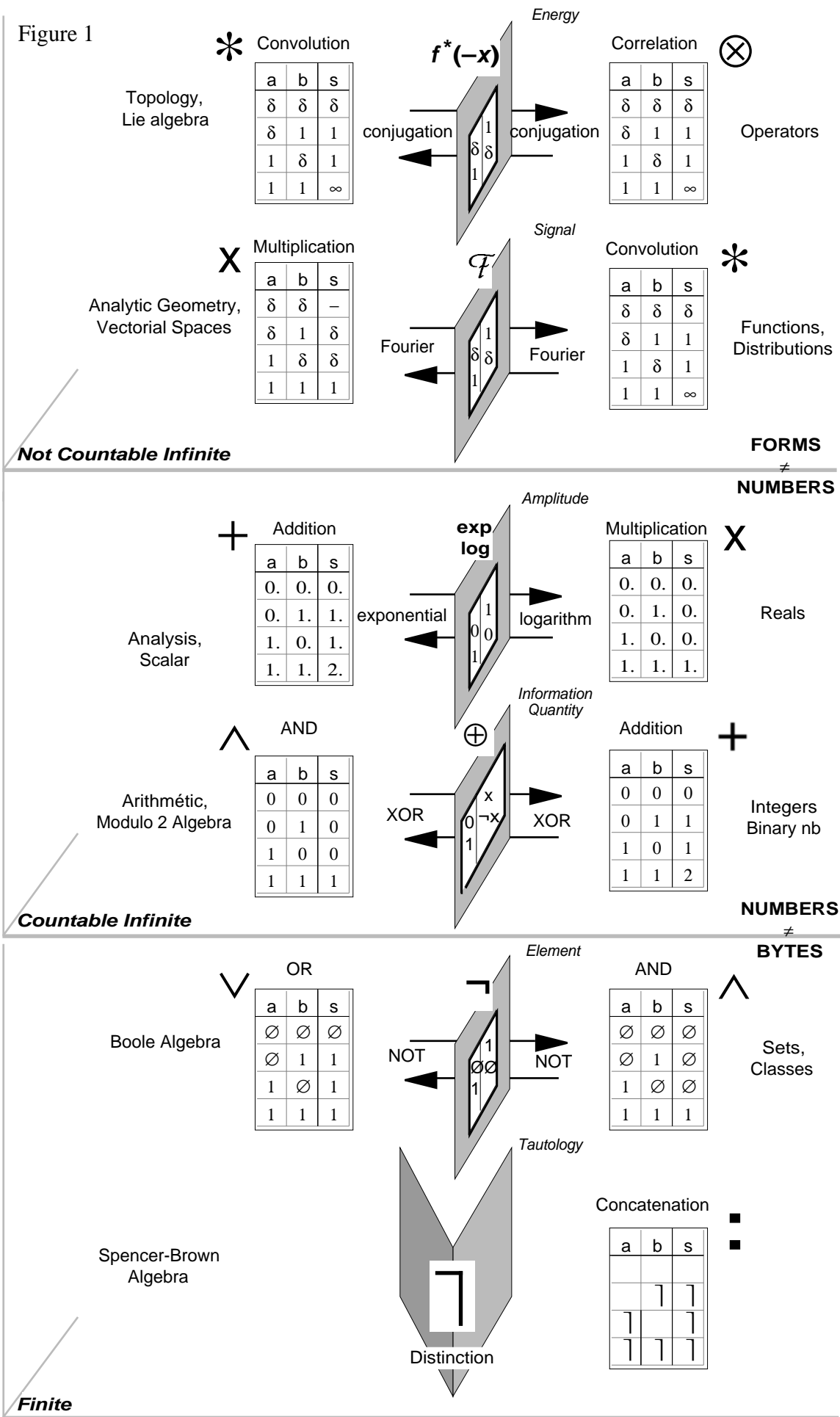
Entropy of a random variable with continuous distribution may be negative, may become infinitely large, and in contrast to the discrete case does not remain necessarily invariant under the transformation of the coordinate systems [5]. Two infinitely close states (dx) cannot be distinguished. It exists a smallest transmitted information quantity : YES/NO. Transmitted information is a quantized state based on a distinction operator, or indication [6], which is depending on resolving power.

