International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 9, No.6 (Nov-2020)
http://dx.doi.org/10.12785/ijcds/0906016

Approximate Population Count Using the Harley-Seal Method for Error-Tolerant Applications

Reem Jaffal[1], Sa'ed Abed[1], Imtiaz Ahmad[1] and Mahmoud Bennaser[1]

[1]*Department of Computer Engineering, Kuwait University, Safat 13060, Kuwait*

Received 26 Mar. 2020, Revised 11 May 2020, Accepted 31 Jul. 2020, Published 1 Nov. 2020

Abstract: Population count is a necessary process in several fields, such as cryptography, database search, data mining, and machine learning. Real-time problems have very large datasets, which require enhanced performance. Therefore, the objective of this paper was to propose a modified approach to the hamming weight algorithm to allow the implementation of approximate calculations on error-tolerant applications, such as machine learning and database searches, which do not require precise results. The proposed approach used approximate adders, rather than the exact carry save adder computations used in the Harley-Seal methodology of bit counting, along with modified imprecise error-tolerant adder type II (ETAII), named Approximate Harley-Seal with Modified ETAII (AHS-METAII). The precise versus imprecise designs of Harley-Seal approach were tested, evaluated, and compared to show that implementing partial logic functions instead of fully logic functions resulted in 27% power reductions with a slight decrease (9%) in the accuracy level over traditional adders on a 64-bit stream. The simulation results demonstrated that the proposed approximate approach model using Verilog was faster than the exact methods by 3% and consumed 16% less area.

Keywords: Population Count, Harley-Seal, Error-Tolerant Adder, Approximation, Delay, Power, Area

1. INTRODUCTION

The term hamming weight, also known as population count, popcount, or sideway sum, is the number of bits set to one in any binary stream given as an input of bits appears at the same time. Implementing this calculation can be done using both hardware and software [1] on strings that vary in length ranges. This paper tackles the never-ending issue of bit counting, a process that deals with the fundamentals of computer organization and has been debated ever since the infancy of launching the computer science field [2].

Hamming weight carries importance in many fields including -but not limited to- cryptography [3], data mining [4], machine learning [5], and database search [6]. Therefore, the need to enhance the speed of the hamming weight calculations has been increased to the point where some processors have a specialized population count method [7]. Suggested solutions expose either scalar parallelism or vector parallelism. However, certain applications are precision-loss tolerant by nature, and due to their existence, the need to implement inexact computing has gained popularity. Many evolving application classes like mining and synthesis that reveal intrinsic error resilience have redundancies in their input data sets, where some of them do not have a correct answer. Thus, the concept of error-tolerance resources was introduced, which delivers approximate results at an enhanced speed while showing improvement in power consumption, latency, and area at the expense of accuracy. Consequently, utilizing the mentioned characteristic to enhance the computation's speed performance through a simplified inaccurate circuit is required [8, 9].

Therefore, to keep up with the progressively high demands of applications requiring high-speed and power-efficiency, a new approach has been proposed to speed up the calculation of counting set bits in a stream by modifying the internal circuit blocks to boost the operation's performance. Since the discussed pop count method in this paper requires the use of traditional adders which play a vital role, the chosen approximate error-tolerance adders are not directly compatible for population count use. Hence, focusing our work-attention to this compatibility challenge allowed us to consider combining both (the exact and approximate adders) to compute the hamming weight at a narrow range of accuracy loss [10] which resulted in faster addition compared to previous results.

An abstract visualization of this approach is shown in Figure 1. The approach proposes the usage of the existing vectorized Harley-Seal algorithm [11] with approximate adders instead of exact adders to perform the addition process. Thus, decreasing dependencies between two related operations. The chosen approximate adder is Error-Tolerant Adder Type II (ETAII) [12], which is further modified to Modified Error-Tolerant Adder Type II (METAII) to be compatible with the mentioned implementation. The hybrid popcount approach consumes less power by 27% and utilized 16% less area as compared the use of exact adders only. In addition, it is faster than the

E-mail: reem.jaffal@ku.edu.kw, s.abed@ku.edu.kw, imtiaz.ahmad@ku.edu.kw and mahmoud.bennaser@ku.edu.kw

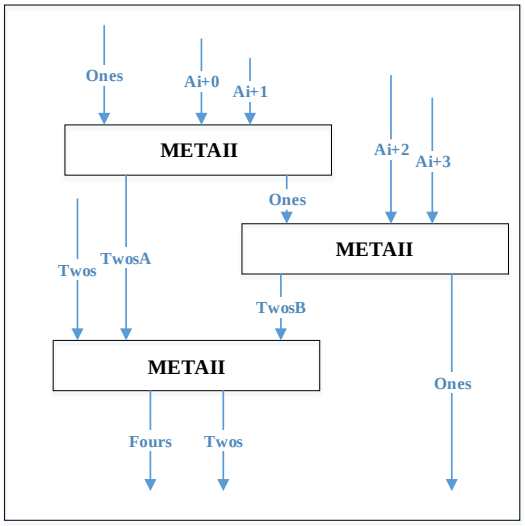

Figure 1. Modified Harley-Seal

exact methods by 3% with a slight loss in the accuracy level (i.e., 9%).

The main objective of this paper is to present an improved hybrid popcount approach based on the approximate adder principle, named as Approximate Harley-Seal with Modified ETAII (AHS-METAII) implemented using Verilog. Our contributions can be summarized as follows:

- Proposing an approximate method to count the number of ones inspired by the vectorized version of the Harley-Seal method.
- Proposing a modified version of the approximate error-tolerant adder type II (ETAII).
- Combining the approximate Harley-Seal with the modified ETAII to enhance speed and reduce power with a slight loss in accuracy.
- Comparing the exact Harley-Seal method results in terms of speed, accuracy, power and area with the modified approximate Harley-Seal method.
- Studying the effect of varying the number of words implemented in our Approximate Harley-Seal with Modified ETAII (AHS-METAII) approach using different popcount functions.
- Determining the best implementation in terms of performance metrics considered.

The organization of this paper is as follows: Section 2 provides an overview of the key existing algorithms related to counting-ones and then introduces current approximate adders. Section 3 gives a more detailed background of the Harley-Seal method and ETAII. Section 4 proposes the enhanced version of the approximate Harley-Seal methodology, which utilizes modified ETAII. Section 5 discusses the experimental results and compares the exact and approximate adders' performances. Finally, Section 6 concludes the paper and presents some future trends.

2. EXISTING ALGORITHMS AND RELATED WORK

This section discusses various existing non-complex methodologies to calculate the hamming weight with different effectiveness. Then, approximate adders are presented to accelerate the process of counting ones.

A. Bit-counting techniques

Hamming weight algorithms can be categorized into scalar algorithms and vectorized algorithms. Algorithms using the scalar approach have been discussed for the longest time [7, 11, and 13] and afterwards the concept of vectorization appeared [7]. Hence, algorithms shifted to the use of vectorization due to the enhancement it offers.

