

# An All-Optical Soliton FFT Computational Arrangement in the 3NLSE-Domain

Anastasios G. Bakaoukas<sup>(✉)</sup>

Computing and Immersive Technologies Department, University of Northampton,  
St. Georges Avenue, Northampton NN2 6JB, UK  
`Anastasios.Bakaoukas@northampton.ac.uk`

**Abstract.** In this paper an all-optical soliton method for calculating the FFT (Fast Fourier Transform) algorithm is presented. The method comes as an extension of the calculation methods (soliton gates) as they become possible in the Cubic Nonlinear Schrödinger Equation (3NLSE) domain, and provides a further proof of the computational abilities of the scheme. The method involves collisions entirely between first order solitons in optical fibers whose propagation evolution is described by the Cubic Nonlinear Schrödinger Equation. The main building block of the arrangement is the half-adder processor. Expanding around the half-adder processor, the “Butterfly” calculation process is demonstrated using first order solitons, leading eventually to the realisation of an equivalent to a full Radix-2 FFT calculation algorithm.

**Keywords:** Solitons · 3NLSE domain · All-optical FFT · Cubic Nonlinear Schrödinger Equation · Soliton collisions · Soliton computational schemes

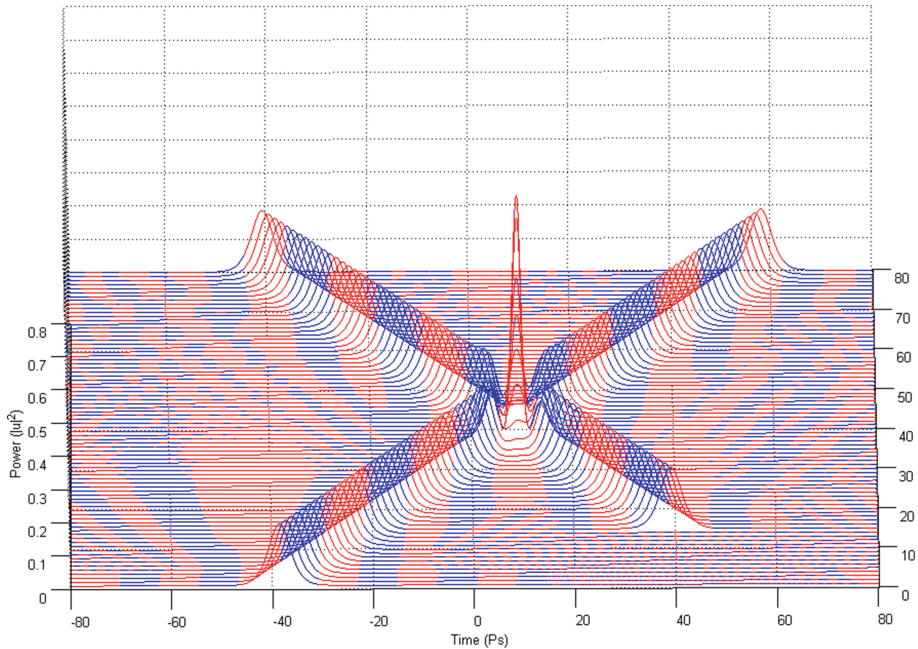
## 1 Introduction

There is a number of studies in which the use of soliton optical pulses for the purposes of carrying out computations has been investigated [1, 2]. In this present paper only temporal solitons (involving a balance between Kerr type nonlinearities and dispersive effects in glass fibres) are concerned. At this early point the fact that the interactions between solitons of this type can be a relatively long-range phenomenon need to be emphasised, because the Kerr nonlinearity is a relatively weak effect. Temporal solitons in optical fibres where the nonlinearity is of the Kerr type, are well described by the 3NLS Equation which, for very short (fs) pulses, requires corrections to account for “Higher Order Dispersion”, “Raman scattering” etc. If pulse widths are such that these higher order effects can be neglected, then solitons in optical fibres, are solutions of the integrable nonlinear Schrödinger equation and since collisions between fibre solitons are elastic they were not previously considered to be capable of useful computation [2].

In what follows in this introduction section, a brief description of the background theory is presented for the benefit of the reader. For a more extensive and

thorough discussion the reader is referred to [3,4] where the application of first order and second order solitons, following the Toffoli gates prototype as well as others, has been presented and verified regarding their computational abilities in terms of logic gates formations.

When higher order dispersive and nonlinear effects are neglected, short pulse propagation in nonlinear optical guides is described by the integrable Cubic Non-Linear Schrödinger (3NLSE) Equation. A positive value for “Dispersion” parameter describes the formation of bright optical solitons whilst a negative value leads to the formation of dark solitons. The 3NLS Equation in general, describes a modulated wave packet propagating through a nonlinear dispersive medium with a constant velocity. For certain initial pulse shapes (the “Reflectionless Potentials”), the 3NLSE is completely integrable and the evolution of the soliton can be found in closed form by means of the Inverse Scattering Transform (IST) [7]. Solitons arising out of a balance between dispersive and Kerr nonlinearity effects possess dominant characteristic features one of which is the elastic collisions between them. Solutions described by non-integrable nonlinear wave equations on the other hand are usually referred to as solitary waves and collisions between solitary waves are inelastic and more complex in character. A solution of the integrable 3NLSE applicable to pulse propagation in optical fibres is the hyperbolic secant where an arbitrary positive number representing the soliton order, the distance along the fibre, and time,