One can show with calculus of quantized transmitted information that resolving power is linked to sampling interval [7]. This is in time domain a demonstration of Nyquist 's sampling theorem usually established in frequency domain :

$$2v_{\max} = \frac{1}{\Delta t} \quad (2)$$

Transmitted information is effectively transmitted, that is to say perceived, on condition that receiver quantizes to distinguish, and samples to quantize. Digital and sequential information computing is a matter for some logical and mathematical rulers lying in Boole algebra, modulo 2 arithmetic, integers arithmetic. Order relationship in \mathbf{N} allows univocal identification by counting transmitted samples.

On the other hand, distinction process being linked to receiver resolving power, if we accept



Teaching information processing

The former hierarchic diagram shows the strong coherence and the deep unity that exist between all the information processes, from boolean calculus to signal processing : fundamental concepts are the same along the whole diagram. For teaching this overview is a convenient way to introduce this topics with students. For instance convolution theorem corresponds to De Morgan relations (see tab. 1). The differences are also instructive : for instance, between $F[F[x(t)]] = x(-t)$ and $\neg\neg a = a$.

SOFTWARE APPLICATION

Specifications

- implementing intelligent calculation functions with signal processing,
- simulating boolean processes by DSP operators,
- extending boolean logic gates properties with DSP features.

Discrete implementation

Physical harmonic analysis of forms follows limitations : quantization, windowing, sampling, holding. This physical limitations are inevitable from the moment that one wants to implement a form, a non-transmitted informational state in real devices. So $\{\mathcal{F}, *, \delta, 1\}$ algebra is "digitalized", \mathcal{F} becomes Digital Fourier Transform (noted F), whose calculation is performed by FFT "butterfly" algorithm. In this case "distributions" multiplication becomes approximable.

This kind of limitations is well-known in the cas of boolean computers : it is impossible to compute infinite quantities, so the only arithmetic wich allows exact calculations is modulo 2 arithmetic. All the more so it's the same for calculations on real quantities or forms : hardware carrying out this operations is also necessarily approximate.

Normalization and NOT operator

FT et IFT (Inverse Fourier Transform) are two symmetrical operations. As $F[F[x(t)]] = x(-t)$, we have, for an even distribution like δ , a relation similar to logic relation $\neg\neg a = a$.

Unfortunately, this is not true in the digital case, because of 1/N factor that takes place in the IFT :

$$X(j) = \sum_{k=0}^{N-1} x(k) e^{-i2\pi \frac{jk}{N}} ; x(k) = \frac{1}{N} \sum_{j=0}^{N-1} X(j) e^{+i2\pi \frac{jk}{N}} \tag{7}$$

So, if we want that $\neg\neg 1(t) \equiv 1(t)$ et $\neg\neg \delta(t) \equiv \delta(t)$, we must translate non-transmitted operators by applying independance scale principle. So the main operator which is Fourier Transform becomes :

$$X_1(j) = F_1[x(k)] = \frac{F[x(k)]}{\sup|F[x(k)]|} \tag{8}$$

With this procedure, all resultant amplitudes are kept equal to one. But this principle of calculation does not modify the shape of forms. So normalized FFT F_1 is seen as a NOT operator : a "dirac" that is normalized to one is the inverse of $1(t)$ function, and conversely.

This static scaling must not be confused with dynamic scaling used in FFT computation to avoid arithmetic overflows. Here, the maximum of FFT result magnitude is calculated, and all samples are divided by this quantity, such the max result is equal to one.

To calculate $X_1(v) = F_1[x(t)]$ with equation (x), we can use the following algorithm :

```

Calculating FFT : X(v) = F[x(t)]
sup|X(v)| = 0
for i:=1 to N, do :
    calculating |X(vi)| = √[Re2(X(vi)) + Im2(X(vi))]
    if |X(vi)| > sup|X(v)| then
        sup|X(v)| = |X(vi)|
    end if
end do
for i:=1 to N, do :
    X1(vi) = Re(X(vi)) / sup|X(v)|
    X1(vi) = Im(X(vi)) / sup|X(v)|
end do
    
```

This normalization inserts a non-linearity in harmonic analysis, which loses some of its features so. This process is similar to a NOT logic gate, that is in fact a linear sign-changing amplifier whose saturation levels are held to 0 V and 5 V. So this NOT device loses some of its features as linear amplifier but keeps this basic one that is reversing.

Scaling and AND operator

To perform this algorithm, we could use a floating-point processor as 'C30. But it is less expensive to perform this calculation with a fixed-point processor.

