$$n_w = n_w^+ + n_w^- = a_{n-1}^+ a_{n-2}^+ a_{n-3}^+ \ldots a_2^+ a_1^+ a_0^+ - a_{n-1}^- a_{n-2}^- a_{n-3}^- \ldots a_2^- a_1^- a_0^-$$
$$= \begin{pmatrix} a_{n-1}^+ a_{n-2}^+ a_{n-3}^+ \ldots a_2^+ a_1^+ a_0^+ \\ a_{n-1}^- a_{n-2}^- a_{n-3}^- \ldots a_2^- a_1^- a_0^- \end{pmatrix} \quad (1)$$

where

$a_i^+ = \langle 0,1 \rangle$ and $a_i^- = \langle 0, \bar{1} \rangle$, in which $i = 0,1,...,n-1$; and

<, > indicates a set of possible values for the variable.

$\bar{1}$ is a shorthand for a nonzero bit in the y substring, consisting of a bar over a digit 1.

The weight of each bit in the binary strings is unity or a positive-integer power of 2. Hence, n_w can be expressed as:

$$n_w = \sum_{i=0}^{n-1} 2^i a_i^+ - \sum_{i=0}^{n-1} 2^i a_i^- = \sum_{i=0}^{n-1} 2^i \left(a_i^+ - a_i^- \right) \quad (2)$$

The domain of n_w covers a range from $-(2^{\mathrm{n}} - 1)$ to $2^{\mathrm{n}} - 1$.

An algorithm based on the above may be implemented by a circuit block (not shown), which takes in an 8-bit binary string and outputs a pair of x substring and y substring, each in 8-bit. When a number in the form of a binary string is fed into the logic circuit, a replica of the binary string is made and provided as the x substring, to which is assigned an all-zero binary string of the same length for the y substring. The pairing completes the process of generalization. However, if the sign bit of the binary string is nonzero, the replica of the binary string for the x substring stops short of this sign bit. The nonzero sign bit is placed in the same bit-position (the most significant) in the otherwise all-zero y substring.

Therefore, a binary string 1000 0011 (-125 in decimal) is generalised into an x substring of 0000 0011 (3 in decimal) and a y substring of 1000 0000 (an absolute number 128 in decimal). As mentioned, the y substring is subtrahend to the x substring as minuend, i.e. x - y = $3 - 128 = -125$.

Upon the pairing, the original binary string has become “generalised” because the string’s value is no longer represented by a binary string but by one of many possible pairs of x and y substring as subtrahend that give and identical sum, in accordance to the General Equivalence Principle.

Preferably, the x substring and y substring are each allocated a string length longer than that of the original binary string by one bit. This extra bit allows the maximum value of the x substring to bigger by one binary order, such that the value of the original binary string falls easily within a large minuend value from which the value of y substring may be subtracting to give the value of the original binary string.

A “Specific” Equivalent Principle Conversion

A particularly useful conversion based on the Equivalence Principle is described in some detail in Box 1. It is easier to appreciate the use of the Generalised Binary Number Format by considering what is called in this description the Specific Equivalence Principle (SEP) conversion.

Box 1

The *Specific Equivalence Principle* in a generalized binary number system states that:

- *Any substring of* $m\ (\geq 1)$ *1’s can be replaced by an 1 (more significant) and an* $\bar{1}$ *(less significant) separated by m-1 zeroes.*
- *Similarly, any substring of* $m(\geq 1)\ \bar{1}$*’s can be replaced by an* $\bar{1}$ *(more significant) and an 1 (less significant) separated by m-1 zeroes.*

The effect of the SEP conversion removes all adjacent nonzeros in a substring and can be applied separately to each bit-position in parallel processes.