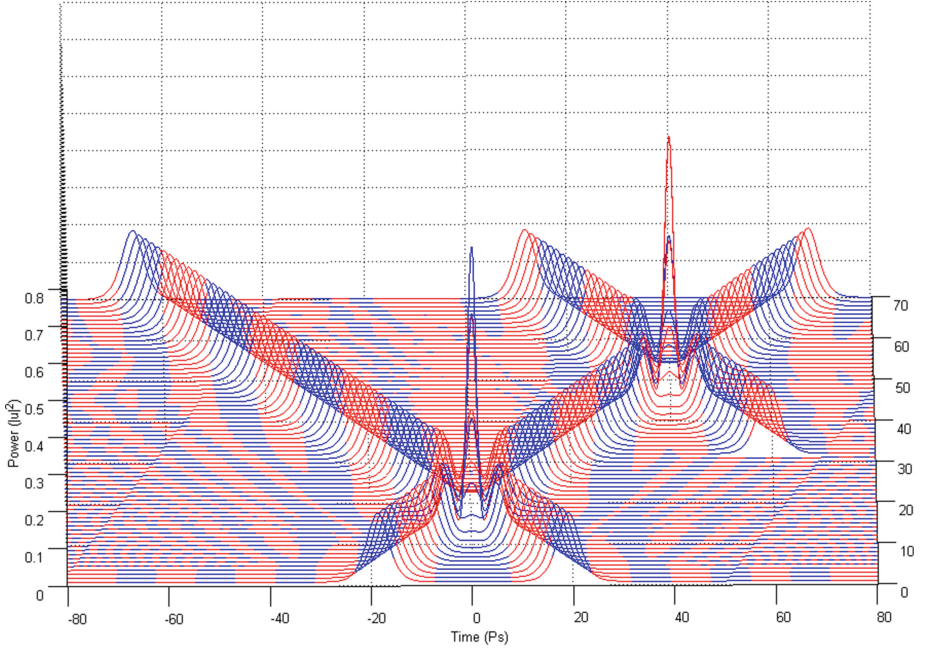


**Fig. 1.** A collision between two solitons. The second soliton is a “Time-Gated” input soliton.

all in normalised dimensionless units, are the main parameters forming the initial soliton propagation envelope. By coupling pulses in and out of a fibre at appropriate points (distance and time), useful computation could be possible based on collisions between solitons within the fibre.

The material presented in [3, 4] shows that in situations where optical solitons are formed within optical fibres (simulations have been carried out using the Split-Step Fourier Technique (SSFT)), with appropriate practical arrangements, computationally universal systems based on collisions between first order solitons are possible using logical gates based on the “Controlled” type of gates originally proposed by Toffoli and Fredkin [5, 6]. As an extension to what presented in the above mentioned papers, in this present paper, the numerical study of collisions between first order solitons is expanded leading towards an all-optical FFT (Fast Fourier Transform) calculation. The CN and CCN soliton gates continue to be the essential ingredient of the computational model.

In what follows in this paper, the encoding rules for the bit/s representation into our system (by admitting the existence of only two solitons, one with a phase value of  $\pi$  and one with a phase value of 0) follow exactly those outlined in [3, 4] where the reader is referred for more details. This way there can only two types of collisions exist between solitons in our system: (a) two solitons



**Fig. 2.** Collision between three solitons in the cubic 3NLSE domain. The third of the solitons taking part in the collision is a “Time-Gated” soliton in phase with the initial two.

collide and are in phase (Figs. 1 and 2) or, (b) two solitons collide and are out of phase (Fig. 4). This way we can directly use the solitons themselves as input values to a soliton logic gate. The most important fact of all is that these two types of collisions possess the property of sequencing, so they can be cascaded. Using this definition we can go a bit further and consider the collision between solitons, as the inner process of the soliton logic gate and the two recovered with their original state after the collision solitons, as the output values of the logic gate. So, basically, we split the whole process of a collision into two important parts. The first part consists of the logic gate length, bounded between initially the point at which solitons begin to propagate through the medium, and the point at which the two solitons collide, creating a characteristic for their phase values “Collision Envelope”. The second part starts from the point of collision, extending all the way up to the point where the two solitons recover their initial time positions in reverse order after the collision.

## 2 The Half-Adder Processor Scheme

The half-adder processor scheme, first introduced in [3], forms the essential central building block on which the overall FFT soliton computational scheme is wrapped around. The system reads the collision envelopes at distance and time specified points and uses this information to generate solitons with an appropriate phase value to represent the output of each “gate”. The phase values of two of the output solitons determine the “sum” and “carry” outputs at the end of the computation process whilst all other solitons are superfluous to this calculation. By definition the half-adder (the sum implementation) is given by:

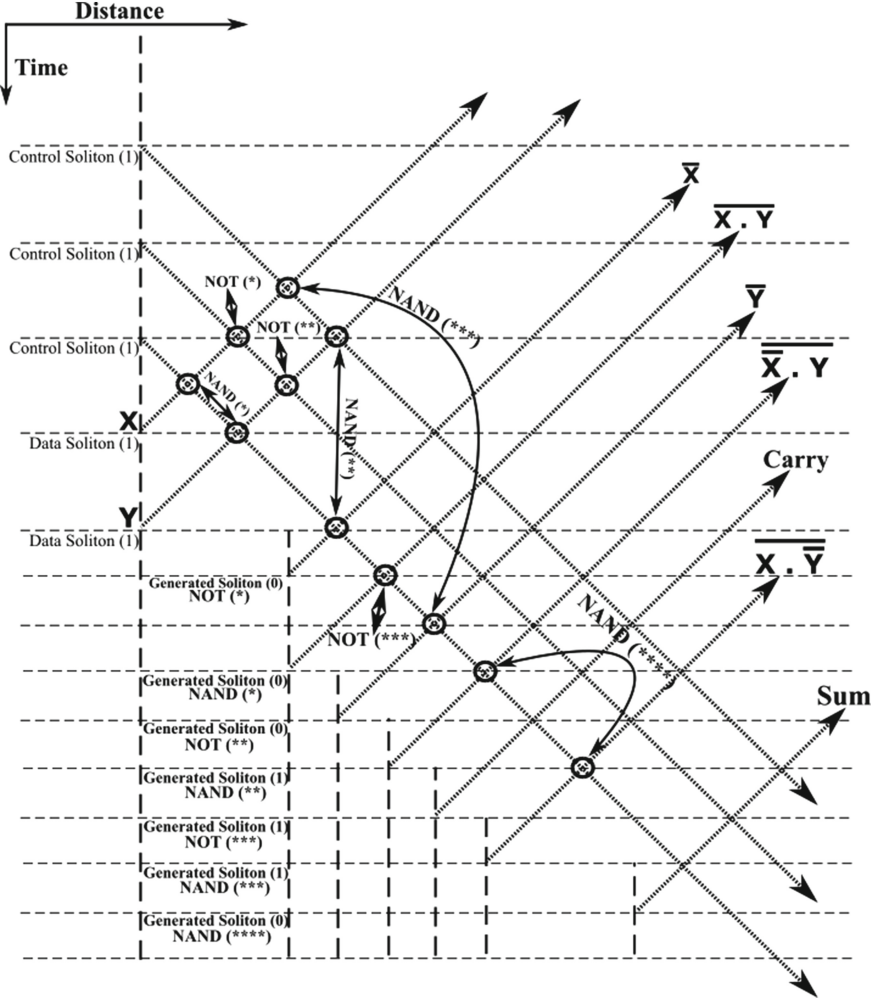
$$\overline{(X \cdot \bar{Y})} \cdot \overline{(\bar{X} \cdot Y)} \quad (1)$$

In Fig. 3 the equivalent soliton scheme, originally presented in [3], is reproduced for convenience. The points highlighted in this schematic representation by means of a bold circle indicate functional points at which a soliton collision, part of a gate, takes place; while, X and Y denote the initial input data. Full “gate” arrangements have been named and numbered (e.g. NAND (\*), indicates the first NAND in the computational arrangement, NAND (\*\*) the second, etc.).

In Figs. 4 and 5, the “Input” and the “Output” of the schematic representation of Fig. 3 is reflected on actual soliton collision simulations. Each individual gate-soliton collision is presented in a separate figure for clarity and comparison purposes. The simulation figures are to be followed in a top-to-bottom approach in the schematic representation of Fig. 3.

In all the figures the input-output “gate” sequence follows the soliton propagation direction. The point at which the soliton propagation begins (point 0 in the propagation scale across the depth of the figure) also reflects the input side of the “gate” and respectively, the point at which the soliton propagation ends (point 100 in the propagation scale across the depth of the figure) reflects the output side of the “gate”.

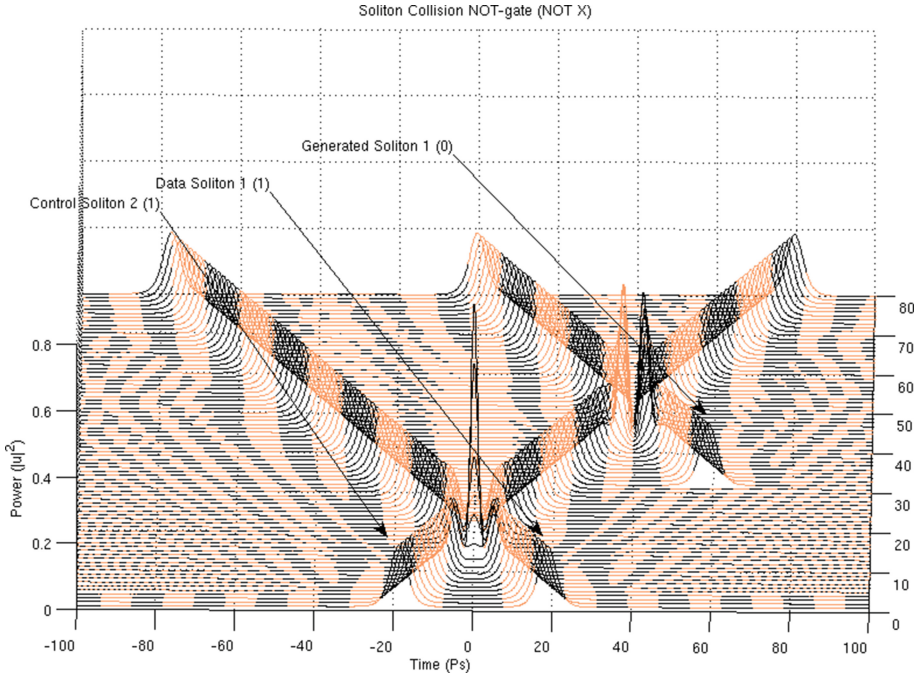




**Fig. 3.** The half-adder processor.

The half-adder computational arrangement plays a vital role in what is to follow as is this particular arrangement the one that is lying at the heart of the more general “Multiplier” arrangement, about to be presented later on, and required for the realisation in the end of the complete “Butterfly” calculation process which directly leads to the all-optical soliton FFT computational arrangement.

