مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (5) العدد(4) 2021

Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (5) No. (4) 2021

تصميم وحدة حساب ومنطق(ALU) متسامحة العطل اعتمادا على جامع Carry Select Adder

د.م. سوزي صالح*

علا جوهره**

(تاريخ الإيداع 1 /3 /2021 . قُبِل للنشر في 13 /6 /2021)

□ ملخّص □

مع التطور الكبير في الأنظمة الإلكترونية الحديثة، توجب على المصممين مراعاة الأخطاء الممكن حدوثها ضمن هذه الدارات. وبما أن الجامع Carry Select Adder(CSeA) يعد من أسرع أنواع الجوامع فسيتناول هذا البحث تصميم وحدة حساب ومنطق لأربعة بت(4bit ALU) متسامحة العطل اعتمادا على دارة جامع (CSeA) متسامح العطل. يستطيع هذا التصميم إصلاح الأخطاء المفردة والمزدوجة دون مقاطعة عمل النظام حيث يقوم التصميم المقترح للجامع بعملية الفحص الذاتي لكشف الأخطاء وتصحيحها ليحافظ على خرج صحيح لوحدة الحساب والمنطق، وبالتالي امكانية استخدامها في التطبيقات و الأنظمة الحساسة بفعالية عالية. تمت عملية محاكاة الدارات الرقمية من خلال برنامج DSCH3.5 واستخدم برنامج Modelsim لحشر الأخطاء والتحقق من التصميم المقترح وكذلك تم تقييم أداء وحدة الحساب والمنطق المقترحة اعتمادا على تكنولوجيا CMOS 0.90nm باستخدام برنامج Microwind 31.

كلمات مفتاحية : جامع CSeA، متسامح العطل، خطأ مزدوج ، ALU، برنامج DSCH3.5 .

* أستاذ مساعد في كلية هندسة تكنولوجيا المعلومات و الاتصالات– قسم النظم الحاسوبية– جامعة طرطوس–طرطوس –سورية

**طالبة دراسات عليا (ماجستير)–كلية هندسة تكنولوجيا المعلومات والاتصالات –قسم تكنولوجيا الالكترونيات –جامعة طرطوس –طرطوس – سورية

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (5) العدد(4) 2021

Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (5) No. (4) 2021

Design A fault tolerant ALU based on Carry select adder (CSeA)

Dr. ENG. Susi Saleh [*]
Ola Jouhra [**]

(Received 1 / 3 / 2021 . Accepted 13 / 6 / 2021)

□ ABSTRACT □

In the advanced microelectronics, designers had to take into account possible errors within these circuits . And since the Carry Select Adder (CSeA) is one of the fastest adder. This paper focuses on the design 4BIT fault tolerant ALU based on fault tolerant Carry select adder (CSeA). This design can repair single and double faults without interrupting the system's work. The proposed design of the fault tolerant adder performs the process of self- checking to detect and correct faults to maintain a correct output of the ALU, and therefore the possibility of using it in applications and systems sensitive. Simulation of each technology was done through DSCH3.5,and we used Modelsim to fault injection and checking of the design, As well as the performance of the proposed arithmetic and logic unit was evaluated based on CMOS 0.90nm technology using Microwind 31.

Keywords: CSLA adder، double fault, fault tolerant, ALU, DSCH 3.5.

[*] Assistante Professor – Faculty of information and communication Technology, ICT Department -Tartous University- Tartous- syria

[***] Postgraduate student, ,Faculty of information and communication Technology, Electronical technology Department , Tartous University, Tartous ,Syria

1. مقدمة

مع ظهور الأنظمة الرقمية الكبيرة جداً VLSI (Very Large Scale Integration) و التي تمتاز بدرجة كثافة عالية (عدد كبير جداً من الترانزستورات ضمن رقاقة صغيرة)، فإن احتمال حدوث الأعطال الفيزيائية (faults) من النوع Stuck at 1 و Stuck at 0 (و التي تنتج عن دارات قصر ما بين أحد الاسلاك في الدارة و أحد منابع التغذية Vdd أو Gnd) تزداد بشكل كبير مسببة في أغلب الأحيان ظهور الأخطاء(errors) في بعض أجزاء النظام و التي بدورها قد تؤدي الى فشل عمل النظام.[2-1]

التسامح مع العطل هو قدرة النظام على أداء وظائفه حتي في حال وجود خطأ، ويرتبط هذا بمفهوم الوثوقية والتشغيل الناجح للنظام. التسامح مع العطل ضروري لأنه من المستحيل عمليا بناء نظام مثالي، لذلك تعد عملية تصحيح الأخطاء التي تحدث في الانظمة و التطبيقات الحديثة والتي من الصعب أن تتدخل اليد البشرية فيها أثناء عملها جزء أساسي و مهم عند تصميم أي نظام.

تعد وحدة الحساب والمنطق الدماغ لأي وحدة معالجة مركزية(CPU) مستخدمة في التطبيقات المختلفة مثل المعالجات الدقيقة ومعالجات الإشارات الرقمية. وقد اولى الباحثون اهتماما كبيرا لدارة الجامع لأنها تمثل الجزء الأساسي من الوحدة الحسابية حيث تنفذ عملية الجمع والعمليات الحسابية الأخرى وبالتالي فإن أي تحسين في دارة الجامع ينعكس تحسينا كبيرا في الـ ALU.[4-3]

يمكن استخدام تقنيات مختلفة خارجيًا أو داخليًا لتحسين الأداء العام لأي نظام . تتضمن التقنيات الخارجية التعامل مع خصائص بيانات الإدخال بينما تهتم التقنيات الداخلية بتصميم الدارة.[9-5]

مع التطور الكبير للأجهزة و الانظمة الإلكترونية، توجب على المصممين مراعاة الامور التالية عند تصميم أي دارة: عدد الترانزستورات التي تدخل في تصميم الدارة، ومساحة الرقاقة المتاحة، واستهلاك الطاقة والتأخير الزمني، ومن هنا ظهرت الحاجة إلى تصميم دارات بطاقة منخفضة وسرعة عالية وأداء عالٍ ومساحة رقاقة صغيرة [11-10] .

