تقرير مشروع مخبر بنيان الحاسوب وتنظيمه

الفئة: **السادسة**

الطالب: **أشرف العاصي**

التاريخ: **الأربعاء 29-12-2021**

المشروع: **Error-Correction Memory Path**

- مقدمة:

- إن نقل المعطيات من مكان إلى آخر غالباً يتعرض إلى نوعٍ من التشويش يشوه (يغير) المعلومة المراد نقلها. فكانت الحاجة ماسة لتصميم خوارزميات تكتشف هذه الأخطاء وتصححها تلقائياً دون الحاجة إلى إعادة إرسالها مرة أخرى، أولى هذه الخوارزميات هي ال Hamming Code.

- طريقة عمل الخوارزمية:

1. التشفير – Encode:

- تعتمد خوارزمية Hamming على ال Parity Bit، والتي تعتمد على بوابة ال XOR.

- لنفرض أننا نريد إرسال n بت (سلسلة أصفار وواحدات عددها n)، نرمز لها ب d_i. ينص ال Hamming Code على أنه يجب إضافة k بت لهذه السلسلة، نرمز لها ب p_i، ليصبح المجموع هو $n + k$. تضاف هذه البتّات إلى المواقع التي تقابل قوى العدد 2؛ أي:

Position	$1(1)$	$2(10)$	$3(11)$	...	$n+k$
Seq.	p_1	p_2	d_1	...	d_n

وتحسب هذه البتّات اعتماداً على **موقعها** في السلسلة. حيث البت الأول هو XOR لجميع البتّات التالية التي يبدأ ال LSB لموقعها في النظام الثنائي ب 1:

$$p_1 = XOR(3, 5, 7, \dots)$$

والبت الثاني هو XOR لجميع البتّات التي تبدأ خانتها التي تلي ال LSB ب 1:

$$p_2 = XOR(3, 6, 7, \dots)$$

وهكذا تحسب جميع البتّات.

- نوجد العلاقة بين k وn عن طريق المتراجحة:

$$2^k - 1 \geq n + k$$

يمكن أن ننشئ الجدول التالي:

$\boldsymbol{k}$	$\boldsymbol{n}$
3	$[2, 4]$
4	$[5, 11]$
5	$[12, 26]$

- يمكن إضافة بت في الموقع صفر، يحسب عن طريق تطبيق ال XOR على **كل** البتّات.

2. فك التشفير – Decode:

- عند استقبال سلسلة من الأصفار والواحدات، يتم حساب ال Check-Bits. تحسب كما يلي:

$$C_1 = XOR(1, 3, 5, 7, \dots)$$

$$C_2 = XOR(2, 3, 6, 7, \dots)$$

وهكذا. أي تحسب بنفس مبدأ حساب ال Parity-Bits، مضافاً إليها البت نفسه.

- يتم فحص قيمة البتّات C_i كالتالي:

$$if\ C_i = 0 \Rightarrow no\ error$$

$$if\ C_i \neq 0 \Rightarrow error\ at\ C_i$$

حيث تعطي قيمة ال C_i **الموقع** التي حصل فيه الخطأ. فيتم إتمامه.

- ننوه على أن ال Hamming Code، تكتشف وتصحح خطأ **وحيد**، حيث لا تعمل هذه الخوارزمية بالشكل الصحيح لأكثر من خطأ. ولكن نسبة البتّات المستخدمة (المضافة) إلى المعطيات صغيرة نسبياً. فيمكن حسابها كالتالي:

$$k = 4, n = 11 \Rightarrow x = 36\%$$

$$k = 5, n = 22 \Rightarrow x = 22\%$$

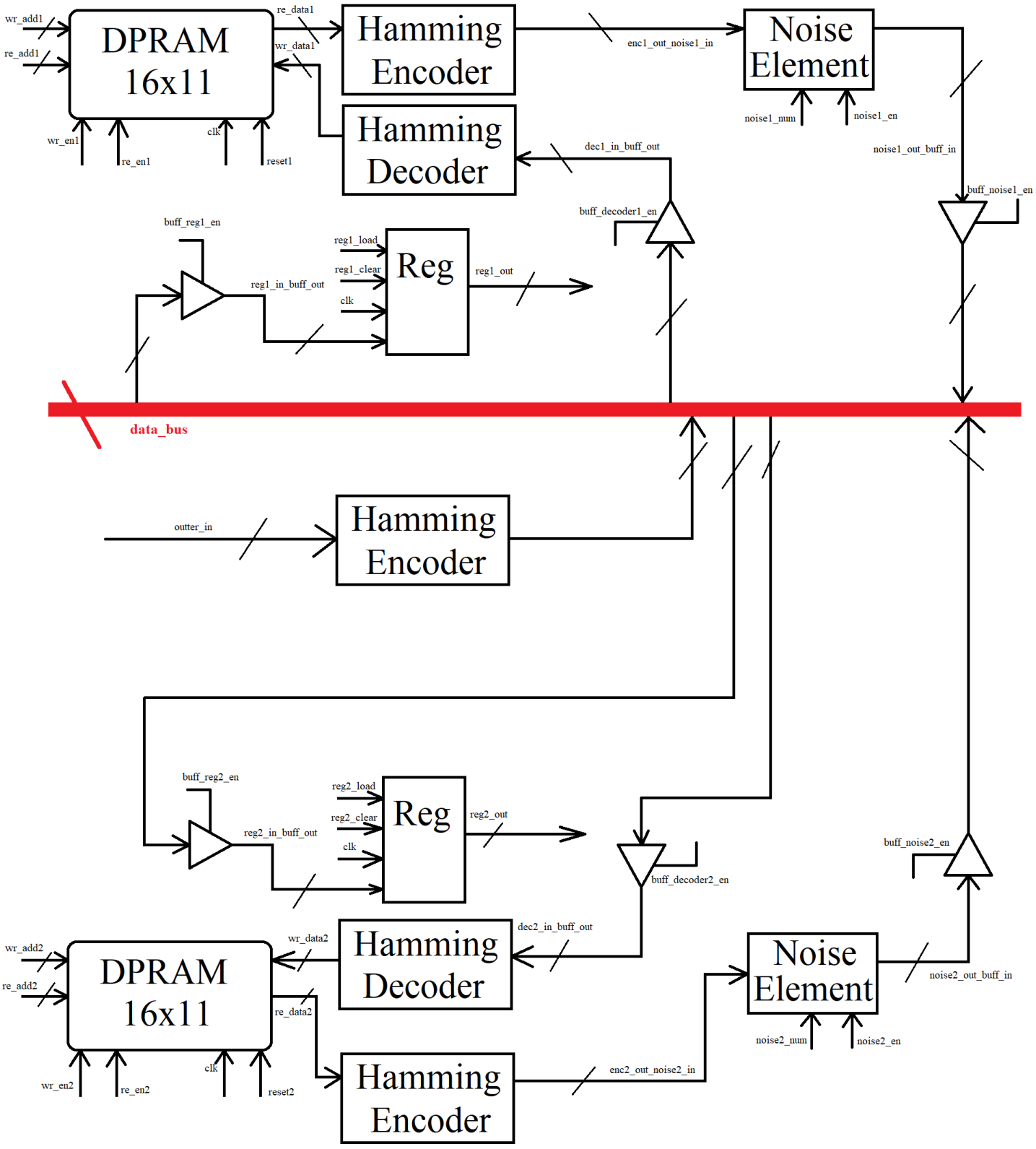
$$k = 21, n \approx 1000000 \Rightarrow x = 0.0021\%$$

- تم تصميم خوارزميات أخرى حديثة أكثر كفاءة تستخدم في مجالات عدة، كخوارزمية ال Cross-Interleaved Reed-Solomon Code، فهذه الخوارزمية مستخدمة في تصحيح الأخطاء في ال CDs، والتي تستخدم 25% من المعطيات كبتّات لاكتشاف وتصحيح الخطأ.

- مخطط الدارة:

- فكرة المشروع هي صنع ذاكرة رام، تنفذ خوارزمية Hamming قبل القراءة والكتابة منها وعليها. فتسمى هذه الأنواع من الذواكر بال Error-Correction Code RAM.

- المخطط:


wr_add1
re_add1
DPRAM 16x11
wr_en1
re_en1
clk
reset1
re_data1
wr_data1
Hamming Encoder
Hamming Decoder
enc1_out_noise1_in
Noise Element
noise1_num
noise1_en
noise1_out_buff_in
dec1_in_buff_out
buff_noise1_en
buff_decoder1_en
buff_reg1_en
reg1_load
reg1_clear
clk
Reg
reg1_out
reg1_in_buff_out
data_bus
outter_in
Hamming Encoder
buff_reg2_en
reg2_load
reg2_clear
clk
Reg
reg2_out
reg2_in_buff_out
buff_decoder2_en
buff_noise2_en
wr_add2
re_add2
DPRAM 16x11
wr_en2
re_en2
clk
reset2
wr_data2
re_data2
Hamming Decoder
dec2_in_buff_out
Noise Element
noise2_num
noise2_en
noise2_out_buff_in
Hamming Encoder
enc2_out_noise2_in

- حيث لدينا ذاكرتين Dual Port، موصولتين على ممر المعطيات، مع مرمز، وفاك ترميز Hamming، بالإضافة إلى عنصر الضجيج الذي، إذا كان مفعلاً، يتمم (يغير) بت وحيد في موقع ما من السلسلة. كما لدينا سجلين يحتفظان بالقيمة المرسلة المشوشة لمقارنتها مع القيمة المرسلة (أو المستقبلة بعد التصحيح). يمثل الدخل، هنا، مرسل خارجي عام، حيث يمكن أن يكون وسط تخزين ما، أو معلومات من كرت شبكة.

- تتألف محاكاة المخطط السابق من نقل سلسلة من الأصفار والواحدات من المرسل الخارجي إلى المستقبل الأول، ثم إعادة إرسالها إلى المستقبل الثاني عبر عنصر الضجيج، والتأكد من وصولها بالشكل الصحيح (استخدام الخوارزمية).

- GitHub:

- The whole *code*, along with the diagram, were uploaded to GitHub:

https://github.com/AlAssi69/ErrorCorrectionMemoryPathFinal