At this point and for the approach used for the presentation of the material to follow in this paper to become clear, we need to stretch-out the fact that the computational complexities involved are extensively simplified if can become apparent that the scheme is flexible enough to be gradually get “packed” in fixed-purpose calculation lengths. This approach doesn’t suppress the system



**Fig. 4.** The soliton “gate” NOT(\*). The number in the brackets next to each soliton description is the bit value carried by the soliton.

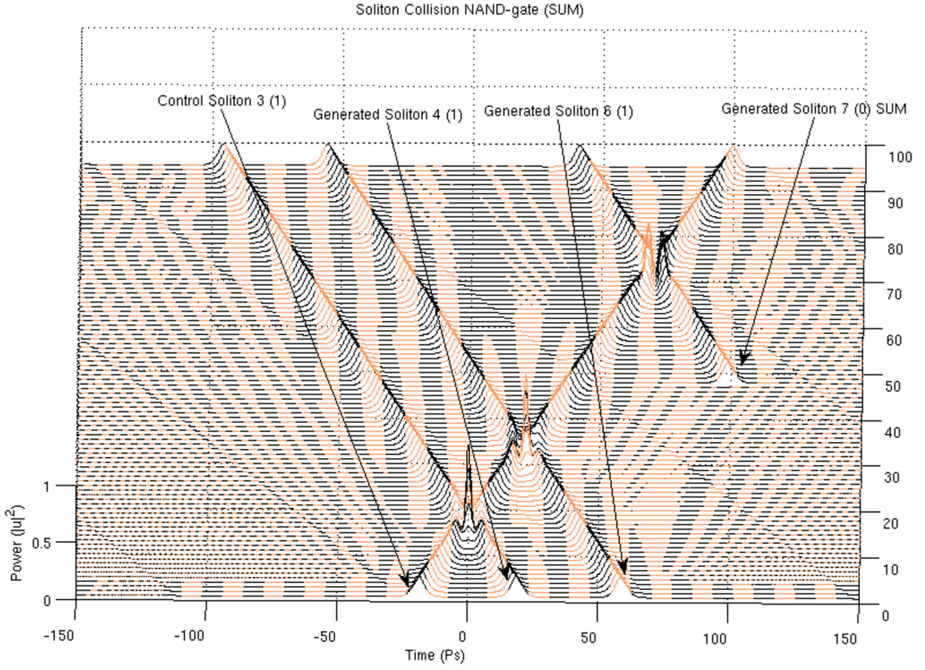
from its generalisation properties, as the fixed reading points (as these have been identified and introduced in [3,4]) still hold their properties and continue to provide the system with all the capabilities initially identified as inherently characteristic of the computational system at hand.

This systematic type of approach, will give us the ability to investigate the properties (as well as the validity) of each individual computational block in turn and, when the individual parts are finally interconnected to form one “Butterfly” arrangement, to do the same regarding the properties and validity of the overall computational scheme.

### 3 The Two 2-Bit Numbers Multiplier

In this section we present the “Two 2-bit Numbers Multiplier”, which involves a half-adder as its lying-in-its-heart functional unit (“Three-bit Adder Arrangement”). The particular arrangement forms the compact small-scale equivalent of the “Two maximum-number-of-bits Numbers Multiplier”, which for general purpose calculations must involve full-adders as well as half-adders in its arrangement.

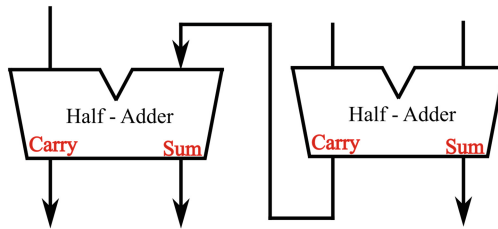
The reason behind choosing the Two 2-bit Numbers Multiplier is only the fact that the particular arrangement possesses all the functionalities and properties



