Optimization of the Hamming Code for Error Prone Media

Eltayeb S. Abuelyaman and Abdul-Aziz S. Al-Sehibani
College of Computer and Information Sciences
Prince Sultan University

Summery

This paper proposes an optimized version of the Hamming code. The paper begins with identifying a seed that can be used as basis to generate Hamming codes. It then shows that starting with a message that is only two bits long, along with three parity bits; one can add extra message bits incrementally. This approach results in significant reduction in the circuitry and provides flexibility to designers. Only the H(7,4) was sufficient to show an improvement of 33 percent in the hardware circuitry. Similar gains for larger sizes are also argued for both hardware and software implementations.

Key words:
Hamming code, parity bits, distance, error correction.

1. Introduction

Transmission errors have various causes including static on devices, environmental interferences or scratches on electronic data storage media. When a channel is error prone, error correction would likely result in better throughput compared to retransmission. However, correction is superior to the multiple transmissions modes used for space vehicle communications. For these modes, each nibble of data is sent several times and when received, a majority function is used to select the version with the highest frequency. The throughput of error correction codes is the main reason they are favored in storage devices, digital subscriber lines and mobile communications.

In general, if the probability of a single bit error is denoted by **e**, then the probability of receiving an error free nibble is $\mathbf{(1- e)^4}$. The key for whether or not the Hamming code is preferable over other modes of retransmission depends on the value of **e**. As stated above, the Hamming code provides an error correction mechanism that is used by many applications including dynamic random access memory chips and satellite communications devices. The Hamming code has been proposed for deeply faded wireless asynchronous transfer mode networks [1]. The authors argued that the Hamming code is a better alternative to solutions suggested in [2-5]. In the same paper, the authors proposed a typical ROM implementation of the Hamming code showing that only 25% of the typical ROM implementation circuitry would be necessary.

We will briefly review the Hamming code in the next section. The rest of the paper is organized as follows. An optimized version of the Hamming code is given in section 3. Section 4 presents analysis of the parity bits. The Hamming code H(7,4) as a basis for larger codes is discussed in section 5 and the conclusion is in section 6.

2. Review of the Hamming Code.

In 1947 R. W. Hamming proposed a thesis in which he declared that if a message **M** with **m** bits is to be transmitted, then, a code **C** with **c** bits must be generated. The length of **M** is related to that of **C** with the following equation:

$$c = m + p \leq 2^p - 1 \qquad (1)$$

where **p** is the number of redundant bits, also called parity bits [6]. Without loss of generality, if we take **m** to be equal to **4**, then solving equation 1 will result in **p** and **c** equal to **3** and **7** respectively.

A Hamming code is computed using either **even** or **odd** parity. Table 1 displays the binary representation of the first 7 non-zero decimal digits. Here, the first row shows the decimal digits 3, 5, 6 and 7 as subscripts of column headers for the 3rd, 5th, 6th, and 7th columns from the right respectively. There are no column headers for the first, second and fourth columns from the right simply because there are not needed at this point. Rows 2 though 4 give the corresponding binary representations of these subscripts. The row headers $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_4}$ are the three parity bits representing the Hamming parity **P**.

Table 1. Relationship between parity and message bits

parity \ code	M_7	M_6	M_5		M_3		
P_1	1	0	1	0	1	0	1
P_2	1	1	0	0	1	1	0
P_4	1	1	1	1	0	0	0

We will assume that the parity bits are interleaved with the bits of the message **M**. To compute the parity bits, Hamming used the respective row for each. The values for $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_4}$ are given by the following equations where the symbol @ represents the ***exclusive or*** operation:

$\mathbf{P_1=M_7@M_5@M_3@\ P_1}$ (2)
$\mathbf{P_2=M_7@M_6@M_3@\ P_2}$ (3)
$\mathbf{P_4=M_7@M_6@M_5@\ P_4}$ (4)

Observe that the message bits used for computing a parity bit are the column headers of the nonzero entries on the row represented by that parity bit. If a column is without a header, it represents a parity bit that is associated with the column number. In the figure, columns 1, 2, and 4 are without headers, hence they represent parity bits $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_4}$. We assume an initial value of 0 for each parity bit to enable evaluation of these equations. Obviously, building a typical circuit to generate these parity bits would require **9** dual input ***exclusive or*** gates. In the next section we will introduce optimization of the Hamming code.