In fixed-point format, the dynamic range is sufficient for our purpose. A 16-bit number can vary between -32,768 and +32,767. Q15 format is a programmer's convention : the location of the binary point affects neither the arithmetic unit, nor the multiplier in the DSP. It affects only the location from which the result will be read and has no relation to the hardware. To perform a NOT operator by FFT, and a AND operator by product, we need only the following rules (in two's complement) :

NOT : $F_1(\text{positive full scale}) \rightarrow \text{positive full scale}$
 AND : positive full scale X positive full scale \rightarrow
 positive full scale

such, if any sample keeps an amplitude equal to 0 or 1, the product result (AND) equals always 0 or 1 (as $1 \times 1 = 1$). For instance, it would be possible to consider the point between bits 14 and 13 (Q14 format), so the most positive number, which is equal to 1-2-15 in Q15 format, would be equal to 2-2-14. In this hypothesis, we have to limit positive numbers (i.e. scaling) to the value of one exactly. With such a convention, we lost 3 dB in dynamic range, but we no longer need floating-point calculation.

OR operator

As for OR operator, we can : either directly perform convolution (as circular convolution) ; either perform it applying convolution theorem, as a logic OR may be calculated with logic AND and NOT according to De Morgan theorem. To be exact, it would be necessary to extend vectors to 2N points, filling with zero segments between N+1 and 2N, to take windowing and aliasing into account. But if we limit us to vectors as $\delta_1(t)$, $\mathbf{1}(t)$ and normalized combs, we do not care of this here.

"Superbits" : Dirac's Combs

Sampling signals make periodic their spectrum, and conversely. So the above calculations are performed in fact with digital Dirac's combs, whose periods are equal to one for $\mathbf{1}(t)$ and N for $\delta_1(t)$. As Fourier Transform is an homothetic operator :

$$\mathcal{F}[x(at)] = \frac{1}{|a|} X\left(\frac{v}{a}\right) \tag{9}$$

we may consider combs whose periods equal a and 1/a. So a large enough value of N allows us full scope to choose superboolean \emptyset and 1 combs. It depends on the value of a only, which must equal a power of two between 1 and N/2.

Clearly, if superboolean \emptyset and 1 are N-point vectors, this variables will get new algebraic properties, that have not ordinary boolean variables.

"Superbytes" : M-dimension signals

Fourier Transform of a separable M-dimension function is separable. So a M-dimension FFT is calculated as (here $M = 2$) :

$$X(j_1, j_2) = \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} x(k_1, k_2) e^{-i2\pi \frac{k_1 j_1}{N_1}} e^{-i2\pi \frac{k_2 j_2}{N_2}} \tag{10}$$

Let $N_1 = N_2 = \dots = N$ (square matrix). Let signals : $\delta_1(k_1) \cdot \delta_1(k_2)$; $\delta_1(k_1) \cdot \mathbf{1}(k_2)$; $\mathbf{1}(k_1) \cdot \delta_1(k_2)$; $\mathbf{1}(k_1) \cdot \mathbf{1}(k_2)$. This signals are equivalent to bits series : $\emptyset\emptyset, \emptyset 1, 1\emptyset, 11$. We have (concatenation marked ".") :

$$\neg \{a:b\} = \neg a : \neg b \quad \Leftrightarrow \quad F1[x(k_1) \cdot y(k_2)] = F_1[x(k_1)] \cdot F_1[y(k_2)] \tag{11}$$

This relations extend to any number of "bits", and calculations on M-bit words are possible, such as classic logic does. Bit-by-bit OR, AND, XOR,... operations ensue. Furthermore, a half-adder would be designed like this. Let :

$$z = x + y \quad \text{with} \quad x, y \in \{ \delta_1(k_1), \mathbf{1}(k_1) \}$$

$$z \in \{ \delta_1(k_1) \cdot \delta_1(k_2), \delta_1(k_1) \cdot \mathbf{1}(k_2), \mathbf{1}(k_1) \cdot \delta_1(k_2), \mathbf{1}(k_1) \cdot \mathbf{1}(k_2) \}$$

Then :

$$s(k_1) = x(k_1) \oplus y(k_1) \quad s \in \{ \delta_1(k_1), \mathbf{1}(k_1) \}$$

$$c(k_2) = x(k_1) \wedge y(k_1) \quad c \in \{ \delta_1(k_2), \mathbf{1}(k_2) \} \tag{12}$$

This algorithm allows enumeration of integers, a N-value integer being in fact designed by a volume in M-dimension space, with $M \geq 1 + E(\log_2 N)$.