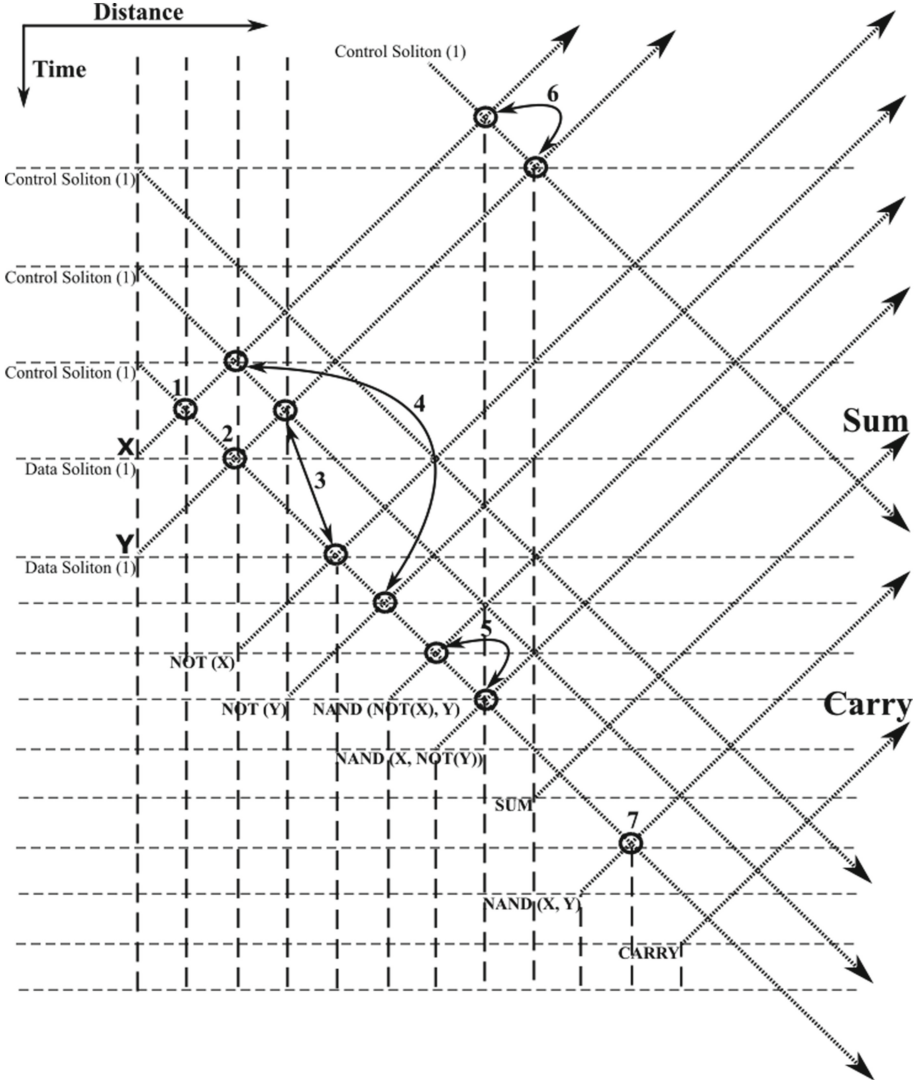
**Fig. 5.** The soliton “gate” NAND(\*\*\*)). The number in the brackets next to each soliton description is the bit value carried by the soliton.

need to be demonstrated, while at the same time gives us the ability to keep the material presented at a minimum of extension and complexity in this paper.

Starting from the half-adder arrangement, if we now take a closer look in Fig. 3 we will notice that all the output solitons need to be ignored after reading and only the output soliton representing the “carry” value is to be allowed to propagate further on and enter the cascading second half-adder arrangement. Is exactly this soliton-bit that is required for the arrangement to complete the Three-bit Adder Arrangement output calculation as presented in a conventional block diagram in Fig. 6. This “Soliton Suppression” requirement at the very end

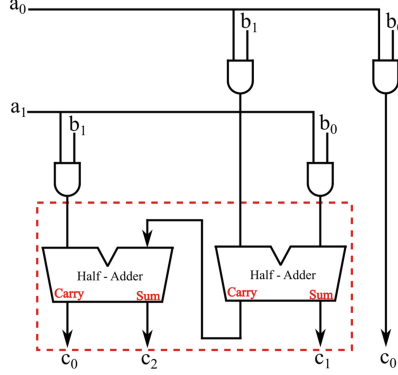


**Fig. 6.** The three-bit adder.



**Fig. 7.** The alternative half-adder arrangement.

of a computational arrangement is not characteristic only of the computational scheme here presented but rather a common characteristic requirement in soliton computational arrangements as, for example, of the one introduced in [8], where the additional property of not intersecting (solitons crossing paths but not colliding) is also a vital system characteristic requirement. The usual formal term coined for such kind of solitons is “Garbage Solitons” and is chosen to emphasise the fact that these solitons are to play no active role in the cascading calculations following the output of an arrangement. The way this “Soliton Suppression” can



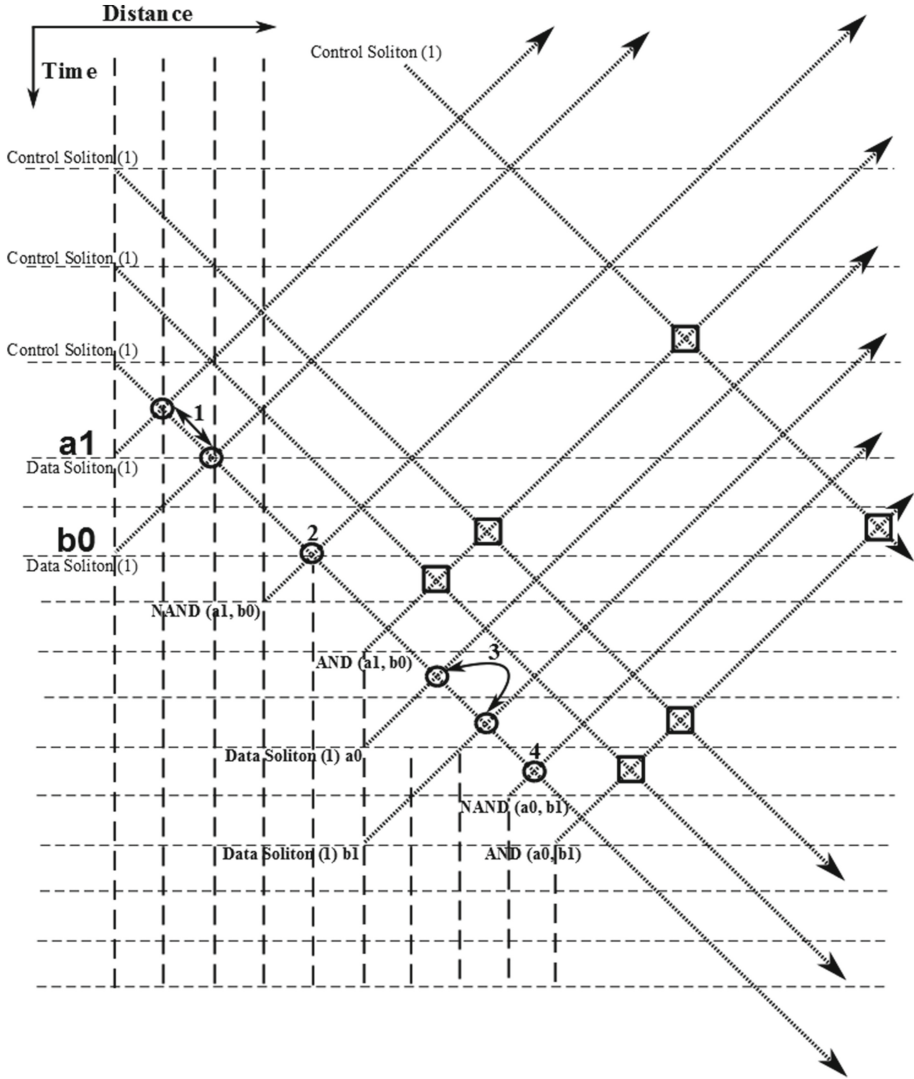
**Fig. 8.** The “Two 2-bit Numbers Multiplier”.