3. Optimized Hamming for the transmission end

Our optimized Hamming approach depends on the nonzero bits in a code. It represents another way of computing the Hamming parity. At best, no computation is necessary and at worst, the number of operations is equal to that of the regular Hamming approach. For example, if only one bit in a message is equal to 1 then the parity for the whole message is given by the binary representation of the subscript of that message bit. On the other hand, if all the bits of a message are equal to one, then the number of operations needed to compute the parity is equal to the number of operations needed to compute the Hamming code. An example is given next to clarify these concepts.

3.1 Computing the parity at the transmitting end

Without loss of generality and for a message $\mathbf{M} = \mathbf{M_7M_6M_5M_3} = \mathbf{1010}$, we can compute **P** by simply performing bit-by-bit ***exclusive-or*** operations on the binary representation of the subscripts of the nonzero bits of **M**.

Therefore, $\mathbf{P} = \mathbf{M_7@M_5}$
$= \mathbf{7\ @\ 5}$
$= \mathbf{111\ @\ 101}$
$= \mathbf{010}$

The transmitted code (message and parity) will therefore be equal to:

$\mathbf{M_7M_6M_5P_4M_3P_2P_1} = \mathbf{1010010}$ (5)

We will discuss computing the error bits at the receiving end next.

3.2 Computing the error bits at the receiving end

At the receiving end, the same process is used but not for computing parity bits, rather it is used for computing error bits. First, we will review the Hamming equations for computing error bits.

3.2.1 Hamming approach

The error $\mathbf{E} = \mathbf{e_1e_2e_3}$ is computed using the following equations:

$\mathbf{e_1=P_4@M_5@M_6@M_7}$ (6)
$\mathbf{e_2=}P_2\mathbf{@M_3@M_6@M_7}$ (7)
$\mathbf{e_3=}P_1\mathbf{@M_3@M_5@M_7}$ (8)

The next step is to use our optimized version of Hamming approach in computing the error bits.

3.2.2 Optimized Hamming

We will first consider the received code to be error free. Then the Right Hand Side (**RHS**) of equation (5) gives the following:

$\mathbf{M_7@M_5@P_2= 000}$ (9)
$\mathbf{M_5@P_2\ = M_7}$ (10)
$\mathbf{M_7\ @P_2\ = M_5}$ (11)
$\mathbf{M_7\ @M_5 = \ P_2}$ (12)

Equation (9) includes only message and parity bit variables that are equal to **1** in the originally transmitted code. However, if any of the bits of the code is flipped during transmission, equation (9) will not be equal to **000**. Remember, only message and parity bit variables that are equal to **1** were included in computing the parity at the transmitting end.

Clearly, a single bit error would change a **0** to **1** or vice versa. If a **0** is changed to **1** then, **M6, P_4, M_3 or P_1** would be ***exclusive or***-ed with the Left Hand Side (**LHS**) of equation (9) at the receiving end. The result would give the binary equivalence of the subscript of the faculty bit on the **RHS**. Consequently, the **RHS** would be equal to the binary representation of the offending bit. On the other hand, if a **1** is toggled to **0** during transmission, then one of the original variables (**M_7, M_5,** or **P_2**) of equation (9) would not be included in the computation of the parity at the receiving end. Hence, the **RHS** of equation (9) would not be equal to **000**. If we ***exclusive or*** both sides of equation (9) with **M_7, M_5,** or **P_2**, we will end up with equation (10), (11) or (12) respectively. The **RHS** of each of these equations gives the dropped variable. The binary representation of the dropped variable gives the Hamming parity for the received code.

In any case, the value on the **RHS** of equation (9) will either be equal to **000** or something else. Henceforth, these values are referred to as the error **E**. Once a gain, a comparison between equations 6, 7 and 8 on the one hand, and equations 9, 10, 11 and 12 on the other hand reveals the advantages of the optimized Hamming approach.