Timing

If we are just calculating with $\delta_1(t)$ and $\mathbf{1}(t)$ variables as noted above, clearly achieved results and properties are quite identical to classical logic. But we have now to consider translation in time of such signals like combs (whose sampling period is between 1 and N). This means to perform calculations with complex values. Algorithm could be executed in two steps : first, a proper calculus on complex amplitudes ; second, a "measure" process which gives the final data as a modulus. This process is equivalent to calculate light intensity from electromagnetic amplitude, or a quantum measure process. It is also evoking neural process, where nerve impulses are pulse groups whose phase is a kind of brain information coding [10].

HARDWARE APPLICATION

Note : at this point, we do not choose among TMS320 family. To evaluate feasibility and properties of super-Boole computer, we only need of an overview of key features of the TMS320 DSP processors, which are indicated by 'C10,..., 'C50.

Accuracy

First, from $N = 128$ to 1024 are usual sizes for discrete signals. Physical considerations such as indetermination principle, which is a consequence of Fourier Transform properties :

$$\Delta t \cdot \Delta \nu \geq \frac{1}{N} \quad (13)$$

mean enhancement of practical and theoretical result interest if N increases : sharp forms require great values of N to avoid aliasing. But indetermination principle means infinite accuracy that is impossible on a super-Boole computer, as N -bit length words of boolean computer are also limited.

Second, with a 2-point FFT ($N=2$), calculus operate on 0-phase and π -phase signals. Phase shift accuracy depends on size N of samples set : with an order of magnitude $N \approx 10^2$ (i.e. $N = 128, 256, \dots$), many cases may be considered.

Memory considerations

-Each super-Boole operator needs a buffer size of $2N$ words (complex data of N samples) for just one superbit. On an other hand, an on-chip memory is more useful, facing the complexity of the whole super-Boole computer.

For instance, 'C20 has 544 words of on-chip data RAM, organized in two 256-word blocks : this memory allows to implement a NOT operator with 256-sample signal and in-place FFT computation. Or it allows a 2-input OR, AND,... with 128-sample signal using 2-block RAM configuration of 'C20.

-Each super-Boole operator needs always the same program, to perform for instance a 2-input NAND. A macro implements a radix-2 DIT N -point FFT, with static scaling to ensure normalization (ranging in 0-1 magnitude) of superbits. With looped code, ROM matrix of TMS320 such 'C25 (4K words on masked ROM) holds easily FFT and normalization implementations.

Calculation time

A radix-2 128-point looped FFT needs 21,879 clock cycles and 4.375 ms execution time on 'C20 (with 5 Mhz clock) [11]. This features must be compared with ordinary boolean circuits, which perform boolean operations in some nanoseconds ! So there is an order

of 10^6 between time calculation of Boole and super-Boole computers. This is inherent to working principles of the latter. In return, it could present new properties that can only appear at this level of complexity, as we will see later.

An other possibility would be to choose special purpose DSP chips. This solution is interesting for a second step of this project, when features of super-Boole computers will be well known. But in a first time, general purpose DSP like TMS320 are a better approach to develop basic algorithms and to draw the simplest diagram.

Timing

One of this new features is to be able to compute with lead or lag superbits. So there is necessary to ensure a good synchronization inside the whole device to keep some phase constant throughout a calculation. For example, simulating an holographic operation, where the result is (hologram = NOR super-Boole op.) :

$$F[x*y] \quad \Leftrightarrow \quad \neg (a \vee b) \quad (14)$$

needs a sharp control of phase between x and y signals. This synchronicity does not appear on an ordinary boolean device, because a and b boolean values are supposed to be holded on input wires of NOR operator during the whole operation. So a timer (like a program counter) must control the calculation development on each supergate using HOLD inputs of TMS. A general RESET (for 'C50) or SYNC operation (for 'C25) before each set of superboolean calculations must be performed to control time origin.

This facilities which are encountered on such silicon devices are the main advantage in relation to quantum computers : it is extremely easy to keep coherence between components, even this ones are separated by a lot of intermediary steps. This is not the case on quantum devices — it is far from being so.