be physically achieved is, in general terms, a technicality, requiring some hands-on experimental work, in order for different methods and their corresponding effects on the overall computational arrangement to be properly studied. For these reasons we postpone, at this point, the explanation of how this “Soliton Suppression” can be accomplished.

In order to present a complete picture of the soliton arrangements as well as the almost unlimited flexibility possessed by the computational system (another reason is that in the view of the author the concept of “Garbage Solitons” is neither entirely satisfactory nor properly defined in its physical terms), in Fig. 7 an alternative soliton arrangement is presented which doesn’t need “Soliton Suppression” any more in order for the cascading half-adder arrangement to commence calculation.

In this new arrangement the general soliton pattern remains the same as in the original version, with the only difference that now the third control soliton is starting propagation at a time position shifted to the left (top) by four time slots (in Fig. 7 the original third control soliton propagation route has been maintained as well for comparison purposes). The order in which the individual gates are presenting their results is slightly changed as well. Shifting the third control soliton by four time slots to the left (top) of the arrangement has as a result for the soliton carrying the “carry” value to appear at the end (bottom) of the output soliton order. So, this soliton can now be taken as the first input soliton of the new half-adder arrangement (literally, as it possesses the same propagation angle as the original input solitons to the half-adder arrangement) which, by use of a second appropriate input soliton and three control solitons, as required by the scheme, can provide us with the final computational result, without the need to include any kind of “Soliton Suppression” procedure.

Having established and demonstrated the Three-bit Adder Arrangement, we can now build around it the full Two 2-bit Numbers Multiplier. The overall arrangement requires the addition of another four AND gates, to accommodate



**Fig. 9.** Part of the “Two 2-bit Numbers Multiplier” (including two of the initial AND gates and the half-adder arrangement without the corresponding generated solitons).

initial bit multiplications. The conventional diagram arrangement for the multiplier is as presented in Fig. 8.

In Fig. 9 part of the Two 2-bit Numbers Multiplier is presented. For illustration purposes Generated Solitons in Fig. 9 are shown to be closer together than they should be in an actual computational arrangement without losing in computational properties or upsetting the result. Circular soliton collision points indicate collisions taking place during the initial AND gates calculations,



while square soliton collision points indicate collisions taking place as part of the half-adder calculation process. The arrangement in Fig. 9 illustrates a certain degree of parallelism in the calculation process, which contributes significantly in increasing the overall computational speed of the arrangement. It comes without saying that the Two 2-bit Numbers Multiplier arrangement illustrated can be extended to cover any bit length required for the multiplication between two individual numbers. Again, the purpose here was to keep the length of the illustration to a minimum.

#### 4 The “Butterfly” Soliton Arrangement

For the remaining part of the “Butterfly” calculation process, we need a soliton arrangement to convert a positive bit-number to a negative one. In order to achieve this we adopt the method of complementing each digit in a bit-number in turn (change 1 for 0 and 0 for 1) and then add 1 to the result. That way, the bit-number taken out of the procedure corresponds to a bit-number representing the negative equivalent of the initial bit-number.

A series of collisions between the solitons carrying the bit-number values and a single control soliton with a phase value opposite to the one possessed by the control soliton that generated the initial bit-number, is enough to produce the bit-number complement. Since all the control solitons used so far in the

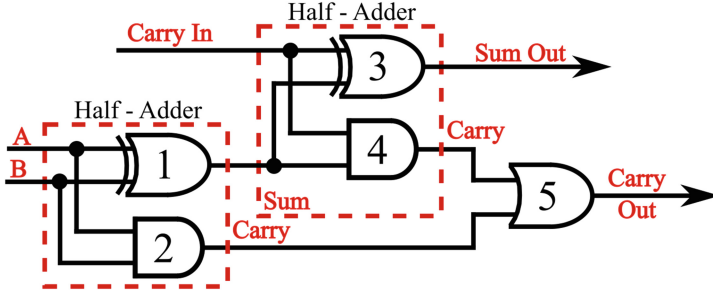


Fig. 10. The full-adder (conventional logic arrangement).