بناءً على ذلك، تم اقتراح العديد من دارات الجامع (CSeA) المتسامحة العطل و التي تراعي شروط التصميم VLSI مثل استهلاك الطاقة والتأخير والمساحة.

قدم الباحثون في [12] نموذجاً لتصميم جامع (CSeA)متسامح العطل قادر على تقليل عدد الترانزستورات بنسبه تتراوح بين 19.15%_ 20.94% وكذلك تقليل المساحة بنسبه تتراوح بين 16.07%_ 20.67%بالمقارنة مع الجامع التقليدي الغير قادر على التسامح مع الخطأ .

هذا التصميم قادر على كشف الاخطاء المفردة فقط ولكنه غير قادر على تحديد موقع الخطأ وكذلك يعاني من مشكلة انتشار الخطأ عبر الحمل.

قدم الباحثون في [13] نموذجا لتصميم جامع (CSeA)متسامح العطل قادر على تحديد موقع الخطأ بالتالي يمكننا استبدال الوحدة المعيبة فقط ، على الرغم من أنه يحتاج مساحة أقل من التصميم السابق و يحتاج تكلفة مادية أقل لكنه ايضا غير قادر على كشف الأخطاء المزدوجة.

قام الباحثون في [14] بتصميم نموذج لجامع (CSeA)متسامح العطل قادر على اكتشاف الاخطاء واصلاحها حيث اعتمد الباحثون على مفهوم التصحيح المشترك للأخطاء لكل من الحمل والجمع وبالتالي هذا التصميم غير قادر على اكتشاف الاخطاء المزدوجة .

2. هدف البحث

سنقوم في هذا البحث بتصميم دارة حساب ومنطق(ALU) متسامحة العطل اعتمادا على دارة جامع (CSeA) متسامحة العطل وقادرة على الاصلاح الذاتي باستخدام تقنية CMOS، حيث يكون هذا التصميم قادر على اكتشاف واختبار الأخطاء المفردة والمزدوجة مع إمكانية تصحيح هذه الأخطاء لإعطاء خرج صحيح ، تم الاعتماد على برنامج DSCH3.5 من أجل محاكاة الدارة، وتم اختبار الدارة وحشر الاخطاء فيها من النوع Stuck at 1 و Stuck at 0 باستخدام برنامج Modelsim وكذلك تم تقييم أداء وحدة الحساب والمنطق المقترحة اعتمادا على تكنولوجيا CMOS 0.90nm باستخدام برنامج Microwind 3.1.

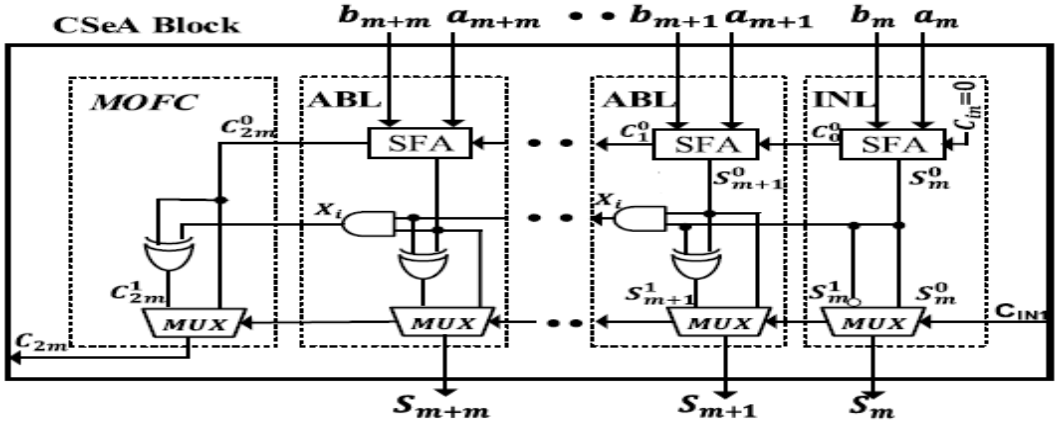
3. طرائق البحث ومواده

3-1 البنية التقليدية لجامع CSeA:

يتكون جامع CSeA من كتلتين أساسيتين هما الكتلة الاولية Initial Block (INL) وكتلة الجامع Adder Block (ABL) ، كما هو موضح في الشكل(1) والسبب لوجود كتلتين لإنشاء CSeA يعود إلى المبدأ الأساسي المستخدم لإنشاء CSeA معتمدا على جامع Ripple carry adder (RCA) فردي والذي ينص على :

باستثناء البتات الأقل اهمية والتي ستكون دائما متممة لبعضها، فإن بتات المجموع المحسوبة للقيمة المتممة لـ Cin الابتدائية ستكون دائما متممة لبعضها اذا كانت جميع البتات الأقل اهمية تساوي 1 .

تستخدم الكتلة الاولية INL لتوليد بتات المجموع الأقل اهمية عندما Cin=0، أما بتات المجموع الباقية يتم توليدها باستخدام كتلة الجامع (ABL) حيث تستخدم بوابة ال AND لتمرير حالة بتات المجموع للحالة السابقة المحسوبة عندما Cin=0، وأما بوابة ال XOR تستخدم لتوليد بتات المجموع عندما Cin=1 من خلال النظر في حالة بتات المجموع السابقة. المجموع والحمل الجزئي المتولد يشار اليه ب S_i^j و C_i^j على التوالي حيث الـ j تشير الى قيمة ال Cin و الـ i تشير الى رتبة البت، أما الـ Cout النهائي فيتم توليده باستخدام الوحدة (MOFC).



الشكل (1): البنية التقليدية لجامع CSeA

3-2. النموذج المقترح وقدرته على كشف الأخطاء:

قام الباحثون في [13] باختبار جزئي لجامع ال CSeA من خلال البنية المبينة في الشكل (2)،حيث تم اختبار وحدة الجامع الكامل (FA) فقط، إن عملية اختبار الخطأ تمت باستخدام دارة FA ودارة لاكتشاف التكافؤ وبوابتين XNOR حيث الغرض من بوابة XNOR الأولى (G1) هو التحقق مما إذا كانت بتات Sum و Cout متساوية أو متممه لبعضهما، مع العلم أن الـ Sum والـ Cout سيكونان دائمًا متممين لبعضهما البعض إلا عندما تكون جميع المدخلات متساوية. و بوابة XNOR الثانية (G2) سوف تتحقق من ناتج G1 من خلال مقارنته مع اختبار التكافؤ(Eq) وبالتالي إنشاء مؤشر الخطأ النهائي(Ef)حسب الحالات التالية.