Multiprocessing

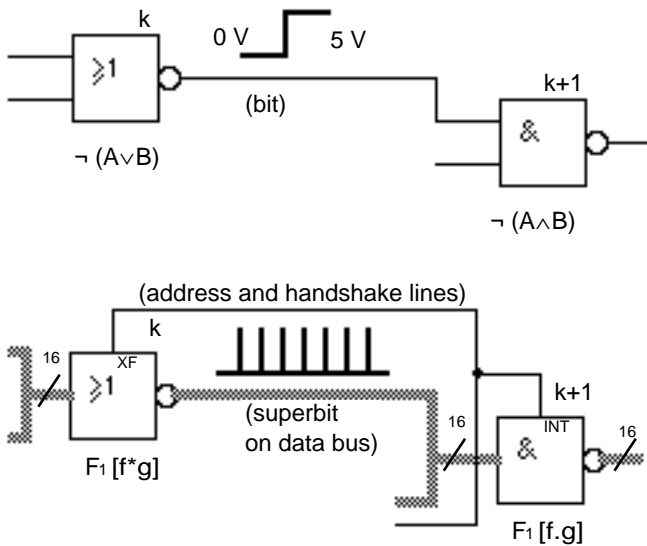
Performing an in-place FFT requires to know the whole set of samples of superbits, like an "offline" process (this is not the case for convolution, that is possible to perform "online"). So we have to transmit the samples from the supergate k to the next one $k+1$ as a "burst" mode as soon as computation by k -gate is ended.

Gates k and $k+1$ can be interfaced using several solutions. Interrupt (INT) and branch on I/O (BIO) are inputs that are provided from asynchronous sources within a system. The BIO input provides a convenient approach to implementing polled I/O. Two processors can also be synchronized by using BR and READY

signals. An other approach for TMS multiprocessing is a multiple master-slave configuration using direct access memory (Fig. 2) [12]. The master (k+1) requests data from the slaves (gates k, "providers"). After each slave's buses control and hold acknowledge signal (from HOLDA slave to BIO master), the slave's XF signal is wired to an INT input of the master : XF is used to indicate to the master when next data are available. Such the supergate k+1 may use up to 4 inputs controled by INT1,2,3,4 with 'C50.

Such a device is not only able to perform boolean calculus from input variables to output results, but also able to perform backward calculation from outputs to inputs : since data buses are bidirectionnal, coefficients (like adaptative filters) can be calculated from results to tune previous operations.

Figure 2 : Master-Slave TMS multiprocessing



XOR

The basic process making a programmed device with logic gates is XOR operator (exclusive-OR). Obviously, we could perform programmed operations by changing subroutines in TMS processors, for example to calculate an AND then an OR. But this is not the purpose of our project : super-Boole computer must not need any external programming to choose some operation : as an ordinary ALU, wich carries out good operations just by external signals on programming inputs.

So the basic function is a YES/NOT programmed operator, wich is performed as $c = a \oplus b$: if $a = 1$ then $c = b$ else $c = \neg b$ (Fig. 3).

The boolean operation XOR $a \oplus b = (a \wedge \neg b) \vee (\neg a \wedge b)$ here means a new DSP operation, where signal and spectrum are multiplied :

$$x \oplus y = x \cdot F_1(y) + F_1(x) \cdot y = x \cdot Y_1 + X_1 \cdot y \tag{15}$$

For example, y is a programming input, while x is the variable. If signals are basic superbits (normalized combs), the condition if... then... else... is performed exactly like a boolean calculation. But if the shape of y at logical state "1" is not exactly like a sampled $\mathbf{1}(u)$, with some samples changing about this value for instance, new conditions add to trigg the execution of the if...then...else condition.

Super-Boole ALU

At last, a whole logic device as 4-bit ALU SN74xx181 can be translated in super-Boole logic. Therefore, the internal diagram of such a device, with about thirty TMS320 interconnected, is to be reconsidered. If we examine the detailed logic diagram of a "simple" SN74xx283 4-bit adder , a basic solution is : each TMS320 takes place of each gate (a maximum of 4 inputs is necessary). But it would be possible to simplify this diagram, because TMS's are underused if they have to perform just a multiplication (AND operators). The internal bus structure of the 283 allows, with multiplexers and further addressing features, connecting some less processors.

RS flip-flop

The flip-flop RS operation means here a recursive calculation :

$$Q_{n+1} = S \vee (Q_n \wedge \neg R) \Rightarrow Z_{n+1} = x^*(Z_n \cdot F_1[y]) \tag{16}$$

This flowchart implies a polling algorithm so that RS is able to change state when a new data occurs on inputs R or S. This means a TMS320 would be performing this calculation at any time, a further handshaking being necessary between the flip-flop and the previous devices to take count of new data.