Clearly, the error **E** may take the equivalence of any decimal value from **0** to **7.** Therefore, the following conclusions can be drawn.

1. If **E** is equal to **0** or equal to a power of **2**, then no action should be taken and the received message section is intact.
2. Otherwise the value of **E** will determine the subscript of the bit in **M** that should be flipped to restore the correct value.

The next step is to introduce our proposed implementation of the optimized Hamming. However, we will furnish for that by a facilitating theorem right after analyzing the parity bits.

4. Analysis of the parity bits

The first row on Table 2 shows the decimal equivalence of the 16 possible combinations of 4 bit binary numbers. Each of the 16 possibilities represents a message. The second row shows the corresponding Hamming parity bits in their decimal digit forms.

Table 2. Decimal equivalence of 4 bit codes and their parity bits

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	3	5	6	6	5	3	0	7	4	2	1	1	2	4	7

This table shows that the parity for a message represented by the number **3** is identical to that of a message represented by the number **4**. The same conclusion is true about messages represented by the numbers **11** and **12**. A closer look at the analysis of equations (10) through (12) will justify such equality. We will comment on two interesting observations here. The first is the equidistance between (**3** and **11)** on the one hand, and (**4** and **12**) on the other hand. This distance is equal to $\mathbf{2^{n-1}}$ where **n** in this case is the number of bits (4). That is, if we replaced the Most Significant Bit (**MSB**) of the 4 bit binary code of the number **3** with the digit **1** we will end up with the number **11** and the same is true for the numbers **4** and **12**. The second observation is that the parity for the number **3** (**4**) is the complement of the parity for the number **11** (**12**). Though trivial, these observations will be helpful in the presentation of the theorem. However, we will also need the following definitions before discussing the theorem:

Definitions:

1. **S (L)** : The set of all codes of length **L** bits.
2. **S_A (L):** The subset of **S (L)** that can be used to represent the symbol **A**.
3. **C(j, k)** : A code with **j** message bits and **k** parity bits where **j +k = L.**
4. **D(C^i,C^j)** : The distance or number of bits in which $\mathbf{C^i}$ and $\mathbf{C^j}$ are different.
5. **'b** : The complement of a binary digit **b.**
6. **M_k**: A message bit where **k** takes any value in the sequence **3, 5, 6, …, m** of positive integers that are not powers of two.
7. **Parity mode:** Defines the **even** or **odd** parity used in computing the Hamming parity bits of a message.

We remind the reader that given any $\mathbf{C^1}$ and $\mathbf{C^2}$ in **S (4)**, if $\mathbf{C^1}$ = **1011** and $\mathbf{C^2}$ = **0101**, then **D** ($\mathbf{C^1}$, $\mathbf{C^2}$) = **3** because the three most significant bits are different. We also remind the reader of a well known coding theorem which states that: for any code space **S (L)**, if $\mathbf{D(C^i,C^j) > 1}$ for every **i** and **j**, where $\mathbf{i \neq j}$, then the space is capable of detecting a single error. However, if $\mathbf{D(C^i,C^j) > 2}$, then the space would be capable of correcting a single error and detecting two errors. Now we are ready to discuss the theorem.

Theorem

For any message $\mathbf{M = M_{m-1} \ldots M_5M_3}$, if:

1. The Hamming parity for **M** is equal to $\mathbf{P_q \ldots P_4 P_2P_1}$
2. **q** is a power of two
3. **m** is equal to $\mathbf{2^q - 1}$

Then the parity for $\mathbf{M_mM_{m-1} \ldots M_5M_3}$ would either be equal to $\mathbf{P_q \ldots P_4P_2P_1}$ or $\mathbf{'P_q \ldots 'P_4'P_2'P_1}$ depending on the value of $\mathbf{M_m}$ being a **0** or a **1** respectively.