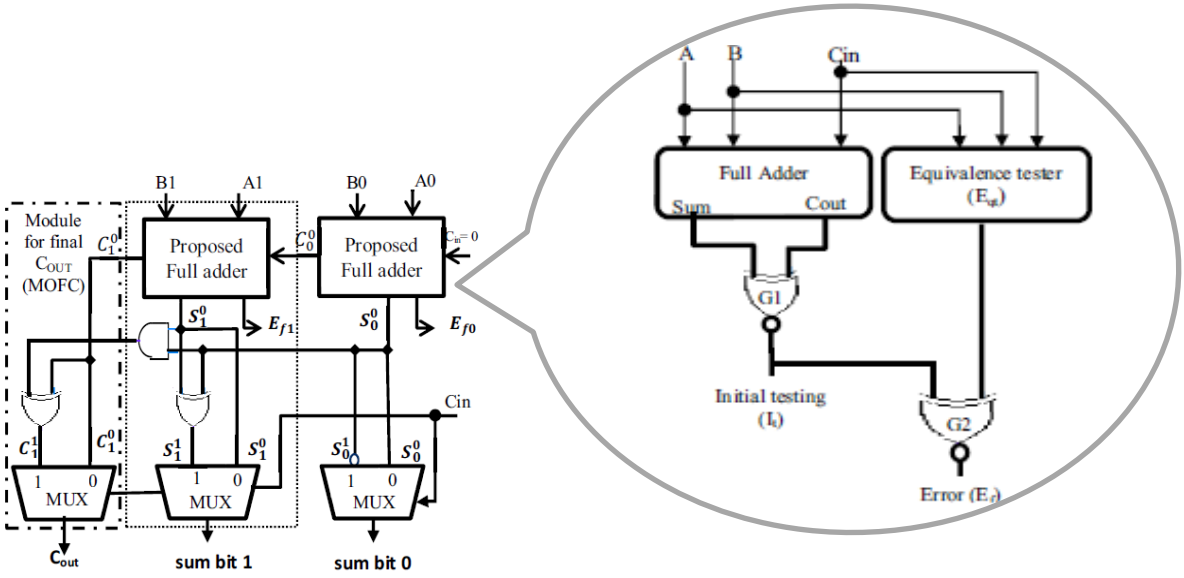
If (Eq == 0) AND (Sum == Cout))=> No fault

If (Eq == 1) AND (Sum ==$\overline{\text{Cout}}$))=> No fault

If (Eq== 0) AND (Sum ≠ Cout)) => fault

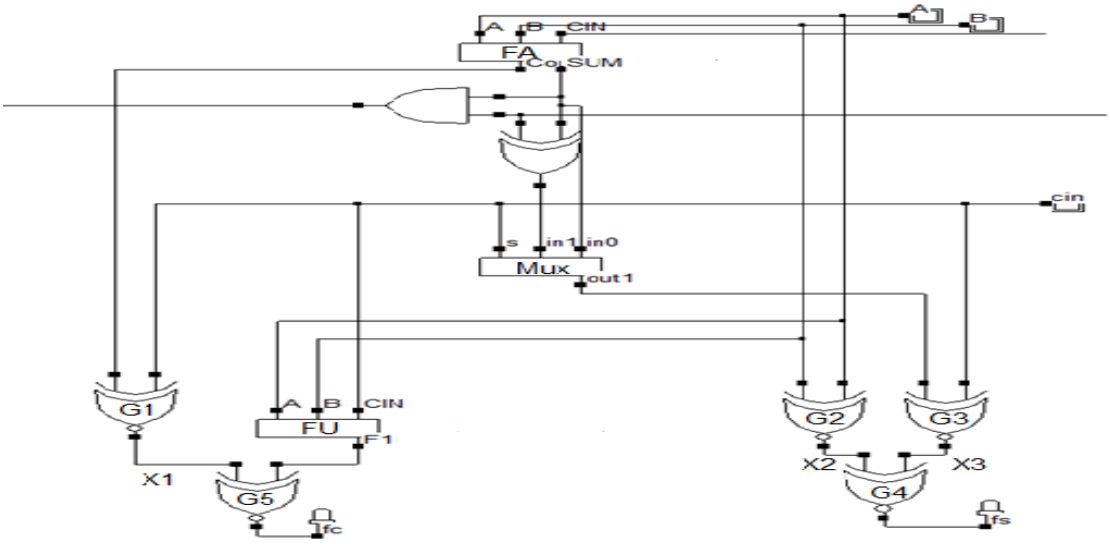
If (Eq == 1) AND (Sum ≠ $\overline{\text{Cout}}$))=> fault

يقوم هذا التصميم باكتشاف الأخطاء المفردة فقط، ففي حال حدوث خطأ مزدوج على كل من الـ Sum والـ Cout سيفشل النظام في اكتشاف الخطأ. وكذلك في حال حدوث خطأ خارج وحدة ال FA سيفشل النظام باكتشافه



الشكل(2): جامع CSLA يكشف الأخطاء المفردة فقط

ولتفادي المشاكل التي واجهت الباحثين في [13] قمنا بتصميم نموذج جديد لجامع CSeA قادر على اكتشاف الاخطاء المفردة والمزدوجة اينما حدثت ضمن وحدة CSeA يوضح الشكل (3) بنية التصميم المقترح .



شكل (3): النموذج المقترح لجامع CSLA يكشف الأخطاء المفردة والمزدوجة .

اعتمدنا في هذا التصميم على وحدة وظيفية (FU) وخمس بوابات XNOR (G1,G2,G3,G4,G5) بغية اختبار الخطأ.