If the shape of f and g at logical states "Ø" or "1" are not exactly like sampled $\delta_1(u)$ and $\mathbf{1}(u)$, with some samples changing about the right shape, new conditions add to trigg the RS :

-first, limit of convolution of some signal (except $\delta_1, \mathbf{1}, \dots$) with itself is a gaussian (central limit theorem). In turn, because of aliasing, limit of sampled gaussian is $\mathbf{1}(u)$:

$$x \vee \left[x \vee \left[x \vee \left[x \vee \left[\dots \right] \right] \right] \right] \rightarrow e^{-\frac{t_k^2}{2\sigma^2}} \rightarrow \mathbf{1}(k) \tag{17}$$

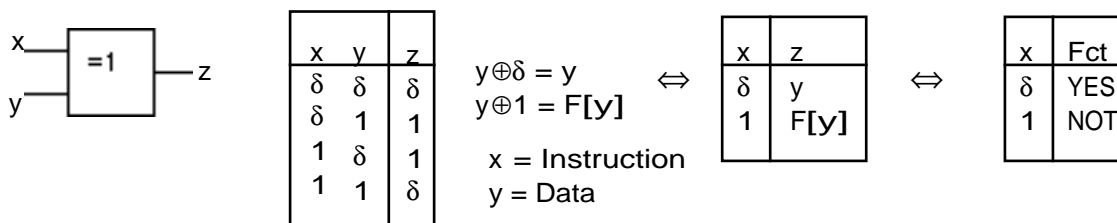
-second, limit of multiplication of a sample with itself is : either 1 if sample amplitude is equal to 1 (1 X 1 = 1) ; either 0 if sample amplitude is contained between 0 and 1 (0.x X 0.x X ... \rightarrow 0). So, for some signals, we have :

$$x \wedge \left[x \wedge \left[x \wedge \left[x \wedge \left[\dots \right] \right] \right] \right] \rightarrow \delta_1(k) \tag{18}$$

This means super-RS is acting like a filter. This is a stability property that, from noise signals, returns normalized combs and basic shapes $\delta_1(u)$ and $\mathbf{1}(u)$.

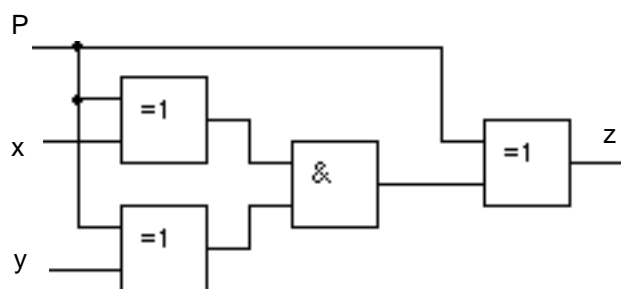
Figure 3 : 1-bit 4-instruction super-Boole ALU

- Simplest computer of the world !



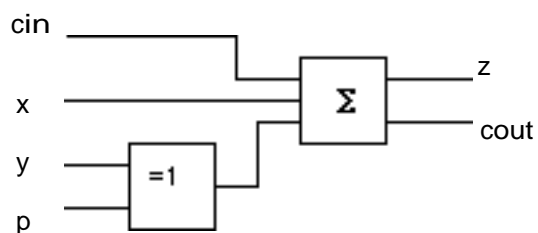
- Programmable Unit Logic

p	z	function
δ	x.y	AND
1	F[F[x].F[y]] = x*y	OR



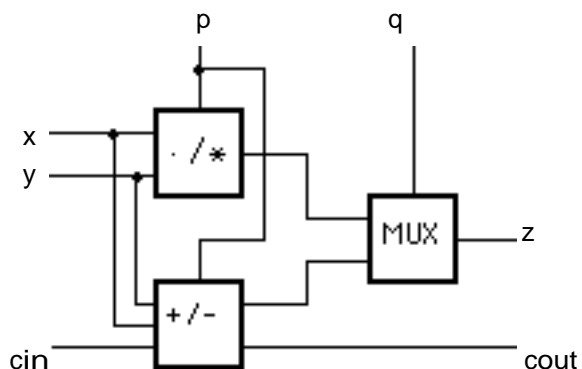
- Programmable Arithmetic Unit

p	cin	z	cout
δ	δ	x + y	cout
δ	1	x + y + 1	cout
1	δ	x - y - 1	F[cout]
1	1	x - y	F[cout]



- 1-bit ALU

q	p	z	cout
δ	δ	x * y	X
δ	1	x . y	X
1	δ	x + y + cin	cout
1	1	x - y - F[cin]	F[cout]



NEW FEATURES

Going beyond Boole algebra with DSP is possible because superbits are vectors on which it is possible to perform more powerful basic operations like Fourier Transform and convolution algebra. Moreover, DSP processors could compute several superimposed signals like combs of different frequencies and phases, so it would be possible to simulate the superposed states of a quantum computer operation. In this section, we shall examine briefly some simple features which could enhance the power of electronic calculus.