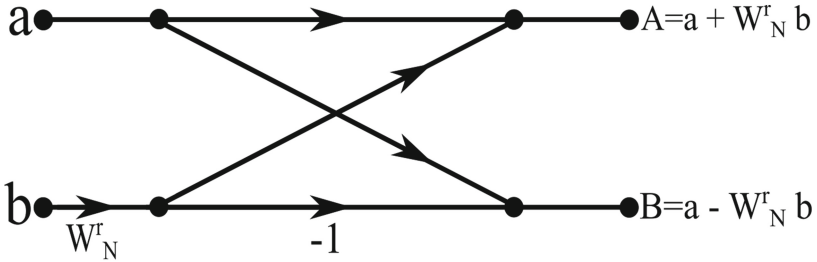
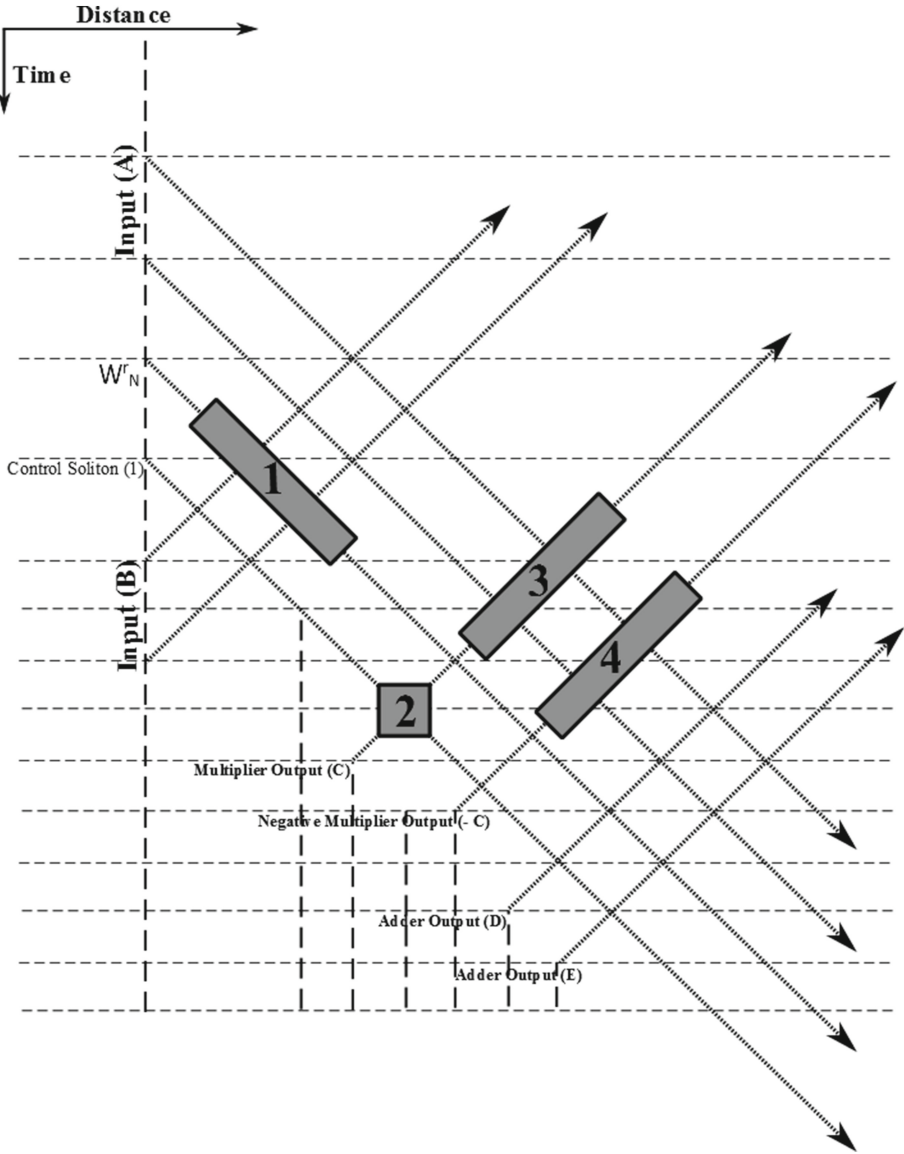


Fig. 11. Basic “Butterfly” computation in the decimation-in-time FFT algorithm.



**Fig. 12.** The “Butterfly” soliton arrangement. [(1) Multiplier arrangement, (2) Negation arrangement, (3) Addition arrangement, (4) Addition arrangement].

computational arrangements presented had a phase value of  $\pi$ , corresponding to a bit value of 1, the appropriate control soliton to achieve the complement calculation must possess a phase value of 0, in turn corresponding to a bit value of 0. The addition of 1 to the complement can be easily achieved by means of full-adder arrangements internally consisting of two interconnecting half-adder

arrangements and an OR gate, according to the conventional logic scheme presented in Fig. 10.

After the complement of a bit-number has been calculated, subtracting it from another bit-number requires the addition between the complement calculated and the second bit-number. That way only half-adder and full-adder arrangements are required for the realisation of all the calculations involved in the “Butterfly” arrangement. Addition and subtraction calculations appear at the final stages of the “Butterfly” (Fig. 11), those that actually are giving the result and passing the values calculated to the next processing stage of the overall FFT calculation arrangement.