حيث لاكتشاف الخطأ الممكن حدوثه على ناتج قيمة الجمع (SUM) سنعتمد على مبدأ انه في حال كانت قيم الدخل (A,B) متساوية فإن المجموع (SUM) سيساوي الـ Cin، وأما عندما تكون قيم الدخل متممة لبعضها فإن المجموع (SUM)سيكون متمم للـ Cin، وسنستخدم لتحقيق ذلك ثلاث بوابات XNOR (G2,G3,G4) ، أما بوابة G2 ستستخدم لمقارنة قيم الدخل(A,B)، بينما G3 ستستخدم لمقارنة الـ Sumو الـCin، في حال عدم وجود خطأ سيكون خرج البوابتين متماثل. أما عند حدوث خطأ سيصبح الخرجين متممين لبعضهما البعض لذلك سنستخدم بوابة G4 لتقوم بمقارنة خرج بوابة G2 مع خرج بوابة G3 يتم التعبير عن خرج بوابة G2 ،G3 ،G4 بالرموز X2,X3,Fsعلى التوالي ويمكن تمثيلها بالعلاقات التالية :

$X2=(\overline{A \oplus B})$ (1)

$X3=(\overline{sum \oplus Cin})$ (2)

$Fs=(\overline{X2 \oplus X3})$ (3)

إذا كانت قيمة X2 مع X3 متطابقتين تكون $Fs = 1$ وتعبر عن عدم وجود أخطاء والعكس إذا كانت $Fs = 0$ تعبر عن وجود خطأ في ناتج قيمة الجمع.

من أجل اكتشاف الخطأ الممكن حدوثه على ناتج قيمة الحمل (Cout) سنعتمد على فرضية أن قيمة الـ Cin تساوي قيمة الـ Cout في كل الحالات ماعدا الحالتين التاليتين:

$Cin = 0$ ،$B = 1$،$A = 1$ أو $Cin = 1$ ،$B = 0$،$A = 0$

وسنستخدم لتحقيق ذلك وحدة وظيفية (FU) وبوابتين XNOR (G1,G5) ،الوحدة الوظيفية سيتم تصميمها بحيث تعطي على خرجها 1 عندما $A = B \neq Cin$ أي في الحالتين التاليتين:

$Cin = 0$ ،$B = 1$،$A = 1$ أو $Cin = 1$ ،$B = 0$،$A = 0$

حيث تستخدم بوابة G1 لمقارنة قيمة ال Cin مع قيمة ال Cout و أما بوابة G5 تستخدم لمقارنة خرج بوابة G1 مع خرج الوحدة الوظيفية (FU) ،يتم التعبير عن خرج بوابة G1 و الوحدة الوظيفية وبوابة G5 بالرموز X1,F1,Fc على التوالي ويمكن تمثيلها بالعلاقات التالية :

X1=$(\overline{Cout \oplus Cin})$ (4)

F1=($\overline{A'B'C + ABC'}$) (5)

Fc=$(\overline{X1 \oplus F1})$ (6)

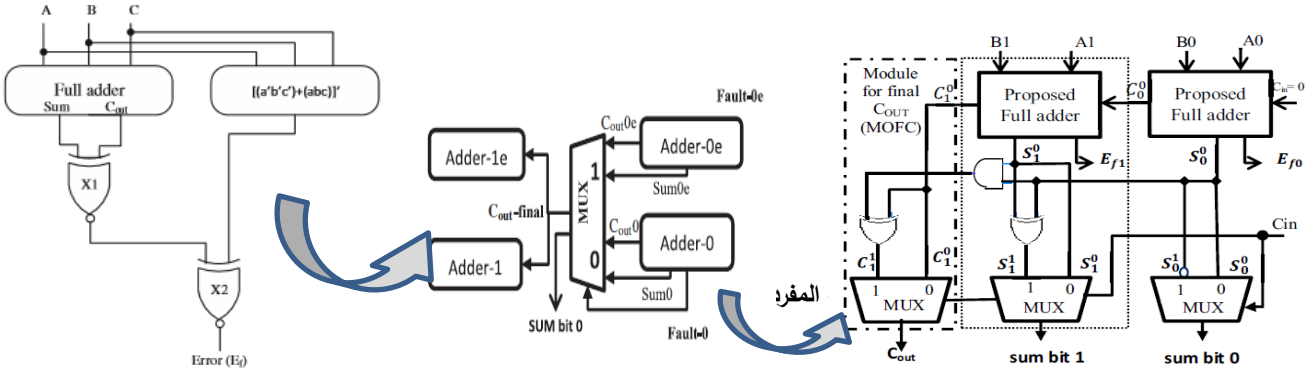
إذا كانت قيمة X1 مع F1 متطابقتين تكون $Fc = 1$ وتعبر عن عدم وجود أخطاء والعكس إذا كانت $Fc = 0$ تعبر عن وجود خطأ في قيمة الحمل، يوضح جدول الحقيقة (2) طريقة كشف الأخطاء في الدارة.

جدول (2): جدول الحقيقة للتصميم المقترح

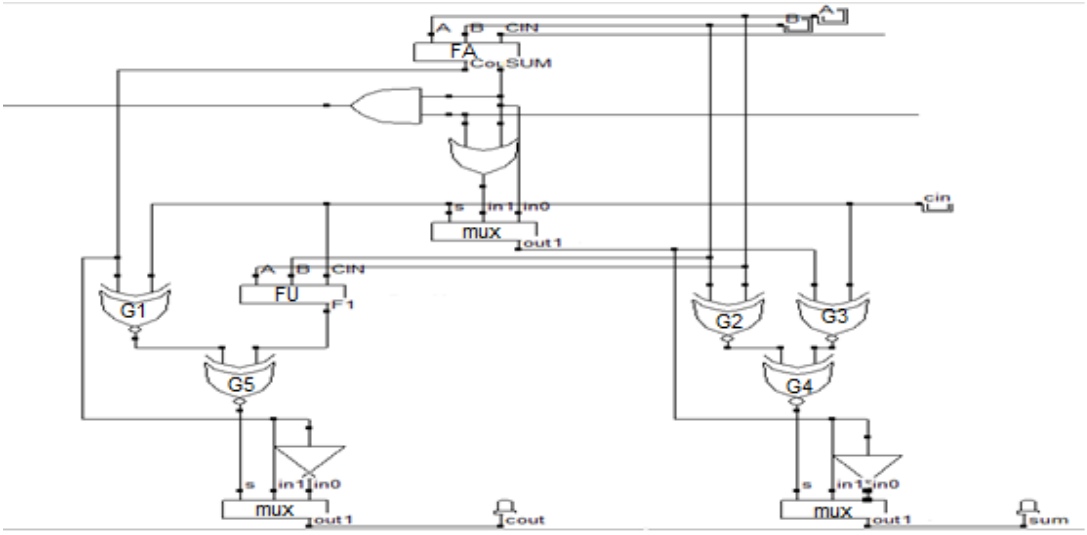
A	B	Cin	Sum	Cout	X2	X3	Fs	X1	F1	Fc
0	0	0	0	0	1	1	1	1	1	1
0	0	1	1	0	1	1	1	0	0	1
0	1	0	1	0	0	0	1	1	1	1
0	1	1	0	1	0	0	1	1	1	1
1	0	0	1	0	0	0	1	1	1	1
1	0	1	0	1	0	0	1	1	1	1
1	1	0	0	1	1	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1