Symetry

Discrete Fourier Transform of a scalar (DFT on a single point) is this scalar itself :

$$F[k] = k \tag{19}$$

It means superbits are necessary vectors to be distinguished. But vectors imply to define the origine and the direction with wich they are defined. We will see the problem of origine latter and consider now the direction.

First, as we noted, $F[F[x(t)]] = x(-t)$: double negation is not equivalent to a simple NOT operator. We do not care of this fact if function $x(t)$ is even. But in the general case, we have to consider hermitian symetry, which transforms convolution in correlation :

$$x(t) \rightarrow x^*(-t) \Rightarrow \begin{aligned} x(t) \otimes y(t) &= x(t) * y^*(-t) \\ x(t) \otimes y^*(-t) &= x(t) * y(t) \end{aligned} \tag{20}$$

Convolution and correlation has two distinct algebraic structures [13], as convolution is a commutative and associative operation, while correlation is not. To study the two algebras noted A^\bullet (avec $\bullet = * \text{ ou } \otimes$) let the terms :

$$[x,y]^\bullet = x \bullet y - y \bullet x \quad (\text{commutator}) \tag{21}$$

$$[f,g,h]^\bullet = f \bullet (g \bullet h) - (f \bullet g) \bullet h \quad (\text{associator}) \tag{22}$$

It is easy to show that :

$$[f,g]^* = 0 \quad [f,g,h]^* = 0 \tag{23}$$

$$[f,g]^\otimes \neq 0 \quad [f,g,h]^\otimes \neq 0 \tag{24}$$

This relations characterize A^* as an abelian algebra (as Boole algebra), while one shows that :

$$[f,f]^\otimes = 0 \tag{25}$$

$$[[f,g]^\otimes, h]^\otimes + [[g,h]^\otimes, f]^\otimes + [[h,f]^\otimes, g]^\otimes = 0 \tag{26}$$

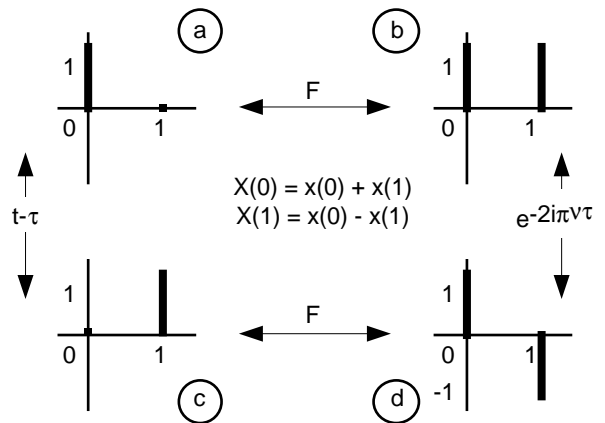
This relations characterize A^\otimes as a Lie algebra.

On a limited integration domain (the case of physical signals and filters), *start* of convolution is calculated summing product of *start* of x form by *start* of y form (Fig. 4). This calculation can be performed in place between input data and system pulse response, as instantaneous coding. While *start* of correlation is calculated somming product of *start* of x form by *end* of y form or, symmetrical but not commutative calculation, somming product of *start* of y form by *end* of x form. So, it is not any more equivalent calculating a $\vee b$ and $b \vee a$. Information direction becomes significant.

Phase

An other feature of super-Boole calculus concerns the definition of origine from which samples are repaired. Consider for exemple a 2-point FFT whose signals are $a = \delta_1(k)$ and $b = 1(k)$. In this conditions, FFT is :

Figure 5 : $x \rightarrow y$



$$\begin{aligned} X(0) &= x(0) + x(1) \\ X(1) &= x(0) - x(1) \end{aligned} \tag{27}$$

Consider c which is the translated of a. With 2-point signals, this corresponds to a phase difference of π (Fig. 5).

If we calculate basic superboolean operators from a and b, we obtain the following results :

Table 2 : 2-point FFT with lead/lag signals

x y	x AND y	x OR y	x → y
a a	a	a	b
a b	a	b	b
b a	a	b	a
b b	b	b	b
<hr/>			
c c	c	a	-d
c b	c	b	0
b c	c	b	c
b b	b	b	b

For AND and OR 2-point supergates, results differ not much from ordinary logic calculations. As we noted former, we can calculate the amplitude of the result (as a measure process), so AND and OR are strictly the same. But there is not the case for logical implication. Consider the arithmetic zero (marked 0) means a null condition, something which is not significant — a "NAM" (Not-A-Meaning) like it exists "NAN" (Not-A-Number) in high level programming languages. Then, something which is false cannot imply a true consequence, contrary to formal logic. Just a non-sens.