Naïve method is an obvious approach presented in [14], to count the number of ones in a string of bits. The method goes as follows: the least significant bit (LSB) is inspected; if it is set to one then the counter is incremented. As long as the remaining value is not equal to zero, the string of bits is repeatedly shifted. Although this is a simple method, it is the poorest in terms of efficiency. As it does not benefit from the distinct nature of this calculation, where it could be done in parallel. This method is further extended by using parallel bit reduction such that a naïve tree of adders (NTAD) which utilizes less operations per word as stated in [7]. As shown in Figure 2, a word of eight-bits (for simplicity) is grouped into levels, such that each two consecutive bits are summed in parallel. Next, the results of the previous step are added into a four-bit sub word. Finally, the four-bit sub words are summed to provide the final result.

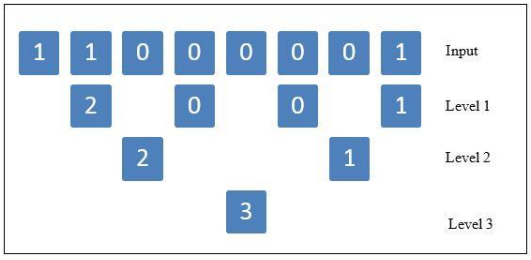

Figure 2. Trees of Adders

Wilkes-Wheeler-Gill (WWG) approach stated in [7, 15], optimizes the tree-of-adders count function such that it involves fewer instructions as shown in Figure 3. Next, keeping in mind that the MSB of each sub-word is zero, it adds consecutive sub-words into bytes. Efficiently, the multiplication and final shift sum all bytes such that the total population count is less than 64.

```
Inn [63:0] = Inn - ((Inn [63:0] >> 1) & 64'h5555555555555555);
Inn [63:0] = (Inn [63:0] & 64'h3333333333333333) + ((Inn [63:0]
>> 2) & 64'h3333333333333333);
Inn [63:0] = (Inn [63:0] & 64'h0f0f0f0f0f0f0f0f) + ((Inn [63:0]>>
4) & 64'h0f0f0f0f0f0f0f0f);
Inn [63:0] = (Inn [63:0] * 64'h0101010101010101) >> 56;
```

Figure 3. Wilkes-Wheeler-Gill method

Memoization (Look Up Table) algorithm is based on tabulations such that, tables are generated only once, where each row has the matching population count for each possible value. For example, in a 32-bit device, a single table size will be equal to 2^{32} and any given word is used as an index to the LUT to determine its hamming weight. Hence, a single read operation is enough. However, the table can be further reduced to be of size 2^{16}. Although the reduction reduces the table size to half, it also adds extra read operations and a splitting operation. This as a result, slows down the calculation process as presented in [16].

Arithmetic logic is another common method based on subtraction and masking bits, which could compete with previous methods when population count is expected to be low (e.g., less than 6.25% of a word) as stated in [12]. The procedure starts by decrementing the original stream by one then the decremented value is AND-ed with the original number. This process is repeated until the checked value equals zero.

In [11], vectorized technique (Harley-Seal) is illustrated, where the Carry Save Adder (CSA) is utilized for calculation. This methodology is chosen in our approach as it is faster than all of the non-vectorized (or scalar) functions [7]. It is favored in the case of having largely sized input bits, as it is quicker to reduce inputs and then bit count on two outputs; instead of directly bit counting on three inputs. Thus, reducing data amounts for which bit counting operate on directly [7]. Further details on Harley-Seal is presented in the next section.

B. Approximate adder designs

Adders are playing a vital role in Harley-Seal method [11]. Approximate addition also known as soft addition, relies on slackening exact and fully deterministic building blocks (like full adders). Since the approach of this work concentrate on redesigning a logic circuit into an approximate version, the result pertains to the functionalities of different logic circuits because of the approximate adder. However, there are many approximate adder approaches with different accuracy levels, as well as different implementations. Approximate adders are classified into four categories: Speculative Adders, Segmented adders, Carry Select Adders and Approximate Full Adders [10]. Speculative Adders include almost correct adder (ACA) [18], while segmented adders include ETAII [12] and Accuracy-Configurable Approximate Adder (ACAA) [9], carry Select Adders include Speculative Carry Select Adder (SCSA) [12] and Approximate Full Adders include Lower-Part-OR Adder (LOA) [19].

ETAI and ETAII approximate adders are discussed in [12]. ETAI deals with small input numbers where it segments the addition into two parts, an accurate part containing the most significant bits (MSBs) and an inaccurate one, which adds the least significant bits (LSBs) of the input. This adder type has a low accuracy result. Similarly, ETAII is a segmentation-based inaccurate adder that takes the value of the carry into consideration. The carry propagation trail is fragmented into several short routes. More explanation is shown in the following section. Thus, the carry propagation can be done in the short routes at the same time as the summation.

In [20], the Speculative Carry Select Adder (SCSA) is proposed. According to theoretical research, the lower bound on the critical path delay of an n-width adder has O(log n), complexity indicating that a sub-logarithmic delay cannot be attained by traditional adders. On the contrary, speculative adders revealed sub-logarithmic delay achievements through neglecting rare critical path input patterns. Based on the above observation, speculative adders were built. Differentiating between both adder types, each traditional adder output depends on all lower or equal significance preceding bits, while each speculative adder output only depends on preceding k bits rather than all preceding bits. Because SCSA is only employed on blocks instead of individual outputs, it is less vulnerable to errors.

In Accuracy-Configurable Approximate Adder (ACAA) introduced in [9], the circuit arrangement changes during accuracy configuration at runtime, thereby attaining a trade-off of accuracy against performance. In this n-bit adder, ⌈n/k - 1⌉ 2k-bit sub-adders are necessary. In order to correct the errors created by each sub-adder, an Error Detection and Correction (EDC) circuit is used in a pipelined architecture with the approximate adder to implement the accuracy configuration [10].

Other recent researches regarding approximate adders include [21, 22, and 23]. In [21], block-based carry speculative approximate adder (BCSA) is proposed. This adder is built on dividing the exact adder into blocks which work in parallel. Different type of adders can be implemented in these blocks as they are non-overlapped. The carry chain length is reduced and a select logic is suggested to speculate the carry input of each block based on some input operand bits of the current and next block. A new 32-bit Approximate Carry Look Ahead Adder (CLA) is introduced in [22] for Error Tolerant Applications. Approximate CLA is built on modified carry propagation equation of the exact CLA. The proposed design consumes less power and utilized less area than the exact one. Lower error obtained when compared to the other approximate CLA. In [23] four Approximate Full Adders (AFAs) are proposed for high performance approximate computing. The proposed adder is built on shortening the length of carry propagation along with achieving minimal error rate.