Having completed the presentation of the individual parts out of which the soliton “Butterfly” arrangement consists of, we can now present the schematic of the overall arrangement required. Figure 12 presents the soliton “Butterfly” arrangement to full extend omitting, by means of a “black box” representation, those parts of the arrangement which have been previously analysed and illustrated. “Adder Output” (D) and “Adder Output” (E) appear at the end of the arrangement as required for the cascading “Butterfly” arrangements to continue further processing the data. All the output soliton propagation routes shown are indicative, since in an actual calculation of bit-numbers more than one solitons will represent the output bit-number of each block of calculation. As it is the case with the conventional Radix-2 FFT algorithm the first and the second decimation process results in a “shuffling” of the input data sequence, which has a well-defined order.

## 5 Conclusions

In this paper we surveyed the possibilities of an all-optical soliton FFT calculation and shown how this can become possible within the boundaries of the optical soliton 3NLSE domain. The outcome of this investigation is leading the way towards a fast all-optical soliton FFT calculation with the FFT phasors (roots of unity) to be represented directly by solitons of corresponding phase values, currently under extensive research by the author. In such a scheme the 8-point FFT phasors, for example, can be directly represented as:  $W_8^0 \rightarrow \text{Soliton phase value} = 2\pi$ ,  $W_8^1 \rightarrow \text{Soliton phase value} = \frac{\pi}{4}$ ,  $W_8^2 \rightarrow \text{Soliton phase value} = \frac{\pi}{2}$ ,  $W_8^3 \rightarrow \text{Soliton phase value} = \frac{3\pi}{4}$ ,  $W_8^4 \rightarrow \text{Soliton phase value} = -2\pi$ ,  $W_8^5 \rightarrow \text{Soliton phase value} = \frac{5\pi}{4}$ ,  $W_8^6 \rightarrow \text{Soliton phase value} = \frac{6\pi}{4}$ ,  $W_8^7 \rightarrow \text{Soliton phase value} = \frac{7\pi}{4}$ ,  $W_8^8 \rightarrow \text{Soliton phase value} = 2\pi$ , while the soliton phase values of  $\pi$  and 0 remain reserved to represent digit 1 and digit 0 respectively for the control and data solitons involved. This additional ability, when properly specified, will provide the overall computational scheme with a separate, well defined, and of a smaller fixed length FFT calculation arrangement without the need for it to consist of individual calculation arrangements based on the scheme’s “gates”.

## References

1. Jakubowski, M.H., Steiglitz, K., Squier, R.K.: Computing with solitons multi-valued logic. Special Issue on Collision Based Computing **6**, 5–6 (2001)
2. Jakubowski, M.H., Steiglitz, K., Squier, R.K.: When can solitons compute? Complex Syst. **10**(1), 1–21 (1996)
3. Bakaoukas, A.G., Edwards, J.: Computing in the 3NLS domain using first order solitons. Int. J. Unconventional Comput. (IJUC) **5**(6), 489–522 (2009). ISSN: 1548–7199
4. Bakaoukas, A.G., Edwards, J.: Computing in the 3NLS domain using first and second order solitons. Int. J. Unconventional Comput. **5**(6), 523–545 (2009). ISSN: 1548–7199
5. Tooli, T.: Reversible computing. In: de Bakker, J., van Leeuwen, J. (eds.) Automata, Languages and Programming. LNCS, vol. 85, pp. 632–644. Springer, Heidelberg (1980)
6. Fredkin, E., Toffoli, T.: Conservative logic. Int. J. Theor. Phys. **21**, 219–253 (1981)
7. Ablowitz, M.J., Segur, H.: Solitons and the Inverse Scattering Transform. SIAM, Philadelphia (1981)
8. Rand, D., Steiglitz, K.: Computing with Solitons, July 1 2007
9. Pelinovsky, D.E., Afanasjev, V.V., Kivshar, Y.S.: Nonlinear theory of oscillating, decaying, and collapsing solitons in the generalised nonlinear schrödinger equation. Phys. Rev. E **53**(2), 1940–1953 (1996)
10. Hasegawa, Akira: Optical Solitons in Fibers. Springer, New York (1990)
11. Fibich, G., Wang, X.-P.: Stability of solitary waves for nonlinear schrödinger equations with inhomogeneous nonlinearitie. Physica D **96**(108), 96–108 (2003). Elsevier
12. Miller, P.D., Akhmediev, N.N.: Do solitons exchange conserved quantities during collisions? Phys. Rev. Lett. **76**(1), 38–41 (1996)
13. Micallef, R.W., Kivshar, Y., Love, J.D., Burak, D., Binder, R.: Generation of spatial solitons using non-linear guided modes. Opt. Quantum Electron. **30**, 751–770 (1998). Chapman & Hall
14. Ostrowsky, D.B., Reinisch, R. (eds.): Guided Wave Nonlinear Optics. Kluwer, Dordrecht (1992)
15. Akhmediev, N.N., Ankiewicz, A.: Solitons: Nonlinear Pulses and Beams, Chap. 12. Chapman & Hall, London (1997)

Unconventional Computation and Natural Computation  
15th International Conference, UCNC 2016,  
Manchester, UK, July 11-15, 2016, Proceedings  
Amos, M.; CONDON, A. (Eds.)  
2016, XVIII, 197 p. 62 illus., Softcover  
ISBN: 978-3-319-41311-2