3-3. النموذج المقترح و قدرته على إصلاح الأخطاء:

قام الباحثون في [14] بتطوير الدراسة المقترحة في [13] بهدف تصحيح الخطأ بدلا من استبدال وحدة ال FA التي حدث فيها الخطأ ،حيث استخدموا جامع كامل مثالي بالإضافة للجامع الكامل الأساسي واعتمادا على اشارة الخطأ Efسيتم اختيار الخرج (عند وجود خطأ سيتم اختيار خرج الجامع المثالي وعند عدم وجود خطأ سيتم اختيار خرج الجامع الأساسي) كما يظهر في الشكل (4)



لكن كما ذكرنا سابقا فهذا التصميم سيفشل في اكتشاف الخطأ عند حدوث خطأ مزدوج على كل من الحمل والناتج وبالتالي سيفشل في تصحيحه، وعلى الرغم من قدرته المحدودة في تصحيح الأخطاء، فهذا التصميم يحتاج الى عدد كبير من المكونات المادية مما يتطلب مساحة كبيرة وكلفة عالية، لذلك تم تحسين هذا النموذج ليصبح كما في الشكل (5).



شكل (5): النموذج المقترح لجامع CSeA يصحح الأخطاء المفردة والمزدوجة.

يعتمد التصميم المقترح على اشارات التحكم Fs,Fc التي يتم الحصول عليها من التصميم المقترح للجامع القادر على الاختبار الذاتي.

حيث نلاحظ أن بتات المجموع والحمل ستكون إما 0 أو 1 اعتمادا على دخل الجامع .

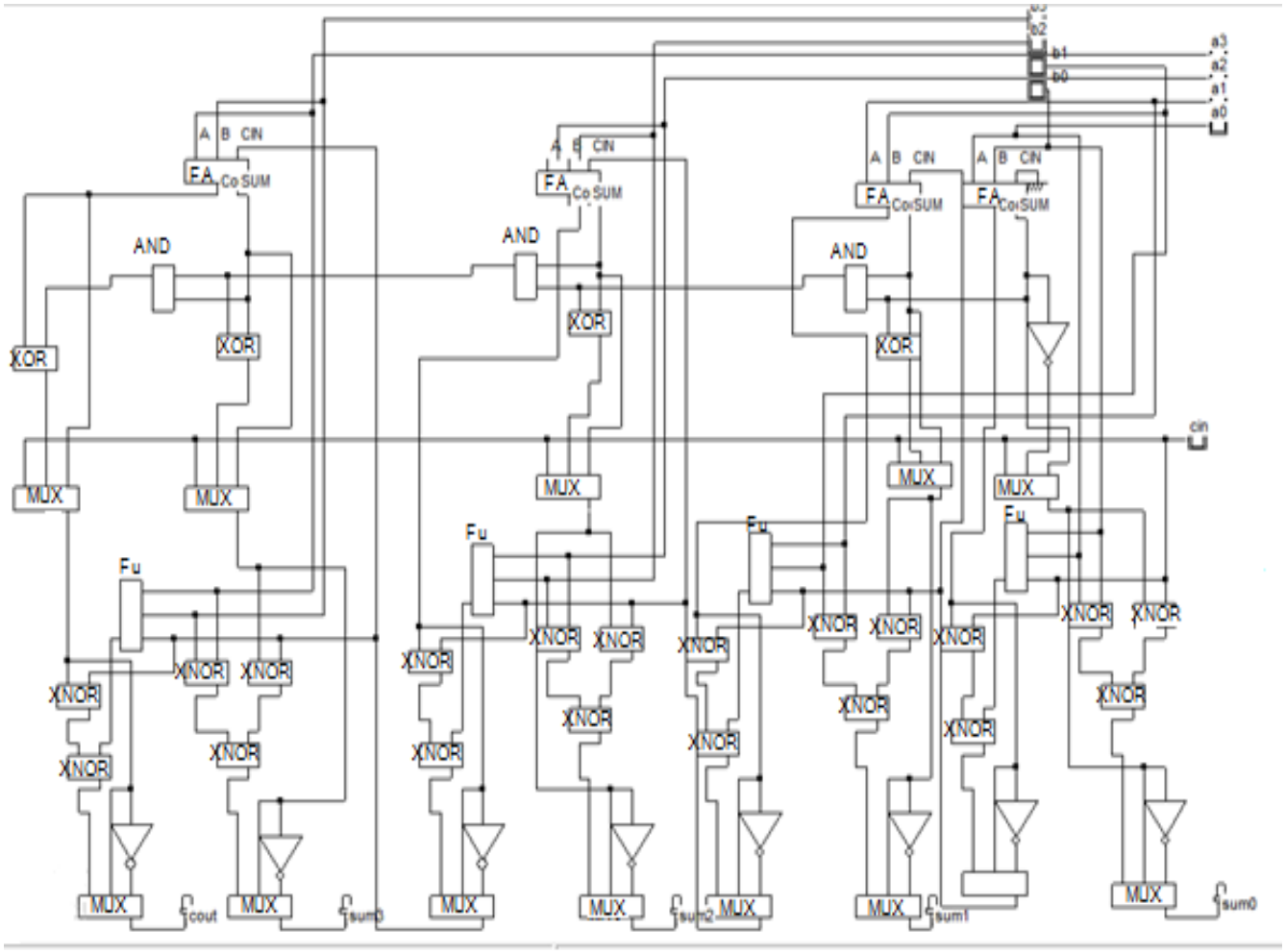
- اذا كانت اشارة Fs تشير الى وجود خطأ في بتات المجموع وبالتالي سيتم اختيار القيمة المعاكسة لبتات المجموع من خلال ناخب يتم التحكم به بواسطة اشارة التحكم Fs
- وكذلك بالنسبة لبتات الحمل اذا كانت اشارة Fc تشير الى وجود خطأ في بتات الحمل وبالتالي سيتم اختيار القيمة المعاكسة لبتات الحمل من خلال ناخب يتم التحكم به بواسطة اشارة التحكم Fc.