According to the comparison of approximate adders conducted in [10], ETAII, SCSA and ACAA have the similar accuracy levels when their parameters are equal and ETAII is considered as one of the most accurate and efficient adders among all the compared designs [17]. Moreover, it consumes less power and area and has shorter delay than ACAA and SCSA, as the logic block of ETAII is simpler than SCSA and ACAA. Regarding the power-

delay-product (PDP) value which is used to estimate the circuit characteristics of approximate adders since smaller delay not necessary means lower power consumption, ETAII has the lowest PDP compared with ACAA and SCSA [17]. It is found that ETAII is a good and compatible choice to be implemented with Harley-Seal bit-counting technique in order to improve its performance in terms of delay, power and area. To the best of our knowledge, this is the first work utilizing Harley-seal technique as one of the efficient and fastest popcount techniques using approximate adders and a great improvement in delay, power and area are achieved to be further implemented in error-tolerant applications.

3. BACKGROUND

This section presents more details on both the Harley-Seal technique and the ETAII approximate adder. Table I illustrates the notations used in the presented algorithms.

A. *Harley-Seal technique (HS-CSA)*

Harley-Seal is a vectorized technique where adders are used in a specific way to reduce the time it takes to calculate the hamming weight. The Carry Save Adder (CSA) is utilized for calculation. The CSA shown in Figure 4 is a sequence of independent n full adder which requires three inputs with two outputs, where each of those bits are binary values [11].

TABLE I. ALGORITHMS NOTATIONS.

Notation	Description
N	Input length in bits
c	Population counter
I	Iteration counter
$\mathbf{A_i}$	Word at i^{ich} iteration
ones	First least significant bits
twos	Second least significant bits
fours	Third least significant bits
eights	Fourth least significant bits
M	Number of input blocks to ETAII adder, where M ≥ 2
N/M	Block length in bits of ETAII adder
W	Number of segmented words
⊕	XOR operation
^	AND operation
∨	OR operation
$\mathbf{C_i}$	Carry of i^{is} full adder
$\mathbf{S_i}$	Sum of i^{th} full adder

In more details, the carry of the i^{th} full adder represents the overlapping bits such that it equals one when at least two of the inputs A_i, B_i, or C_i have their bits set. This is represented by (1):

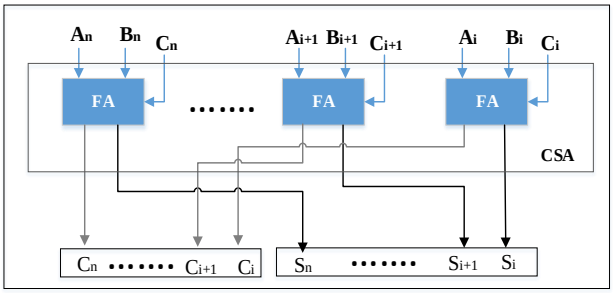

Figure 4. Carry Save adder

$$C_i = (A_i \wedge B_i) \vee ((A_i \oplus B_i) \wedge C_i) \quad (1)$$

On the other hand, sum accumulates bits such that the bit is set only when $A_i + B_i + C_i$ results in an odd number. The Sum result for each full adder is represented by (2).

$$S_i = A_i \oplus (B_i \oplus C_i) \quad (2)$$

The result is calculated using five logical operations, two of which are repeated (XOR, AND). While, OR logical operation is used once only, due to dependencies, the CSA operation takes at least three cycles.

For example, instead of iterating through the 16-bit of the bit stream 1100 1101 0011 0010, this method segments the stream into two parts where A=1100 1101, B=0011 0010, then adds them together. Therefore, the result of sum= 1111 1111 and carry=0000 0000, utilizing one of the mentioned scalar algorithms to count 8-bit instead of 16-bit to find the hamming weight.

Subsequently, translating the above explanation into an algorithm is presented below. As well as, an example of the algorithm is shown in Figure 5 with 16-bit input divided into eight words.

Assuming an input of N-bit stream is divided into a number of words (A_0, A_1, …) divisible by 8. Starting with three words (*Ones*, *Twos*, *Fours*) initialized to zero, represents the first, second and third least significant bits, respectively. Initializing a population counter *c* and an iteration counter *i* to zero (i.e., *c* = 0, *i* =0). The following steps take place:

1) Two new words (A_i, A_{i+1}) with *Ones* word are loaded into CSA function to write their output sum to the *Ones* register, while their resulting carry is loaded to a temporary register called *TwosA*.
2) Perform addition to the next inputs (*Ones*, A_{i+2}, A_{i+3}), where sum is stored to the *ones* and carry is loaded to a temporary register called *TwosB*.
3) Current three words with second LSBs (carry) are created (*Twos*, *TwosA* and *TwosB*). These three values are summed, and the result of the sum is stored in the *Twos* register, while carry is stored in a temporary register called *FoursA*.
4) Steps (1-3) are repeated with (A_{i+4}, A_{i+5}) and (A_{i+6}, A_{i+7}). Three words will be gained: *Twos*, *TwosA*, and *TwosB*. The results are stored in register *Twos* (sum) and temporary register called *FoursB* (carry).

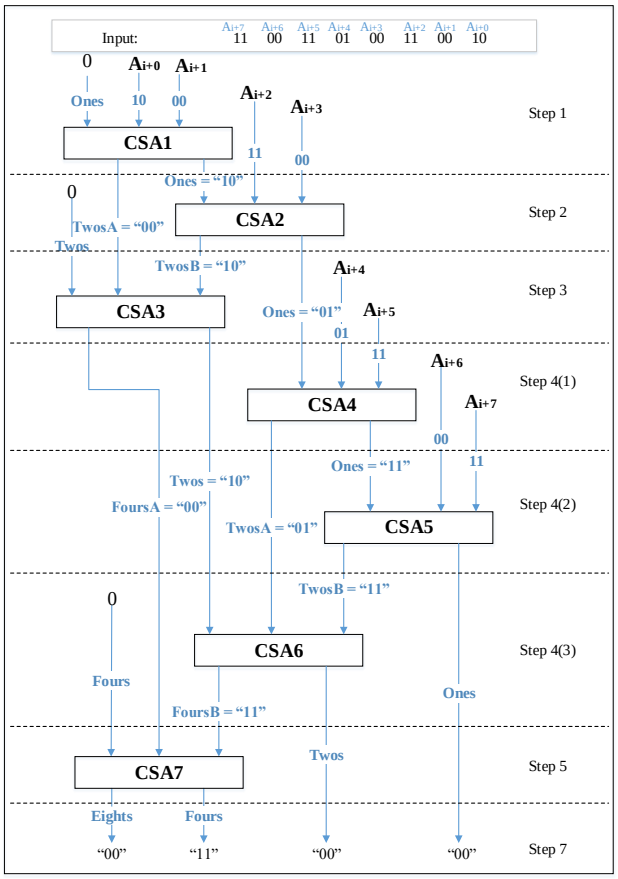

Figure 5. Harley-Seal Technique

5) So far, we have obtained three words containing third LSBs (*Fours*, *FoursA*, and *FoursB)*. Once again, these words are summed. The final sum result is stored in the *Fours* register, while carry bits are stored in the *Eights* register.
6) Increment counter *i* by 8 and repeat steps (1-6) until all the words are iterated through.
7) Once the loop terminates, the population count will be equal to (3):

$$c = 8 * \text{popcnt}(Eights) + 4 * \text{popcnt}(Fours) + 2 * \text{popcnt}(Twos) + \text{popcnt}(Ones). \quad (3)$$