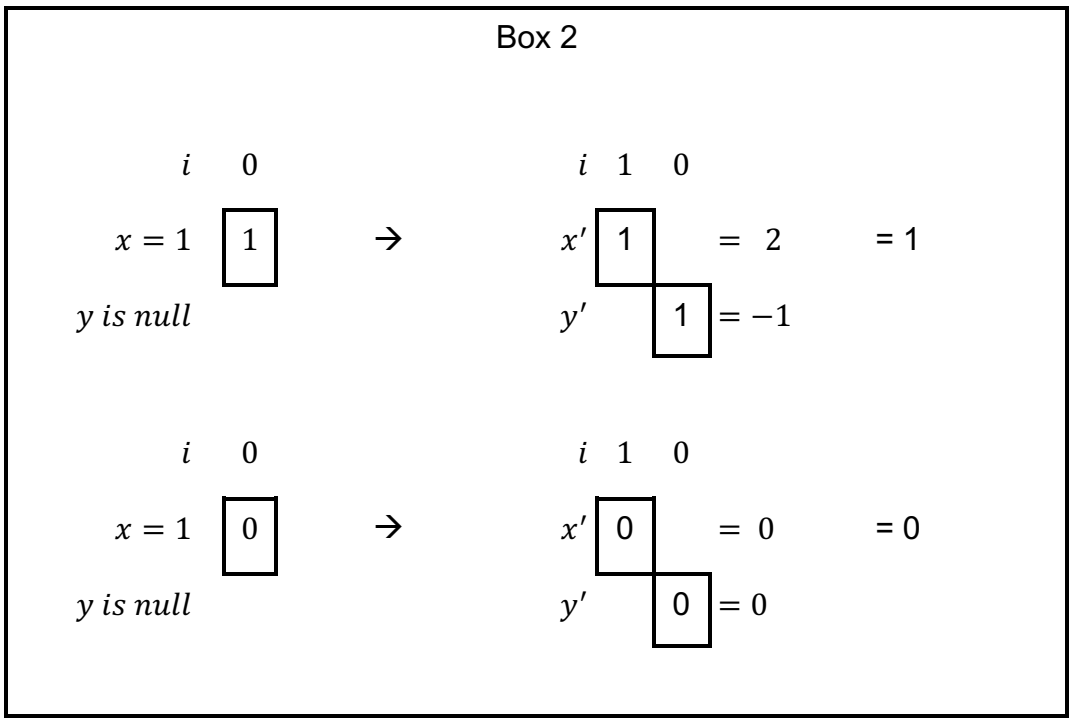
Taking the positive string as $x_i = \langle 0,1 \rangle$ and the negative string as $y_i = \langle 0, \bar{1} \rangle$, the *Specific Equivalence Principle* for a substring of *m* (≥ 1) 1's can be expressed by Equations (3a) and (3b), while that for a substring of *m* (≥ 1) $\bar{1}$'s can be expressed by Equations (4a) and (4b).

$$X_i = \bar{x}_i x_{i-1} \quad ; \quad Y_i = x_i \bar{x}_{i-1} \qquad \text{(3a,3b)}$$

$$X_i = y_i \bar{y}_{i-1} \quad ; \quad Y_i = \bar{y}_i y_{i-1} \qquad \text{(4a,4b)}$$