To prove this theorem we need to show that if the distance between any pair of codes that are represented by $\mathbf{M_{m-1} \ldots M_5M_3\, P_q \ldots P_4\, P_2P_1}$ is at least **3** then:

1. the distance between any pair of codes represented by $\mathbf{0M_{m-1} \ldots M_5M_3\ P_q \ldots P_4P_2P_1}$ is at least **3**.
2. the distance between any pair of codes represented by $\mathbf{1M_{m-1} \ldots M_5M_3\ P_q \ldots P_4P_2P_1}$ is at least **3**.
3. The distance between any code from the first group (1) and any code from the second (2) is also at least **3**.

These conditions simply imply that appending a **0** to the **MSB** of a code at the ($\mathbf{2^q - 1}$)'s position will not change the parity bits. On the other hand, appending a **1** at the same position will force the inversion of each of the parity bits.

The simplest way to prove this theorem is by induction hypothesis. Since for a code to correct a single error, the distance **D** must be at least **3**, it follows that the code must have a minimum length of **3** bits. However, for **L** equals **3**, there are exactly two valid codes in the space **S (3)** that satisfy the minimum distance. In this case the codes are $\mathbf{C^1}$ and $\mathbf{C^2 = 'C^1}$.

A legitimate question is as follows: why are there only two valid codes in **S(3)**? The answer is given in the following example. Given the space **S(3)** = { 000, 001, 010, 011, 100, 101, 110, 111}, let us assume without loss of generality that the code $\mathbf{C^1}$ is encoded as **000**. It follows that $\mathbf{C^2}$ would be encoded as **111**. But what about the rest of the bit patterns of **S(3)** which are included between the following curl brackets {001,010,001,011,101 and 110}? Here is the key for the answer. Assume that the letters **A** and **B** are encoded using $\mathbf{C^1}$ = **000** and $\mathbf{C^2}$ = **111** respectively. Technically speaking, a receiver at the other end of transmission would recognize the letter **A** upon receiving any of the members of the subset $\mathbf{S_A(3)}$ = {**000**, 001,010, 001}. Similarly, the letter **B** would be recognized by a receiver if any of the members of $\mathbf{S_B(3)}$ = {011,101, 110, **111**} is received. In both cases, receiving a boldfaced pattern implies no error had occurred, otherwise, receiving any of their subset mates implies that a single bit correctable error has occurred.

Since **L = 3** is not challenging enough, we will use a code with **L** = **4**, or a **C (1, 3)** code. We will first prove that our hypothesis is true for **L** = **4**. Then prove the hypothesis for **L** = **11** or **C (4, 3)**. Finally, we will assume that the hypotheses is true for **L = m-1** and then prove it to be true for **L = m**, where **m** is greater than **11**.

Our **C (1, 3)**, which can be represented by $\mathbf{M_3P_4P_2P_1}$ is equal to $\mathbf{'M_30M_3M_3}$. This shows that there are only two valid codes in **S (4)**; namely **(0011)** and **(1000)** according to our theorem. Obviously the distance between these two codes is **3**, so, the hypothesis is true for **C (1, 3)**.

Our **C (4, 3)** is just another notation for Hamming code **H (7, 4).** The proof for **C (4, 3)** which is represented by $\mathbf{M_7M_6M_5M_3P_4P_2P_1}$ is as follows:

Assume the theorem is true for **C (3, 3)**, that is, the minimum distance between any pair of codes represented by $\mathbf{R_1 = M_6M_5M_3P_4P_2P_1}$ is at least **3**. Here, $\mathbf{R_1}$ represents an **S(6)** code space.

We will divide the proof into three parts. That is, we will prove that:

1. $\mathbf{D(C^i,C^j) > 2}$ where both $\mathbf{C^i}$ and $\mathbf{C^j}$ are represented by $\mathbf{R_2 = M_6M_5M_3'P_4'P_2'P_1}$ and $i \neq j$.
2. $\mathbf{D(C^i,C^j) > 2}$ where $\mathbf{C^i}$ is represented by $\mathbf{0M_6M_5M_3P_4P_2P_1}$ and $\mathbf{C^j}$ is represented by $\mathbf{1M_6M_5M_3'P_4'P_2'P_1}$, and i can be equal to j.
3. $\mathbf{D(C^i,C^j) > 2}$ where $\mathbf{C^i = 0N_6N_5N_3Q_4Q_2Q_1}$ and $\mathbf{C^j = 1M_6M_5M_3'P_4'P_2'P_1}$ and **i** can be equal to **j**