It is worth mentioning that population count equation can be implemented using one of the count functions (e.g., Naïve, NTAD, WWG). Using the following stream bit example: 1100 1101 0011 0010, segmented into eight words (2-bit each) and performing the Harley-Seal on it, the final population count is calculated using (3) as shown below:

popcnt = (8 * popcnt (*Eights*)) + (4 * popcnt (*Fours*)) + (2 * popcnt (*Twos*)) + (1 * popcnt (*Ones*)) = (8*0) + (4*2) + (2*0) + (1*0) = 8

In the mentioned implementation, segments of eight words were used. However, this can be extended to say that the Harley-Seal method works for segments of 2^n words, where n = 3, 4, 5,…(8, 16, 32, …). As a result, $2^n - 1$ CSA operations/instances are required and one call to a popcount/count function (e.g., Wilkes-Wheeler-Gill) [7].

After observing the significance of the addition process within the algorithm, it was concluded that faster adders could enhance the performance of the Harley-Seal speed wisely. As stated earlier, there has been a lot of recent development in the approximation field, more specifically in approximate adders. One of the most efficient approximate adders currently available is the ETAII [10], as it has a reasonable delay compared to exact adders where most of their delay rises from carry propagation (critical path). Hence, additional aspects of ETAII will be provided next.

B. Error Tolerant Adder Type II (ETAII)

ETAII is a further development of the ETAI [12]. Even though the ETAI reduces the overall delay and consumed power by disregarding the carry propagation in the approximate portion while proceeding with standard addition to the MSBs, it faces larger error rates for small figure inputs. In ETAII, carry propagation is not eliminated entirely as its predecessor ETAI.

ETAII divides the carry propagation trail into several short trails and ends the carry propagations within the short trails simultaneously as shown in Figure 6. ETAII divides an N-bit adder into M blocks where M ≥ 2. Each block consists of N/M bits and is computed in two different circuitries; carry generator and sum generator circuits. Those elements are implemented using standard design. The first circuit (i.e., the carry generator) produces the carry-out signal without needing any previous carry signal, and the second circuit (i.e., the sum generator) utilizes a carry-in signal from a prior block to produce its sum. Therefore, the sum generator block is presented by 1-bit adder while carry generator block is implemented using the carry look-ahead adder (CLA). Thus, carry propagations do not lie in the entire adder structure, but instead occur between two adjacent blocks. The power consumption of the ETAII adder is reduced due to the restraints put on the carry propagation delay. Moreover, the adder's speed is enhanced. During the addition of binary numbers, the carry signal is provided from the previous input bits. In case the signal is generated in the LSB and propagated to the current bit position, which is considered as a worst-case scenario, then an increased amount of power and time are expended in the carry propagation trail. However, the occurrence of the previous case is infrequent [12].

As an example, take two 8-bit streams (N=8); A=11001101 and B= 00110110. Add them using ETAII with blocks of four (M=4), meaning that the inputs of carry and sum generators are 2-bit length. The approximate sum=0 11000011 (195), while the exact sum=1 0000 0011 (259). The error rate in this case is 25%.

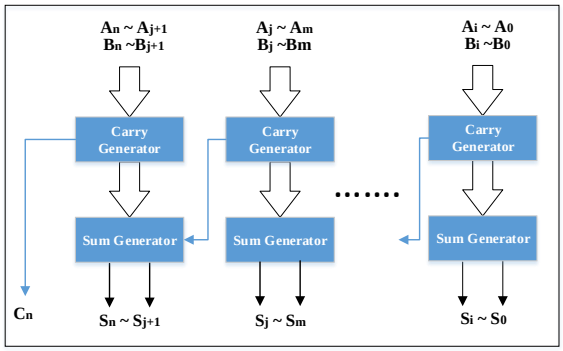

Figure 6. ETAII Adder

4. PROPOSED METHODOLOGY

Harley-Seal utilizes CSAs, and to achieve the objective of this paper, this adder will be replaced with ETAIIs. However, an obstacle is faced due to incompatibly issues, as vectorized Harley-Seal needs word values of both sum and carry, not single bits of each. Acknowledging the fact that ETAII provides a single carry out bit with a sum word, when combined with the Harley-Seal it will provide extremely incorrect results. For example, considering the following 64-bit stream 0101 0100 0101 1010 1001 0101 0101 0001 1010 0001 1101 0100 0101 0001 0001 0011, using the original Harley-Seal will result in 27 ones.

On the other hand, taking the same bit stream and using ETAII instead will result in 18 ones. Therefore, to overcome this problem, ETAII will be modified as follows. Since the carry calculation is necessary, an additional output register will be added called "Carry Register". This necessity considers that internal carries are used for the next step's calculation and not within the same step. At the end, the architecture of ETAII will be modified as shown in Figure 7. The sum generator and carry calculation for 1-bit are shown by (4) and (5), respectively.

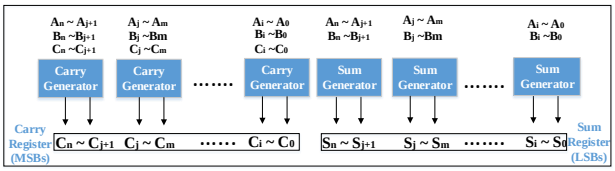

Figure 7. Modified ETAII

$$S_i = A_i \oplus B_i \quad (4)$$

$$C_i = (A_i \wedge B_i) \oplus ((A_i \oplus B_i) \wedge C_i) \quad (5)$$

To further improve the speed, we opted to approximate the Harley-Seal by applying the modified ETAII to it. Since the current exact Harley-Seal considers previous sum into the current addition, this step is modified so that previous summation is considered only into the carry calculations. This new approximate Harley-Seal with modified ETAII architecture noted as AHS-METAII is shown in Figure 8 with an example.

Based on the modified ETAII, the approximate Harley-Seal is implemented on an N-bit stream input, where N is segmented into W vectors. The larger W gets, the more the accuracy decreases. As an example, taking the previously presented 64-bit stream and applying the approximate Harley-Seal with modified ETAII on it, where the input is segmented into four words, gives the hamming weight as 26. On other hand, if the same input is segmented into eight words, the hamming weight will result in 21. As can be seen, the accuracy has decreased as demonstrated in the next section.