Modulation

After symetry and phase, a third possibility of changing and improving the basic logic is to modulate or to translate the superbits. Then, the following diagram show that there is a common structure between Fourier calculus and modal logic :

$$F[x(t-\tau)] = e^{-2i\pi v\tau} \cdot F[x(t)] \quad \neg\Box p = \Diamond\neg p \quad (28)$$

$$F[e^{-2i\pi at} \cdot x(t)] = X(v-a) \quad \neg\Diamond p = \Box\neg p \quad (29)$$

$$x(t-\tau) = F[e^{-2i\pi vt} \cdot X(v)] \quad \Box p = \neg\Diamond\neg p \quad (30)$$

$$e^{-2i\pi at} \cdot x(t) = F[X(v-a)] \quad \Diamond p = \neg\Box\neg p \quad (31)$$

For instance, such a modal scheme can be interpreted by \forall and \exists quantifiers (Fig. 6). With $x(t) = \delta_1(t)$, $\mathbf{1}(t)$ it is possible to show that :

$$\Box p \rightarrow p \iff |F_1[x(t-\tau) \cdot F_1[x(t)]]| = 1 \quad (32)$$

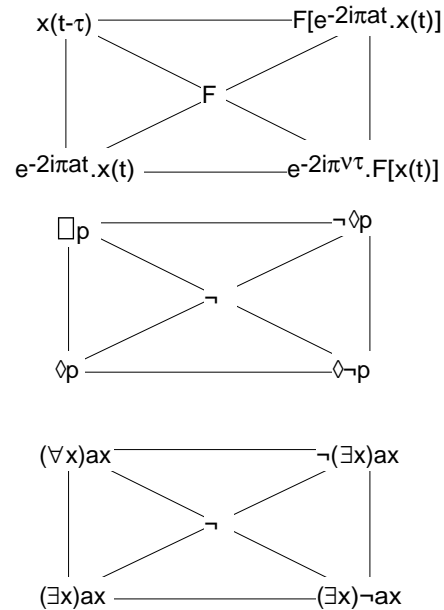
$$p \rightarrow \Diamond p \iff |F_1[x(t) \cdot F_1[e^{-2i\pi at} \cdot x(t)]]| = 1 \quad (33)$$

are true (a complete discussion of this topics is in preparation)

This means it would be possible to go beyond classical logic calculus, introducing a modal logic nearer thought features than boolean calculus. A computer designed with DSP TMS320 processors performing intelligent calculations from the underlying

hardware level, could be powerful to solve some challenges of automatic reasoning.

Figure 6 : modulation and modal logic



REFERENCES

- [1] PINSON G. : Une théorie cognitive de l'information, *Rev. Int. Systémique*, vol 9, n°1, 1995, p 27-66 and vol 9, n°5, 1995, p 541-544.
- [2] PINSON G. : Cognitive Information Theory, *14th Intern. Congress on Cybernetics*, Namur, Belgium, 1995
- [3] MACKAY D. M. : *Information, Mechanism and Meaning*, MIT Press, Cambridge, Mass. 1969.
- [4] DEMAN P. : La compression d'information à la division télécommunications, *Rev. tech. Thomson-CSF*, vol 7, n°4, dec. 1975, p 723-730.
- [5] REZA F. M. : *An Introduction to Information Theory*, McGraw-Hill, New-York, 1961.
- [6] SPENCER-BROWN G. : *Laws of form*, Dutton, New-York, 1972.
- [7] OSWALD J. : *Théorie de l'information ou Analyse diacritique des systèmes*, Masson, Paris, 1986.
- [8] WOLPER P. : *Introduction à la calculabilité*, InterÉditions, Paris 1991.
- [9] COLOMBEAU J.-F. : *Multiplication of Distributions*, Springer-Verlag, Berlin, 1992.
- [10] EGGERMONT J. : *The correlative brain*, Springer-Verlag, Berlin, 1990.
- [11] PAPAMICHALIS P. : Implementation of Fast Fourier Transform Algorithms with the TMS32020, *Digital Signal Processing Applications with the TMS320 Family : Theory, Algorithms and Implementations*, vol. 1, Texas Instrument, 1989, p 69-168.
- [12] *TMS320C5x User's Guide*, Texas Instrument, 1993.
- [13] BORSELLINO A., POGGIO T. : Convolution and Correlation Algebras, *Kybernetik*, vol. 13, 1973, p 113-122.