Part (1): To prove this part we need to examine $\mathbf{R_1}$. Since every pair of codes represented by $\mathbf{R_1}$ satisfies a minimum distance of at least **3**, it follows that $\mathbf{R_2}$ will also satisfy the minimum distance of at least **3**. The reason is simple, the parity bits for $\mathbf{R_1= M_6M_5M_3P_4P_2P_1}$ which are given by $\mathbf{P_4P_2P_1}$, were originally computed using **even** (**odd**) **parity mode**. If we invert each of these parity bits, then we would get the **parity mode** changed to **odd** (**even**) and the message part of $\mathbf{R_1}$ along with the inverted parity bits would be equal to $\mathbf{M_6M_5M_3'P_4'P_2'P_1}$. However, the distance between any pair of codes of the modified $\mathbf{R_1}$ would still remain at least **3**. Since the modified $\mathbf{R_1}$ is identical to $\mathbf{R_2}$, it follows that the distance between every pair of codes represented by $\mathbf{R_2}$ is also at least **3**.

Part (2): This part follows from the first. That is, appending $\mathbf{M_7 = 1}$ as the most significant digit to $\mathbf{M_6M_5M_3P_4P_2P_1}$ results in a code that is equal to $\mathbf{1M_6M_5M_3P_4P_2P_1}$**.** However**,** this code cannot retain both the same parity bits and the original **parity mode**. Either the **parity mode** has to survive at the cost of inverting the parity bits or the parity bits have to survive at the cost of accepting the flipped **parity mode**. Since for our case the **parity mode** has to remain the same, we need to complement each of the Hamming parity bits. This proves that the two forms: $\mathbf{0M_6M_5M_3P_4P_2P_1}$ and $\mathbf{1M_6M_5M_3'P4'P2'P1}$ follow the same **parity mode** and the distance between any code from the first and another from the second is at least **3**.

Part (3): For the last part, we have to consider the distance between $\mathbf{1M_6M_5M_3'P_4'P_2'P_1}$ and $\mathbf{0N_6N_5N_3Q_4Q_2Q_1}$**.** If the distance between $\mathbf{N_6N_5N_3}$ and $\mathbf{M_6M_5M_3}$ is the minimum, that is, equals to **1**, then we need to show that the distance between $\mathbf{Q_4Q_2Q_1}$ and $\mathbf{'P_4'P_2'P_1}$ is at least **2**. Without loss of generality, assume that: $\mathbf{N_6N_5 = M_6M_5}$; $\mathbf{N_3 = 1}$; and $\mathbf{M_3=0}$. Hence, using the optimized Hamming approach we can compute the parity bits for $\mathbf{M_6M_5M_3}$ and $\mathbf{N_6N_5N_3}$ as $\mathbf{M_6@M_5\ @000 = 'P_4'P_2'P_1}$ and $\mathbf{N_6@N_5@011= Q_4Q_2Q_1}$ respectively. Since: $\mathbf{N_6N_5 = M_6M_5}$ it follows that $\mathbf{Q_2Q_1}$ would be equal to $\mathbf{P_2P_1}$. Hence, the distance between the parity bits $\mathbf{'P_4'P_2'P_1}$ and $\mathbf{Q_4Q_2Q_1}$ is at least **2**. This completes the proof for the case when the distance between the message bits is **1**. When the distance between the message bits is **2** (or **3**), a similar argument will prove that the distance between their respective parity bits is at least **1**(or **0**). In any case, the overall distance (message + parity) will always be at least **3**. This completes the proof of the theorem.

Corollary 1

The parity bits for a Hamming code of length $\mathbf{2^n - 1}$ can be obtained from the set of parity bits of the Hamming code of length $\mathbf{2^{n-1} - 1}$.

Lemma 1

For a Hamming code where $\mathbf{c = m + p < 2^p - 1}$, the parity for a message of length **m** bits can be computed from the parity of the least significant (**m-1**) message bits simply by performing ***exclusive-or*** operation of the bits of the number **m** with those of the parity bits for the (**m-1**) bits section.