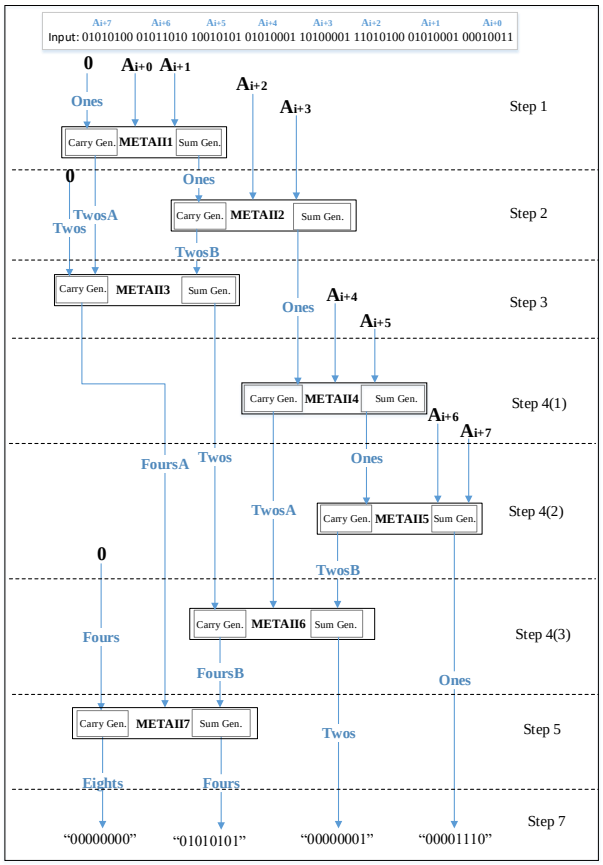

Figure 8. Approximate Harley-Seal with Modified ETAII

The approximate Harley-Seal follows these steps with eight words segments where initially: $i = 0$, $c = 0$, *Ones* = 0, *Twos* = 0, *Fours* = 0, *Eights* = 0.

1) Two new words (A_i, A_{i+1}) are loaded into the sum and carry generators of our METAII (i.e., Modified ETAII) with *Ones* word to provide the carry bits stored in the temporary register *TwosA*. While the sum is calculated in the sum generator with the two inputs only (i.e., A_i, A_{i+1}) and loaded to *Ones* register.
2) The same process is repeated with the next inputs (A_{i+2}, A_{i+3}) loaded in METAII where carry calculation is done with carry in equal to *Ones*, then load the result to *TwosB* temporary register. After that, the sum is calculated with the two inputs only then overrides *Ones* register.
3) Currently, three words with the second LSBs are created (*Twos*, *TwosA*, and *TwosB*) and loaded in METAII. The result of the sum is stored in the *Twos* register, while carry is stored in a temporary register called *FoursA*.

4) Steps (1-3) are repeated with (A_{i+4}, A_{i+5}) and (A_{i+6}, A_{i+7}), three words will be gained (*Twos*, *TwosA*, and *TwosB*). The sum is stored in the *Twos* register, while carry is stored in a temporary register called *FoursB.*
5) Currently, three words with the third LSBs are created (*Fours*, *FoursA*, and *FoursB*). The process is repeated for these three values. The sum result is stored in the *Fours* register, while carry is stored in a temporary register called *Eights*.
6) Increment *i* by 8 and repeat until all the words are iterated through.
7) Once the loop terminates, the population count will be equal to (3): *c* = 8 * popcount (*Eights)* + 4 * popcount (*Fours)* + 2 * popcount (*Twos)* + popcount (*Ones).*

The same previous stream bit example is presented in Fig 8 to illustrate the new algorithm segmented into eight words (8-bit each):
01010100 01011010 10010101 01010001 10100001 11010100 01010001 00010011

After performing the approximate Harley-Seal, one last step is required to calculate the final population count that is done below using (3):
popcnt = (8 * popcnt (*Eights*)) + (4 * popcnt (*Fours*)) + (2 * popcnt (*Twos*)) + (1 * popcnt (*Ones*)) = (8*0) + (4*4) + (2*1) + (1*3) = 21

5. EXPERIMENTAL SETUP AND RESULTS

This section discusses the experimental setup and specific simulation tools used to find the effectiveness of the proposed methodology under certain metrics (i.e., speed, power and area). After that, experimental results are presented and analyzed.

A. Experimental Setup

Different classical bit-counting techniques were implemented in order to show the effectiveness of Harley-Seal technique. Then, AHS-METAII was implemented using 64-bit word for the input. Additionally, the AHS-METAII was segmented into different number of words (i.e., two, four, eight and sixteen) to study the effect of varying the number of the words on different performance metrics. For each number of word implementation, different count function (i.e., Naïve, NTAD and WWG) is used for calculating the final population count value using (3), in order to determine the best implementation in terms of count function as well as the number of segmented words.

To achieve the research goals, the design flow shown in Figure 9 was applied. Similar design flows were used in other studies as [24-28].

The design steps were as follows:

- The design was implemented at the register transfer level (RTL) using Verilog™ (i.e., hardware description language). The Verilog implementation was verified using ModelSim™ [29]. ModelSim is a dynamic simulation tool,

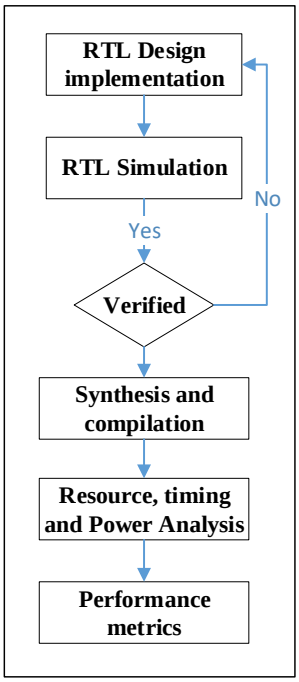

Figure 9. Design flow

which provides wave-files capturing the node activity used to compute the power dissipation of the design.
- The design was synthesized and compiled using the Altera software package Quartus-II [30].
- For Quartus-II synthesis, timing constraints were used during the compiling process.
- The design underwent through the following analysis using Quartus-II:
  - Timing analysis reported the maximum frequency of the design. All designs were compiled with clock constraints of 50 MHz.
  - Resource utilization analysis showed the number of logic elements (LEs) and the type (i.e., combinational, register logic, or both) used in the design [25]. LE is the smallest unit of logic in the Altera architecture; it is compact and facilitates efficient logic utilization.
  - Power analysis computed the average power of the design. The computed power was the dynamic core power that consists of combinational, register, and clock. The power required by node activities was extracted from the value change dump (VCD) files generated by ModelSim simulations. This approach for computing power was used in other works, such as references [24, 25].
- Performance metrics including accuracy, delay, power and area were calculated based on the implementation results in order to show the effectiveness of the proposed methodology over other techniques.