سنناقش حالة الجامع كما يلي :

- عندما Fs=1&Fc=1 اذا الجامع خالي من الاخطاء وسيتم اختيار الخرج لبتات الناتج والحمل من الجامع مباشرة بواسطة الناخب.
- عندما Fs=0&Fc=1 اذا يوجد خطأ في بتات المجموع لإصلاح هذا الخطأ سيقوم الناخب باختيار عكس بتات المجموع أما بتات الحمل سيتم اختيارها بشكل مباشر من الجامع.
- عندما Fs=1&Fc=0 اذا يوجد خطأ في بتات الحمل لإصلاح هذا الخطأ سيقوم الناخب باختيار عكس بتات الحمل أما بتات المجموع سيتم اختيارها بشكل مباشر من الجامع.
- عندما Fs=0&Fc=0 اذا يوجد خطأ في بتات الحمل والمجموع لإصلاح هذا الخطأ سيقوم الناخب باختيار عكس بتات الحمل والمجموع .

يضمن النموذج المقترح اصلاح جميع الأعطال المؤقتة والدائمة بنسبة 100% ويجعل الجامع خالي من الأخطاء، وكذلك يستهلك مساحة أقل بالمقارنة مع التصميم المقترح في [14].

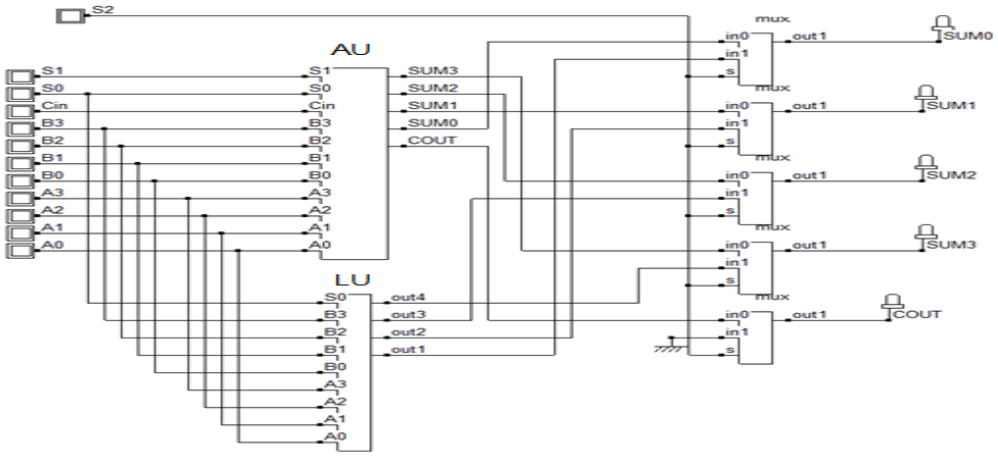
بالإضافة لذلك يتميز هذا التصميم ببساطته، وامكانية توسيعه بسهولة حيث يمثل الشكل(6)جامع CSeA (4 بت) قادر على اكتشاف الأخطاء المفردة والمزدوجة واصلاحها، وسنستخدم هذا الجامع كبنية أساسية لتصميم ال ALU .



شكل (6): النموذج المقترح لدارة جامع CSLA 4بت متسامح العطل

3-4. تصميم وحدة الحساب والمنطق (ARITHMETIC AND LOGICAL UNIT):

تعد وحدة الحساب والمنطق(ALU) المكون الأساسي لوحدة المعالجة المركزية (CPU) حيث تتألف من وحدة الحساب (AU) ووحدة المنطق (LU) ونواخب 1*2 كما هو موضح بالشكل (7) .



شكل (7): النموذج المقترح لوحدة الحساب والمنطق ALU المؤلفة من AU وLU ونواخب.

3-4-1. تصميم وحدة الحساب(Arithmetic Unit) :

يوضح الشكل(8) تصميم وحدة الحساب باستخدام جامع CSeA (4 بت) متسامح مع الخطأ ونواخب 4*1. حيث يتم اختيار إحدى العمليات التالية (الجمع، الطرح، الزيادة، النقصان، وتمرير الدخل الى الخرج) اعتمادا على قيم الدخل(S0,S1,Cin) كما يظهر بالجدول(2). وتنفذ العمليات الحسابية كما يلي:

A. الجمع :

يتم تطبيق الدخل B كدخل الى الجامع عند S0=0,S1=0، حيث عندما Cin =0سيكون الناتج $A+B$ ، أما عندما Cin=1 سيكون الناتج $A+B+1$ (سيتم تنفيذ عملية الجمع في كلتا الحالتين مع أو بدون وجود حمل)