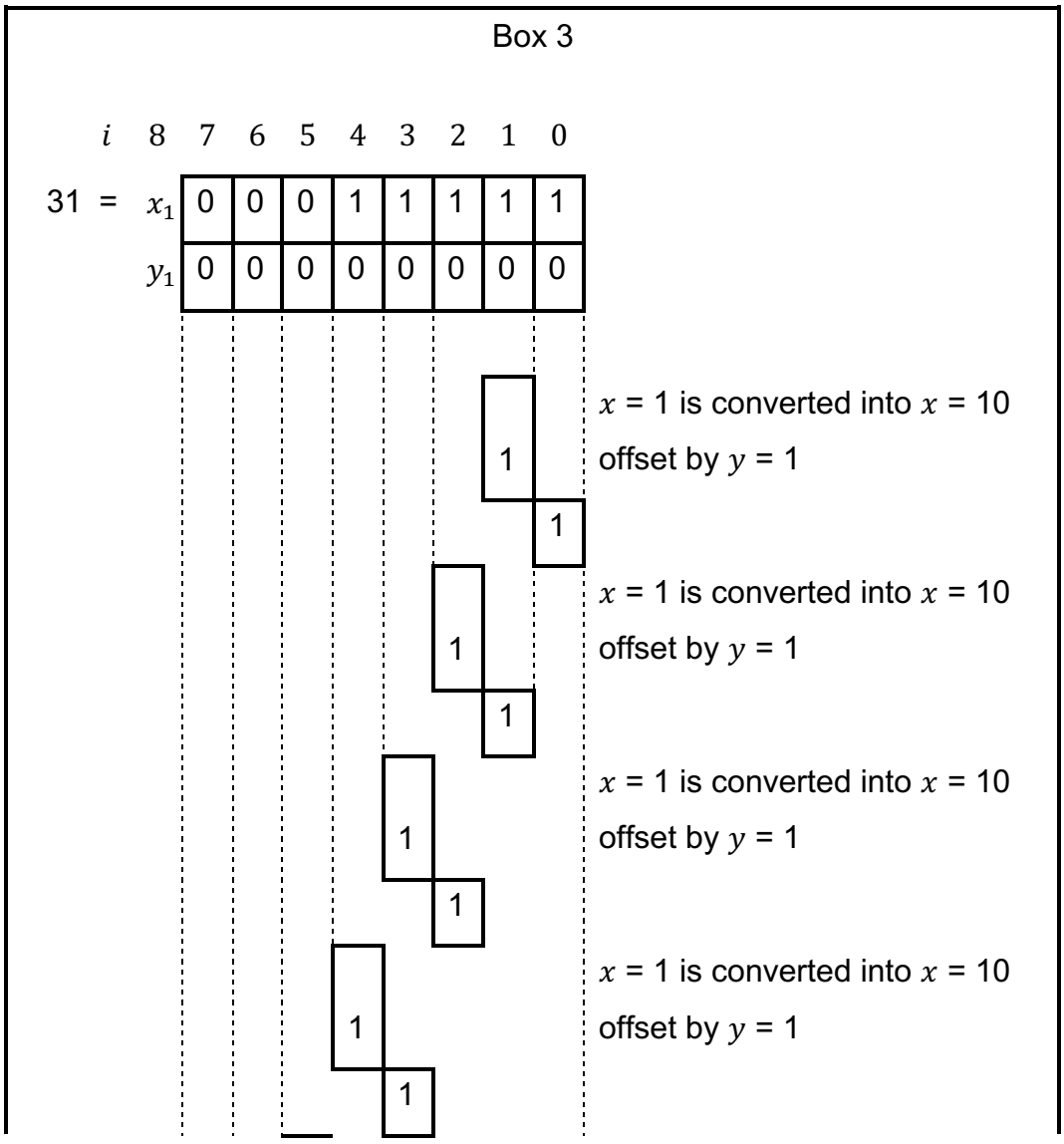
Box 2 below shows the SEP conversion, which is based on the possibility of re-expressing a positive bit having a nonzero value of 1 as a greater number offset with a smaller and negative number, such as $x - y = 2 - 1 = 1$, while observing the Equivalence Principle. For completeness, Box 2 shows that if the bit has zero value the output is zero at a higher order. Actually, this is determined passively from the active re-positioning of nonzero bits in the output.