B. Experimental Results

This subsection summarizes the performance effectiveness of the proposed methodology in terms of different metrics (i.e., accuracy, delay, power and area). The Performance of AHS-METAII proposed methodology implemented with different number of words (i.e., 2, 4, 8, 16) and different count functions (i.e., Naïve, NTAD and WWG) for final population count step using (3) is shown in Table II. Exact Harley-seal approach with NTAD count function chosen to be compared with the approximate Harley-seal approach as it demonstrates almost the best results as will be shown below in Table III.

Our results presented in Tables II and III and Figures (10-16) validate the effectiveness of our proposed methodology in ways we shall explain subsequently in the following subsections.

TABLE II. PERFORMANCE METRIC COMPARISON FOR PROPOSED METHODOLOGY WITH DIFFERENT NUMBER OF WORDS AND COUNT FUNCTIONS.

Proposed approach with different words	Count function	Accuracy	Delay (ns)	LEs	Power (mW)
Approximate Harley-Seal (AHS-METAII) (2 words)	Naïve	100%	19.091	202	4.66
	NTAD	100%	18.818	182	4.7
	WWG	100%	24.588	265	6.99
Approximate Harley-Seal (AHS-METAII) (4 words)	Naïve	95%	18.543	151	3.61
	NTAD	95%	18.457	134	3.37
	WWG	95%	19.635	198	6.65
Approximate Harley-Seal (AHS-METAII) (8 words)	Naïve	91%	17.838	136	3.86
	NTAD	91%	17.498	127	2.8
	WWG	91%	18.875	162	4.47
Approximate Harley-Seal (AHS-METAII) (16 words)	Naïve	86%	17.167	99	2.56
	NTAD	86%	17.440	97	2.33
	WWG	86%	17.771	97	2.21

TABLE III. EXACT HARLEY-SEAL METHOD WITH APPROXIMATE HARLEY-SEAL METHOD USING 8 WORDS AND NTAD COUNT FUNCTIONS

Type	Accuracy	Delay (ns)	LEs	Power (mW)
Exact (HS-CSA) with NTAD (8 words)	100%	17.989	151	3.81
Approximate (HS-CSA) with NTAD (8 words)	91%	17.498	127	2.8

1) Accuracy

For all designs, the accuracy was computed for 10 different random bit streams. To calculate the accuracy level, each output value was contrasted with the exact value after the simulation. Table II shows the accuracy for all implementations.

The results show the maximum accuracy level of our AHS-METAII is 100% seen in the two words implementation, which is equal to exact Harley-Seal approach (HS-CSA). Implementing the Harley-Seal approach with ETAII (HS-ETAII) without modifying it, results in decreasing the accuracy to unacceptable levels (i.e., 56%) as discussed previously in Section 4. However, after modifying the ETAII, it demonstrated good accuracy levels with different number of words. It is noticeable that, sixteen words implementation shows the lowest accuracy. The AHS-METAII methodology with four and eight words shows an acceptable level of accuracy as they are above 90%. The relationship between the number of words and accuracy level is inversely proportional as the proposed algorithm does not consider the previous data when calculating the sum. As number of words increases, the word size decreases resulting in ignoring more carry-in values which reduces the accuracy as shown in Figure 10.

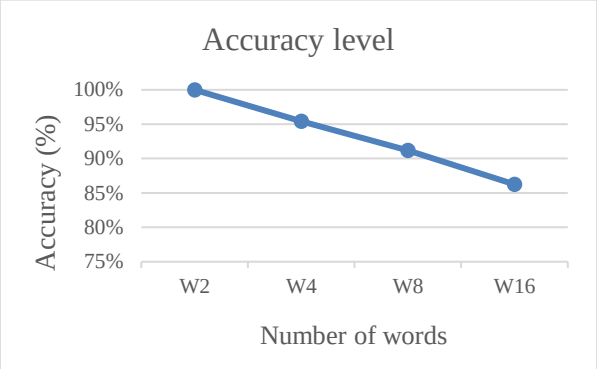

Figure 10. Accuracy for AHS-METAII segmented into different number of words

2) Delay (Speed)

Table II shows the delay calculated for all the AHS-METAII methodology implementations with different number of segmented words and count functions.

Figure 11 represents the delay of AHS-METAII with different count functions and number of words. It is noticed that applying NTAD with the AHS-METAII approach results in minimum delay followed by Naïve. However, using it with WWG is not recommended as it demonstrated the maximum delay among all word's implementations. In general, increasing number of words, decreases the average delay as shown in Figure 12. It is clear that sixteen words implementation has the minimum delay among all implementations followed by eight words as the process is divided into smaller bit streams working on parallel manner which result in better delay results. Even though, the proposed methodology with sixteen words shows the best in terms of delay, however, this design is not recommended as it has the lowest accuracy. So, the best one to be considered is the proposed one with eight words using NTAD count function, as it is faster than the exact Harley-seal approach by 3% as shown by Table III.

Additionally, it is observed that for all experiments the performance results are sensitive to the random data values in the sense that the distribution of ones across the bit-stream affects the addition processes, which in turn affect the speed of the pop-count methods correspondingly.

3) Area utilization

Design area is represented by the total number of Logic Elements (LEs) and calculated for all designs as illustrated in Table II. To get a clear view of the area utilized by the

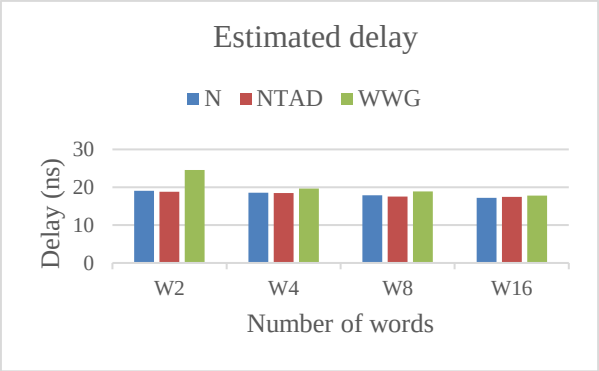

Figure 11. The delay of AHS-METAII using different count functions and number of words.

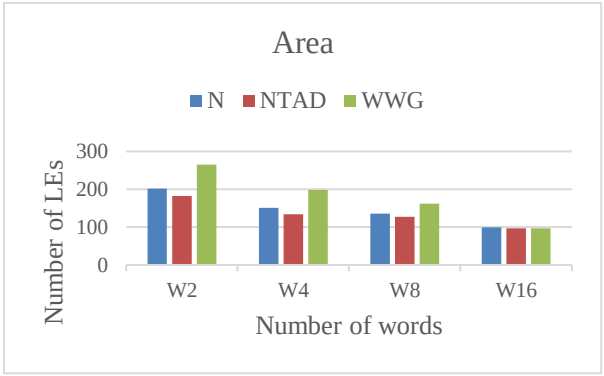

Figure 13. The utilized area of AHS-METAII using different count functions and number of words.