B. الطرح :

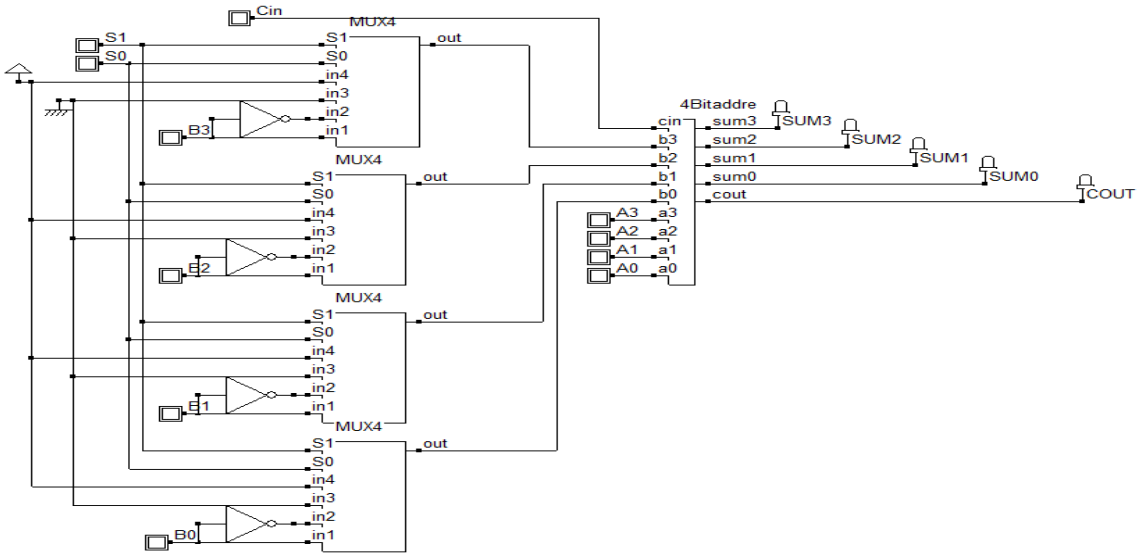
ستتم عملية الطرح باستخدام المتمم الثنائي لذلك سنمرر الى الجامع متمم B باستخدام بوابة العاكس عند S0=1,S1=0، بالتالي سيكون الخرج $A+\bar{B}+1$ عندما Cin=1، أما عندما Cin =0 سيكون الخرج $A+\bar{B}$ والذي يكافئ الاستعارة مع حمل أي $A-B-1$.

C. الزيادة والتمرير

عندما S0=0,S1=1 يتم اهمال الدخل B ويتم تمرير قيمة ال 0 الى دخل الجامع، فيصبح الخرج A + 0 + Cin وهذا يمثل A عندما Cin =0، وأما عندما Cin=1 سيكون الخرج $A+1$

D. النقصان والتمرير

عند S0=1, S1=1 يتم تمرير قيمة 1 الى دخل الجامع، فيكون الخرج $A-1$ عندما Cin =0 لان 1 هو متمم ثنائي للعدد 1 وعند جمع العدد A مع المتمم الثنائي للعدد 1 سيعطي $A-1$ ، وأما عندما Cin=1 سيصبح الخرج $A-1+1$ مما يؤدي الى تمرير قيمة A الى الخرج .



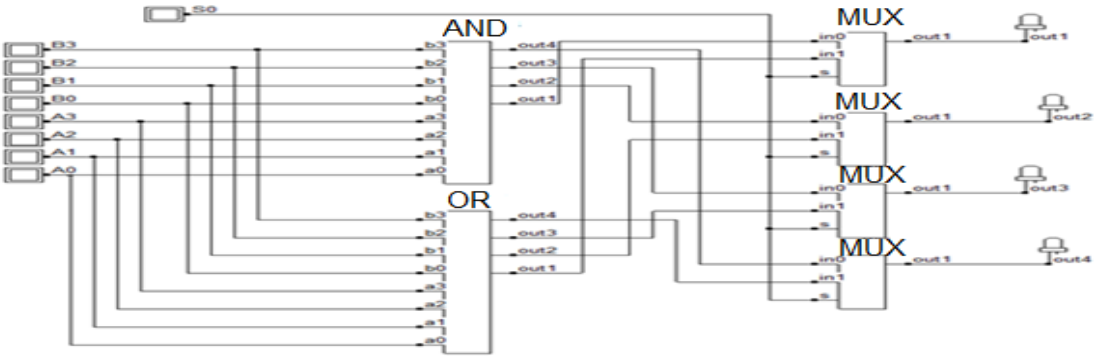
شكل (8): النموذج المقترح لوحدة الحساب AU

الجدول(2):يوضح العمليات التي تقوم بها وحدة الحساب

S1	S0	Cin	Out	Function
0	0	0	$A+B$	Add
0	0	1	$A+B+1$	Add with carry
0	1	0	$A+\bar{B}$	Subtract with Borrow
0	1	1	$A+\bar{B}+1$	Subtract
1	0	0	A	Transfer A
1	0	1	$A+1$	Increment A
1	1	0	$A-1$	Decrement A
1	1	1	A	Transfer A

3-5-2. تصميم وحدة المنطق(Logical Unit) :

سيتم تصميم وحدة المنطق باستخدام دارة AND (4 بت) متسامحة مع الخطأ ودارة OR (4 بت) متسامحة مع الخطأ ونواخب 1*2. حيث تنفذ وحدة المنطق المبينة بالشكل (9) اعتمادا على قيمة S0 إحدى العمليتين التاليتين (AND,OR). يبين الجدول (3) العمليات التي تقوم بها وحدة المنطق، بينما يوضح الجدول (4) العمليات التي تقوم بها الـ ALU.