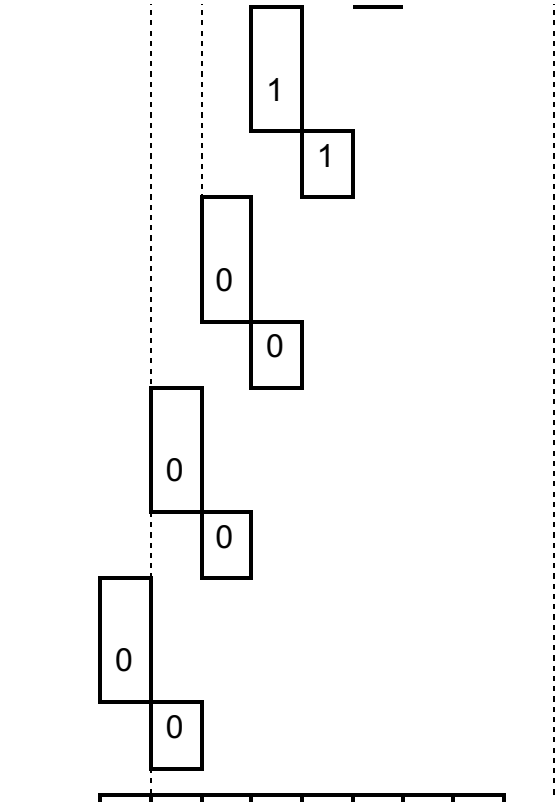
Box 2



An example of applying the SEP conversion in Box 2 to a binary string is given in the following paragraphs. Taking as example the number 31, which in 8-bit binary format is 0001 1111, generalization re-expresses 0001 1111 as an x substring = 0001 1111, paired with an all-zero y substring = 0000 0000 as there is no sign bit. Box 3 below shows the SEP conversion shown in Box 2 applied to the x substring. The x substring is converted into 0010 0000 (32 in decimal) and the y substring is converted into 0000 0001 (1 in decimal). Therefore, $x - y$ in the decimal system is $32 - 1 = 31$, giving the same sum and observing the General Equivalence Principle. For convenience, these substrings may be written as 0010 000$\bar{1}$ in the recommended shorthand described in Equation (1).

Box 3





x = 1 is converted into x = 10 offset by y = 1

x = 0 is converted into x = 00 and y = 0

x = 0 is converted into x = 00 and y = 0

x = 0 is converted into x = 00 and y = 0

x_1' =	0	0	0	1	1	1	1	1	
y_1' =		0	0	0	1	1	1	1	1

These results are obtained by compiling the output of the SEP conversion of each bit-position.

32 =	0	0	0	1	0	0	0	0	**0**
-1 =	**0**	0	0	0	0	0	0	0	1

Where x = 1 an y = 1 in the same bit-position, the two nonzero bits cancel out into zero, i.e., simplification.

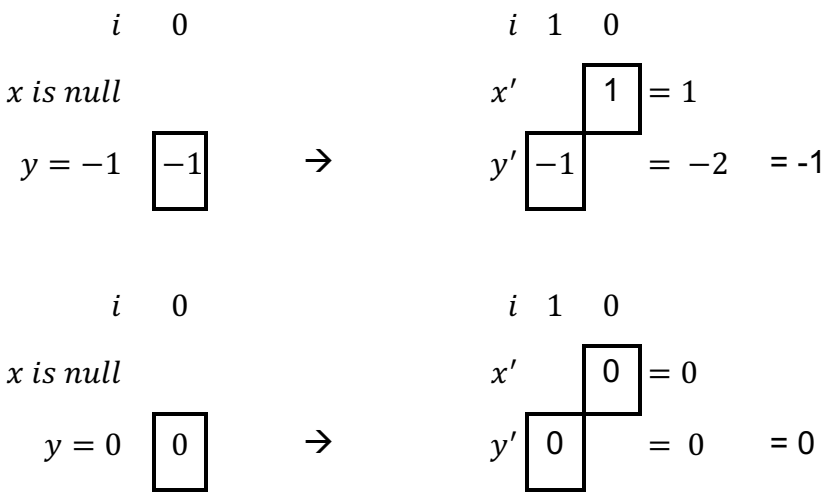
As described already in a preceding paragraph, it can be seen that the overall effect of the SEP conversion lies in the removal of all immediately adjacent nonzero bits in the x substring, but without changing the sum 31, i.e. 32-1. The SEP conversion must be applied consistently to every bit-position in the x substring.

Using the shorthand $\bar{1}$ to denote a nonzero bit in the y substring, a pair of substrings x =0001 0000 and y = 0000 0001 may be written in a single line as

0001 000$\bar{1}$. There cannot be a 1 and a $\bar{1}$ in the same bit-position as they cancel out and becomes zeros.