The proof follows directly from the proof of theorem 1, hence it is ignored.

Corollary 2

Our C(4,3) forms a basis from which any Hamming code of any length could be generated.

These corollaries are helpful in situations where a lookup table has to be stored to avoid real time computation. Instead of storing all the message and parity bits of a large Hamming code, one can simply store the table for C (4, 3) and then generate the required code size on real time quite easily. In the next section we will discuss the advantages of using C (4, 3) as basis.

5. The H (7, 4) as basis

Figure 1 show an implementation of our C(4,3) code. One can easily verify that this figure is consistent with the proposed theorem since the second level ***exclusive or*** gates will either pass or invert the outputs of the first level ***exclusive or*** gates depending on the value of $\mathbf{M_7}$. Expectedly, the first level gates produce the parity bits for codes with up to three message bits. It is important to assume that the default value for an input line is logical zero.

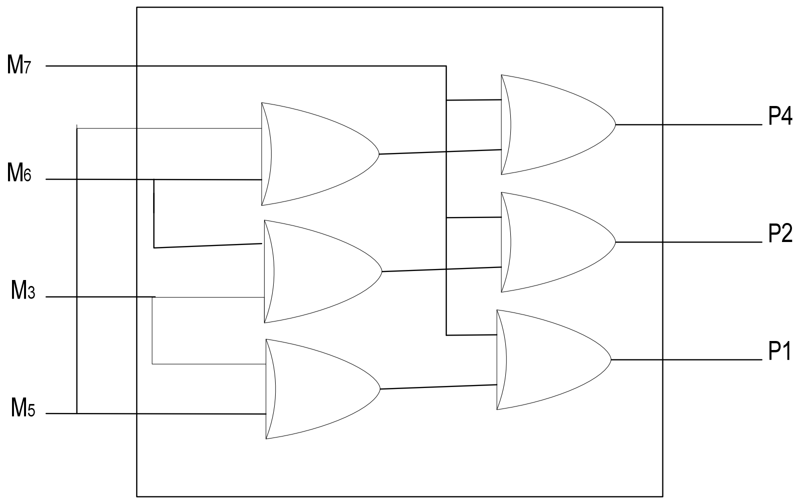

Figure 1. *Exclusive Or* Implementation of the H(7,4) Code

As discussed earlier, the implementation of an H(7,4) with dual input ***exclusive or*** gates will require **9** gates compared to only **6** in figure 1. That is a saving of **33%**. We can achieve a better design with only **1 *exclusive or*** gate and **5** inverters as shown in figure 2. The parity bits can be computed for two, three and four message bits. In the first stage we start with only two message bits $\mathbf{M_3}$ and $\mathbf{M_5}$ and show the required circuit. We then add $\mathbf{M_6}$ in the second stage followed by $\mathbf{M_7}$ in the third. The addition of $\mathbf{M_6}$ will enable the inverters to toggle $\mathbf{P_4}$ and $\mathbf{P_2}$ when $\mathbf{M_6}$ is equal to **1**. When $\mathbf{M_6}$ is equal to **0** no inversion takes place. Similarly, on the last stage, the addition of $\mathbf{M_7}$ would flip all three parity bits when $\mathbf{M_7}$ is equal to **1**. Otherwise, no inversion takes place. The figure assumes both $\mathbf{M_6}$ and $\mathbf{M_7}$ to be equal to 1.