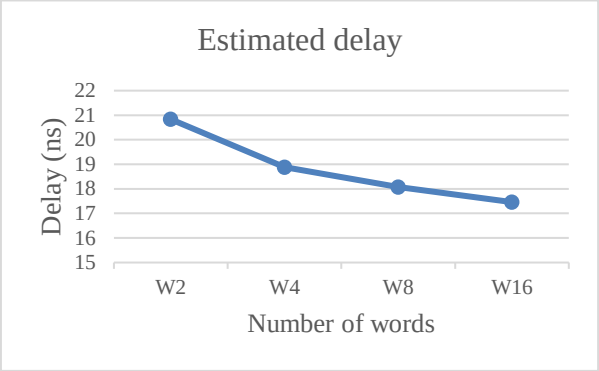

Figure 12. The average delay of AHS-METAII using different number of words.

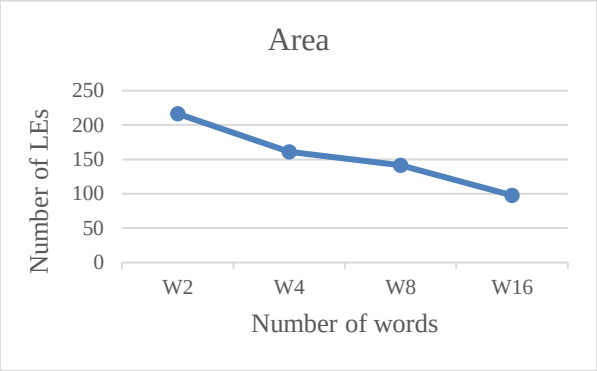

Figure 14. The average utilized area of AHS-METAII using different number of words.

AHS-METAII proposed methodology, Figure 13 plots all the designs with different words numbers and count functions. It is observable that our AHS-METAII using NTAD count function has the least number of LEs using different number of words. Overall, the number of LEs decays as number of words doubles due to halving the size of registers (in number of bits) as it is clear in Figure 14. The best design of our proposed methodology is AHS-METAII using NTAD with eight words as it utilizes 16% less LEs than the exact Harley-seal as presented in Table 3. This reduction in the utilized area is due to the reason that our proposed methodology uses one less XOR gate to calculate the sum. However, there was an increase in the area when compared to the original ETAII because a carry register equal in size to the sum register, instead of the one-bit carry that was previously needed. Although the design with sixteen words utilizes a smaller number of LEs, it is not recommended as stated before because of the low accuracy achieved.

4) Power consumption

Dynamic power consumption which consists of combinational and sequential power is calculated for AHS-METAII methodology with different words and count functions as presented in Table 2. Figure 15 represents the power consumption of AHS-METAII proposed with different count functions and number of words. It is clear that among all word's implementations, AHS-METAII with NTAD approach recorded approximately the minimum power consumption followed by AHS-METAII proposed with Naïve. Moreover, power consumption decreases as the number of words increases, until reaching the minimum value at sixteen words implementation as demonstrated in Figure 16. This decreasing trend is due the number of combinational logic elements as they decrease with doubling the number of words. The synthesis tool works better with larger number of words as it tries to reduce connections and optimize the design in an efficient way. It is worth mentioning that the best implementation of our AHS-METAII methodology (i.e., AHS-METAII with eight words using NTAD) is consuming less power than the exact Harley-seal approach by an average of 27% as it is noted in Table III. This shows the performance of applying approximation methods with the pop-counting techniques in reducing the consumed power. Although using sixteen words consumes much less power than the exact Harley-Seal approach, it recorded less accuracy as illustrated in Table II.

6. CONCLUSION

This work analyzed the problem of computing the hamming weight of bit streams. A review and evaluation of the existing approaches revealed that they exposed either scalar parallelism or vector parallelism. Therefore, a new approximate approach that exposes vector parallelism but alters the internal logic gates of the approximate adder was proposed.

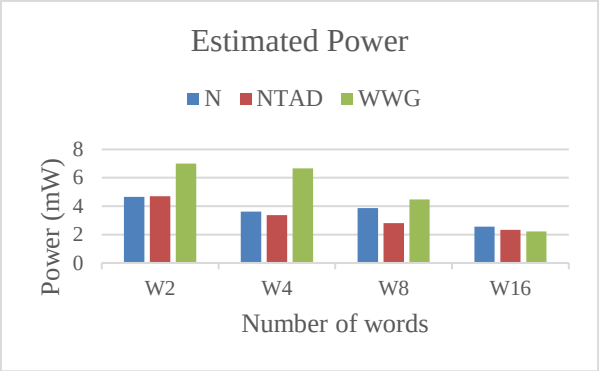

Figure 15. The power of AHS-METAII using different count functions and number of words.

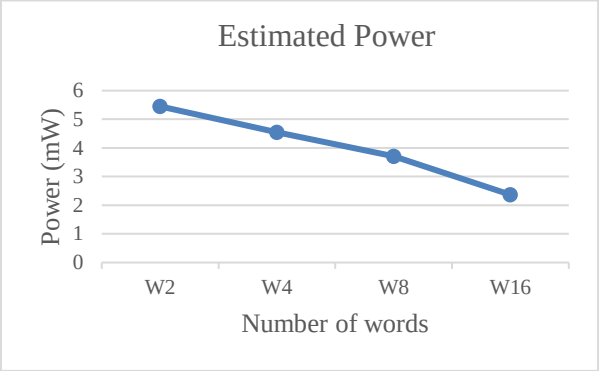

Figure 16. The average power of AHS-METAII using different number of words.

This implementation was useful in error-tolerant applications. The evaluation showed that the proposed hybrid solution (AHS-METAII) outperformed the exact Harley-Seal approach (HS-CSA) in all metrics when eight words were used as it offered overall enhanced performance suitable for error tolerant applications with a 27% dynamic power reduction, 3% delay reduction, and 16% area reduction along with a slight decrease (9%) in the accuracy level. Applying sixteen words in the proposed approach resulted in better performance than eight words. However, the accuracy decreased by 14%.

In the future, the proposed methodology will be tested on a larger bit stream (128-bit) to observe the results of the methodology when the parameters (e.g., number of words and popcount method used) are altered. Additionally, different approximate adders can be implemented and tested using this approach. Other possibilities include implementing the proposed approach using different computer architectures, such as Single Instruction Multiple Data (SIMD), or using pipelining to enhance the delay.

REFERENCES

[1] Berkovich, S., Lapir, G.M., Mack, M., "A bit-counting algorithm using the frequency division principle," Software: Practice and Experience, 2000, 30, (14), pp.1531-1540.

[2] Gomuffkiewicz M., Kutyfflowski M., "Hamming Weight Attacks on Cryptographic Hardware—Breaking Masking Defense," InEuropean Symposium on Research in Computer Security, Berlin, Heidelberg, 2002, pp. 90-103.