The SEP conversion may have to be applied also to every bit-position the y substring too, although there is nothing to demonstrate in the example of an all-zero y substring, Box 4 below shows the SEP conversion in Box 2 expressed for the y substring.

Box 4



The example in Box 4 has a final step of simplifying the SEP conversion output, which is not desirable for it adds an additional step to the overall embodiment.

Box 5 below relates to a variation of the SEP conversion, which produces outputs from inputs without the need of a simplification step after SEP conversion.

CLAIMS

1. A latch circuit for receiving a pair of bits, comprising:

a first latch input for one of the bit in the pair and a first latch output, a second latch input for the other one of the bit in the pair and a second latch output

(1) the first latch input being one of two pins of a first upstream logic gate, the first latch output being the output pin of a first downstream logic gate;

the output of the first upstream logic gate connected to one of two input pins of the first downstream logic gate;

the first latch output connected to the other one of two pins of the first upstream logic gate as a feedback;

(2) the second latch input being one of two pins of a second upstream logic gate, the second latch output being the output pin of a second downstream logic gate;

the output of the second upstream logic gate connected to one of two input pins of the second downstream logic gate;

the second latch output connected to the other one of two pins of the second upstream logic gate as a feedback;

(3) a midstream logic gate having two input pins each connected to one of the output of the first upstream logic gate and the output of the second upstream logic gate;

the output of the midstream logic gate connected the other of the two input pins of the second downstream logic gate and the other of the two input pins of the first downstream logic gate.

2. A latch circuit for receiving a pair of bits as claimed in claim 1, wherein each of the first and second upstream logic gates, the midstream logic gate and the first and second downstream logic gates being NOR gate.

3. A latch circuit for receiving a pair of bits as claimed in claim 1, wherein each of the first and second upstream logic gates, the midstream logic gate and the first and second downstream logic gates being NAND gate.

4. A latch circuit for receiving a pair of bits as claimed in claim 2, comprising the logic layout of**:**

inputs x_i, and y_i;

(i) 2-input NOR Gate 1; (ii) 2-input NOR Gate 2; (iii) 2-input NOR Gate 3; (iv) 2-input NOR Gate 4; and (v) 2-input NOR Gate 5;

wherein:

x_i is an input of NOR Gate 1;

y_i is an input of NOR Gate 2;

the output of NOR Gate 1 and the output of NOR Gate 2 are each one of the two inputs of NOR Gate 3;

the output of NOR Gate 1 and the output of NOR Gate 3 are each one of the two inputs of NOR Gate 4;

the output of NOR Gate 2 and the output of NOR Gate 3 are each one of the two inputs of NOR Gate 5;

the output of NOR Gate 4 is a feedback input of NOR Gate 1 and a first output X_i of the latch circuit; and

the output of NOR Gate 5 is a feedback input of NOR Gate 2 and as a second output Y_i of the latch circuit.

5. A latch circuit for receiving a pair of bits as claimed in claim 4, comprising the logic layout of:

inputs x_i, y_i and Clk;

(i) 2-input OR Gate 1; (ii) 2-input OR Gate 2; (iii) 2-input NOR Gate 3; (iv) 2-input NOR Gate 4; (v) 2-input NOR Gate 5; (vi) 2-input NOR Gate 6; and (v) 2-input NOR Gate 7;

wherein:

x_i is an input of OR Gate 1;

y_i is an input of OR Gate 2;

Clk is an input of OR Gate 1, and an input of OR Gate 2;

the output of OR Gate 1 is an input of NOR Gate 3; the output of OR Gate 2 is an input of NOR Gate 4;

the output of NOR Gate 3 and the output of NOR Gate 4 are each one of the two inputs of NOR Gate 5

the output of NOR Gate 3 and the output of NOR Gate 5 are each one of the two inputs of NOR Gate 6;

the output of NOR Gate 4 and the output of NOR Gate 5 are each one of the two inputs of NOR Gate 7;

the output of NOR Gate 6 is a feedback input of NOR Gate 3 and a first output X_i of the latch circuit; and

the output of NOR Gate 7 is a feedback input of NOR Gate 4 and a second output Y_i of the latch circuit.