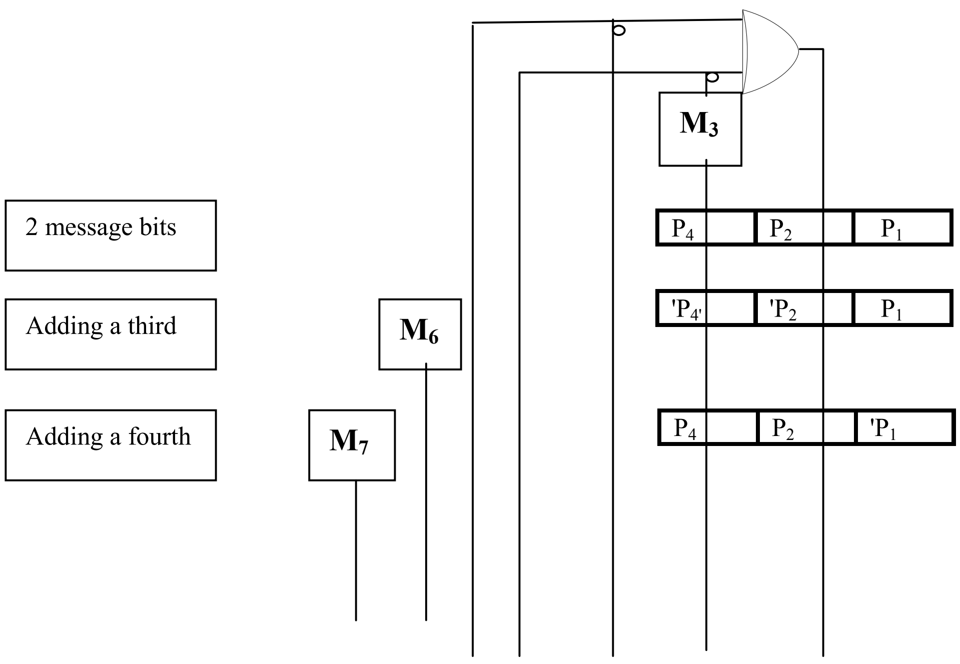

Figure 2. Incremental Implementation of the Hamming code

6. Conclusion

A modular implementing of a Hamming code has been introduced. The implementation is based on first building a seed circuit for a smaller size code and then incrementally adding message bits as necessary. Such approach provides flexibility to both hardware and software designers. Moreover, incremental addition of message bits provides savings in circuitry that is directly proportional to the size of the intended Hamming code. The paper demonstrated a reduction of 33% in circuitry when incremental approach is used for the H(7,4) case. Furthermore, in situations where a Hamming table needs to be store, the proposed optimization enables storing only the seed thereby saving storage space.

References

[1] Abuelyaman E. and Daniel S. "SPEC: Single-Packet Error Control Protocol for WATM Networks" Proceedings of the 2003 ICWN, pp 504-509, Las Vegas, USA, 2003

[2] G. Bernelli, et al " A Data Link Layer Protocol for Wireless ATM," IEEE Tran on Communications, pp 1438-1442, 1997

[3] I. F. Alkyildiz et al "A new ARQ Protocol for Wireless ATM Networks," IEEE Trans on Communications, pp 1109-1113, 1998

[4] H. Xie et al "Data Link Control Protocol for WATM Access Channels" Internet Document on E6950 Wireless on Mobile Networking, 1996

[5] G. A. Aderounmu et al " Performance Comparison of Data Link Control Protocols for wireless ATM Networks," International Journal of Computers and Applications 2002, vol 24, No 3, pp 144-152, September, 2002

[6] R. W. Hamming "Error Detection and Error Correction Codes" Bell Systems Tech. Journal, vol 29, pp 147-160, April, 1950

Eltayeb Salih Abuelyaman received a PhD degree in Computer Engineering from the University of Arizona in 1988. He served as faculty member at various universities in the US for 18 years before moving to Prince Sultan University in Saudi Arabia where he served as a Faculty Member, the Director of the Information Technology and Computing Services and currently serves as the Dean of the College of Computer and Information Sciences. His current research Interest is in the areas of Computer Networks, Information Security and Database.

Dr. Abdul-Aziz Sultan Alsehibani received a BS and MS degrees in Computer Engineering from King Saud University in 1989 and Syracuse University in 1993 respectively. He received a PhD degree in Computer Science from Syracuse University in 1999. He served as faculty member at King Saud University before joining Prince Sultan University where he is currently serving as the Dean of Admissions and Registration. He also served as interim Dean for the College of Computer and Information Sciences form 2006 to 2007. Dr. Alsehibani's research is in the areas of Multimedia and Object Allocations, Computer Architecture, and Networking Security.