[3] Suciu A., Cobarzan P., Marton K., "The never ending problem of counting bits efficiently," In2011 RoEduNet International Conference 10th Edition: Networking in Education and Research, 2011, pp. 1-4.

[4] Manku GS., Jain A., Das Sarma A., "Detecting near-duplicates for web crawling," InProceedings of the 16th international conference on World Wide Web, 2007, pp. 141-150.

[5] Nugrahaeni RA., Mutijarsa K., "Comparative analysis of machine learning KNN, SVM, and random forests algorithm for facial expression classification," In2016 International Seminar on Application for Technology of Information and Communication (ISemantic), 2016, pp. 163-168.

[6] Baldi P., Hirschberg DS., Nasr RJ., "Speeding up chemical database searches using a proximity filter based on the logical exclusive OR," Journal of chemical information and modeling, 2008, 28;48, (7), pp.1367-1378.

[7] Muła W., Kurz N., Lemire D., "Faster population counts using AVX2 instructions," The Computer Journal. 2018,61,(1), pp.111-120.

[8] Liang J., Han J., Lombardi F., "New metrics for the reliability of approximate and probabilistic adders," IEEE Transactions on computers, 2012,62,(9), pp.1760-1771.

[9] Kahng AB., Kang S., "Accuracy-configurable adder for approximate arithmetic designs," InProceedings of the 49th Annual Design Automation Conference, 2012, pp. 820-825.

[10] Jiang H., Han J., Lombardi F., "A comparative review and evaluation of approximate adders," InProceedings of the 25th edition on Great Lakes Symposium on VLSI, 2015, pp. 343-348.

[11] Warren Jr HS., The quest for an accelerated population count, Beautiful code: leading programmers explain how they think, 2007, pp.147-60.

[12] Zhu N., Goh WL., Wang G., Yeo KS., "Enhanced low-power high-speed adder for error-tolerant application," In2010 International SoC Design Conference 2010, pp. 323-327.

[13] Sun C., del Mundo CC., "Revisiting POPCOUNT Operations in CPUs/GPUs," InACM Student Research Competition Posters at the ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis, 2016.

[14] Morancho E., "A hybrid implementation of Hamming weight," In2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2014, pp. 84-92.

[15] Wilkes MV., Wheeler DJ., Gill S. The preparation of programs for an electronic digital computer,1958.

[16] El-Qawasmeh E., "Beating the popcount." International Journal of Information Technology, 2003,9,(1), pp.1-8.

[17] Jiang H., Liu C., Liu L., Lombardi F., Han J., "A review, classification, and comparative evaluation of approximate arithmetic circuits," ACM Journal on Emerging Technologies in Computing Systems (JETC), 2017, 13, (4), pp.1-34.

[18] Verma AK., Brisk P., Ienne P., "Variable latency speculative addition: A new paradigm for arithmetic circuit design.," InProceedings of the conference on Design, automation and test in Europe, 2008, pp. 1250-1255.

[19] Mahdiani HR., Ahmadi A., Fakhraie SM., Lucas C., "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Transactions on Circuits and Systems I: Regular Papers, 2009, 57, (4), pp. 850-862.

[20] Du K., Varman P., Mohanram K., "High performance reliable variable latency carry select addition," In2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012, pp. 1257-1262.

[21] Ebrahimi-Azandaryani F., Akbari O., Kamal M., Afzali-Kusha A., Pedram M., "Block-based Carry Speculative Approximate Adder for Energy-Efficient Applications," IEEE Transactions on Circuits and Systems II: Express Briefs, 2019.

[22] Barman J, Kumar V., "Approximate carry look ahead adder (cla) for error tolerant applications," In2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), 2018, pp. 730-734.

[23] Dutt S., Nandi S., Trivedi G., "Analysis and design of adders for approximate computing," ACM Transactions on Embedded Computing Systems (TECS), 2017, 17, (2), pp. 1-28.

[24] Mohd BJ., Abed S., Alouneh S., "Carry-based reduction parallel counter design," International Journal of Electronics, 2013, 100, (11), pp. 1510-1528.

[25] Abed S., Jaffal R., Mohd BJ., Alshayeji M., "FPGA modeling and optimization of a Simon lightweight block cipher," Sensors, 2019, 19, (4), pp. 1-28.

[26] Abed S., AlKandari M., AlRasheedi H., Ahmad I., "FPGA Implementation of Enhanced JPEG Algorithm for Colored Images," International Journal of Computing and Digital Systems, 2020, 1, 20, (1), pp. 13-22.

[27] Hayajneh T., Ullah S., Mohd BJ., Balagani KS., "An Enhanced WLAN Security System with FPGA Implementation for Multimedia Applications," IEEE Systems Journal, 2015, 11, (4), pp. 2536-2545.

[28] Mohd BJ., Hayajneh T., Abed S., Itradat A., "Analysis and modeling of FPGA implementations of spatial steganography methods," Journal of Circuits, Systems, and Computers, 2014, 23, (02), pp. 1450018:1-1450018:26.

[29] ModelSim-Altera Software Simulation User Guide. Retrieved from: https://www.altera.co.jp/ja_JP/pdfs/literature/ug/ug_gs_msa_qii.pdf, accessed (April 2018).

[30] Altera Quartus II Hnadbook version 13.1, volume 1: Design and Synthesis. Retrieved from: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/qts/archives/quartusii_handbook_archive_131.pdf, accessed (May 2018).

Reem Jaffal received the B.S. degree in computer engineering from College of Engineering and Petroleum, Kuwait University, Kuwait, in 2014. Her M.Sc. degree in computer engineering was received in 2018 from College of Engineering and Petroleum, Kuwait University, Kuwait. She is currently a Research Assistant at Kuwait University. Her current research interests include hardware design, power/energy optimization and cryptography.

Sa'ed Abed received his B.Sc. and M.Sc. in Computer Engineering from Jordan University of Science and Technology in 1994 and 1996, respectively. In 2008, he received his Ph.D. in Computer Engineering from Concordia University, Canada. Currently, he is an Associate Professor in the Department of Computer Engineering at Kuwait University. His research interests include VLSI Design, formal methods and Image Processing.

Imtiaz Ahmad has a B.Eng in electrical engineering, from University of Engineering & Technology, Lahore, Pakistan in 1984, M. Eng. in electrical engineering from King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia in 1988, and Ph.D. in computer engineering from Syracuse University, Syracuse, New York, USA, in 1992. His research interests are in design automation of digital systems, parallel and distributed computing. He is currently a Professor with the Computer Engineering Department at Kuwait University, Kuwait.

Mahmoud Bennaser received his B.Sc. in Computer Engineering from Kuwait University in 1999. His M.Sc. in Computer Engineering was received from Brown University in 2002. In 2008, he received his Ph.D. in Computer Engineering from University of Massachusetts. Currently, he is an Assistant Professors in the Department of Computer Engineering at Kuwait University. His research interests include VLSI Design, CMOS, Circuit Analysis and Integrated Circuits