6. A latch circuit for receiving a pair of bits as claimed in claim 3, comprising the logic layout of:

inputs x_i, and y_i;

(i) 2-input NAND Gate 1; (ii) 2-input NAND Gate 2; (iii) 2-input NAND Gate 3; (iv) 2-input NAND Gate 4; and (v) 2-input NAND Gate 5;

wherein:

x_i is an input of NAND Gate 1;

y_i is an input of NAND Gate 2;

the output of NAND Gate 1 and the output of NAND Gate 2 are each one of the two inputs of NAND Gate 3;

the output of NAND Gate 1 and the output of NAND Gate 3 are each one of the two inputs of NAND Gate 4;

the output of NAND Gate 2 and the output of NAND Gate 3 are each one of the two inputs of NAND Gate 5;

the output of NAND Gate 4 is a feedback input of NAND Gate 1 and a first output X_i of the latch circuit; and

the output of NAND Gate 5 is a feedback input of NAND Gate 2 and a second output Y_i of the latch circuit.

7. A latch circuit for receiving a pair of bits as claimed in claim 6, comprising the logic layout of:

inputs x_i, and y_i;

(i) 1-input NOT Gate 1; (ii) 1-input NOT Gate 2; (iii) 2-input NAND Gate 3; (iv) 2-input NAND Gate 4; (v) 2-input NAND Gate 5; (vi) 2-input NAND Gate 6; (vii) 2-input NAND Gate 7; (viii) 1-input NOT Gate 8; and (ii) 1-input NOT Gate 9;

wherein:

x_i is the input of NOT Gate 1;

y_i is the input of NOT Gate 2;

the output of NOT Gate 1 is an input of NAND Gate 3;

the output of NOT Gate 2 is an input of NAND Gate 4;

the output of NAND Gate 3 and the output of NAND Gate 4 are each one of the two inputs of NAND Gate 5

the output of NAND Gate 3 and the output of NAND Gate 5 are each one of the two inputs of NAND Gate 6;

the output of NAND Gate 4 and the output of NAND Gate 5 are each one of the two inputs of NAND Gate 7;

the output of NAND Gate 6 is a feedback input of NAND Gate 3, and the input of NOT Gate 8;

the output of NAND Gate 7 is a feedback input of NAND Gate 4, and the input of NOT Gate 9;

the output of NOT Gate 8 is a first output X_i of the latch circuit; and

the output of NOT Gate 9 is a second output Y_i of the latch circuit.

8. A latch circuit for receiving a pair of bits as claimed in claim 6, comprising the logic layout of:

inputs x_i, y_i and Clk;

(i) 2-input AND Gate 1; (ii) 2-input AND Gate 2; (iii) 2-input NAND Gate 3; (iv) 2-input NAND Gate 4; (v) 2-input NAND Gate 5; (vi) 2-input NAND Gate 6; and (v) 2-input NAND Gate 7;

wherein:

x_i is an input of AND Gate 1;

y_i is an input of AND Gate 2;

Clk is an input of AND Gate 1, and an input of AND Gate 2;

the output of AND Gate 1 is an input of NAND Gate 3;

the output of AND Gate 2 is an input of NAND Gate 4;

the output of NAND Gate 3 and the output of NAND Gate 4 are each one of the two input of NAND Gate 5

the output of NAND Gate 3 and the output of NAND Gate 5 are each one of the two input of NAND Gate 6;

the output of NAND Gate 4 and the output of NAND Gate 5 are each one of the two input of NAND Gate 7;

the output of NAND Gate 6 is a feedback input of NAND Gate 3 and a first output X_i of the latch circuit; and

the output of NAND Gate 7 is a feedback input of NAND Gate 4 and a second output Y_i of the latch circuit.