شكل (9): النموذج المقترح لوحدة المنطق LU

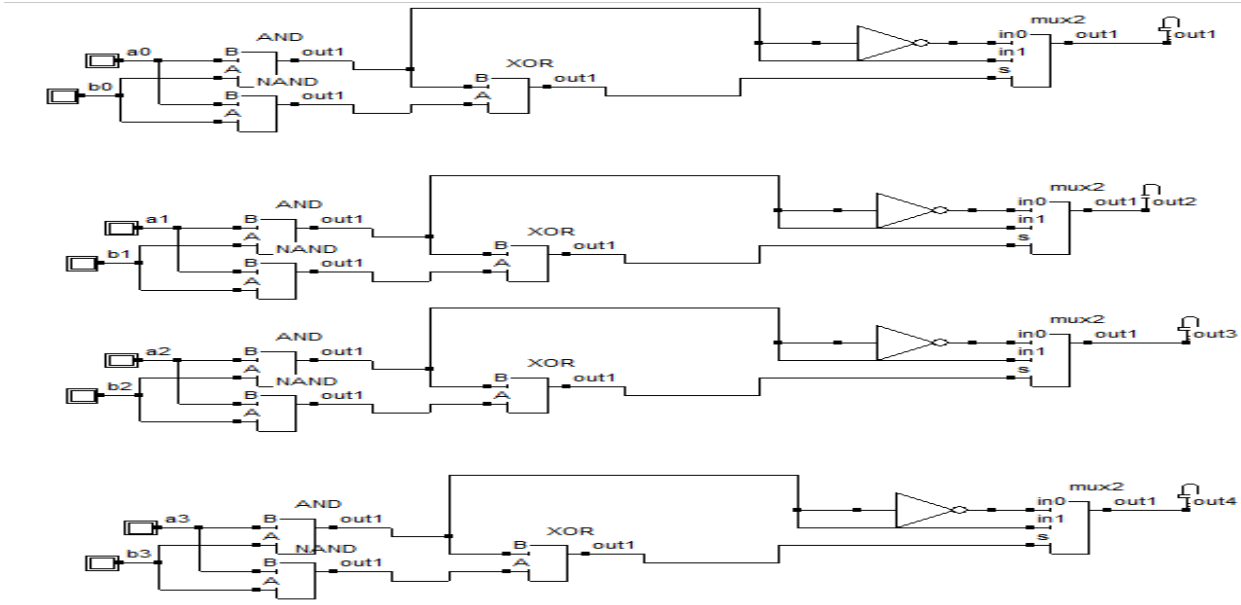
الجدول(3):يوضح العمليات التي تقوم بها وحدة المنطق

S0	OUT	Function
0	A AND B	AND
1	A OR B	OR

A. تصميم دارة AND متسامحة مع الخطأ.

يوضح الشكل (10) تصميم لبوابة AND (4 بت) قادرة على اكتشاف الخطأ واصلاحه، حيث سيتم اختبار بوابة AND باستخدام بوابة NAND وبوابة XOR كما يلي:

- في حال عدم وجود خطأ سيكون خرج بوابة AND يعاكس بالقيمة خرج بوابة NAND وبالتالي خرج بوابة XOR يساوي الواحد.
- عند حدوث خطأ ضمن بوابة AND سيكون خرجها مماثل لخرج بوابة NAND وسيكون خرج بوابة XOR يساوي صفر دلالة على حدوث خطأ .

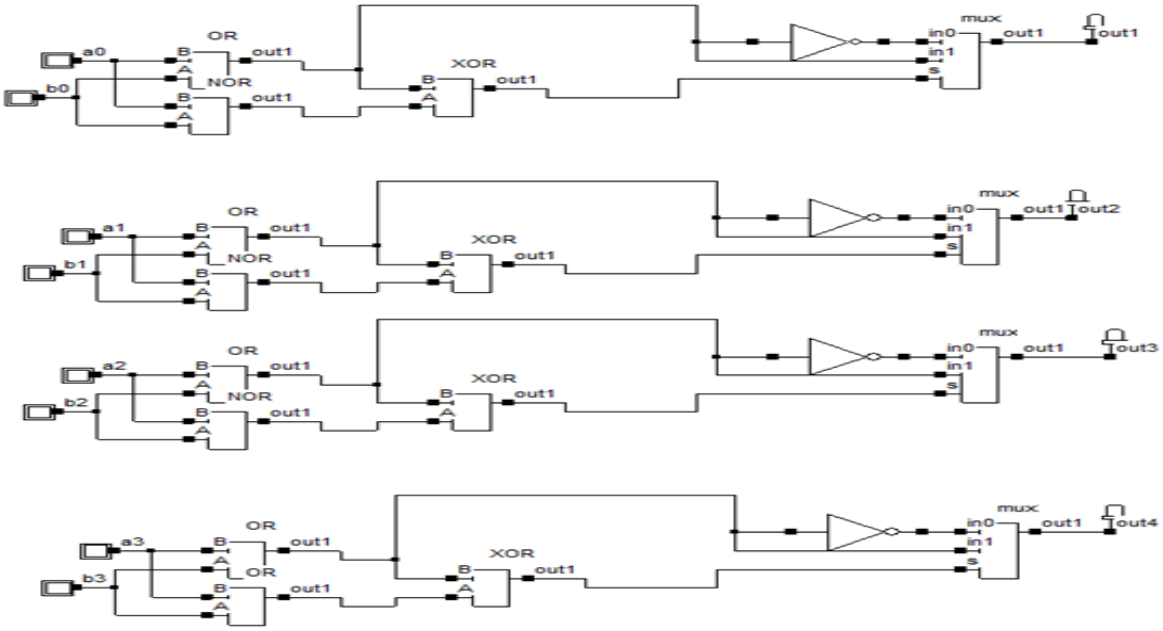


شكل (10): النموذج المقترح لدارة ال AND

لإصلاح الخطأ في حال حدوثه سنستخدم دارة ناخب (MUX) حيث سيمرر الناخب خرج بوابة AND عندما خرج بوابة XOR يساوي الواحد، وسيمرر عكس خرج بوابة AND عندما خرج بوابة XOR يساوي الصفر .

B. تصميم دارة OR متسامحة مع الخطأ.

يوضح الشكل (11) تصميم لبوابة OR (4 بت) قادرة على اكتشاف الخطأ واصلاحه ، حيث سيتم اختبار بوابة OR باستخدام بوابة NOR وبوابة XOR بنفس الآلية المتبعة لاختبار بوابة And.



شكل (11): النموذج المقترح لدارة ال OR

الجدول(4):يوضح العمليات التي تقوم بها وحدة الحساب والمنطق ALU

S2	S1	S0	Cin	Out	Function
0	0	0	0	$A+B$	Add
0	0	0	1	$A+B+1$	Add with carry
0	0	1	0	$A+\bar{B}$	Subtract with Borrow
0	0	1	1	$A+\bar{B}+1$	Subtract
0	1	0	0	A	Transfer A
0	1	0	1	$A+1$	Increment A
0	1	1	0	$A-1$	Decrement A
0	1	1	1	A	Transfer A
1	X	0	X	A AND B	AND
1	X	1	X	A OR B	OR

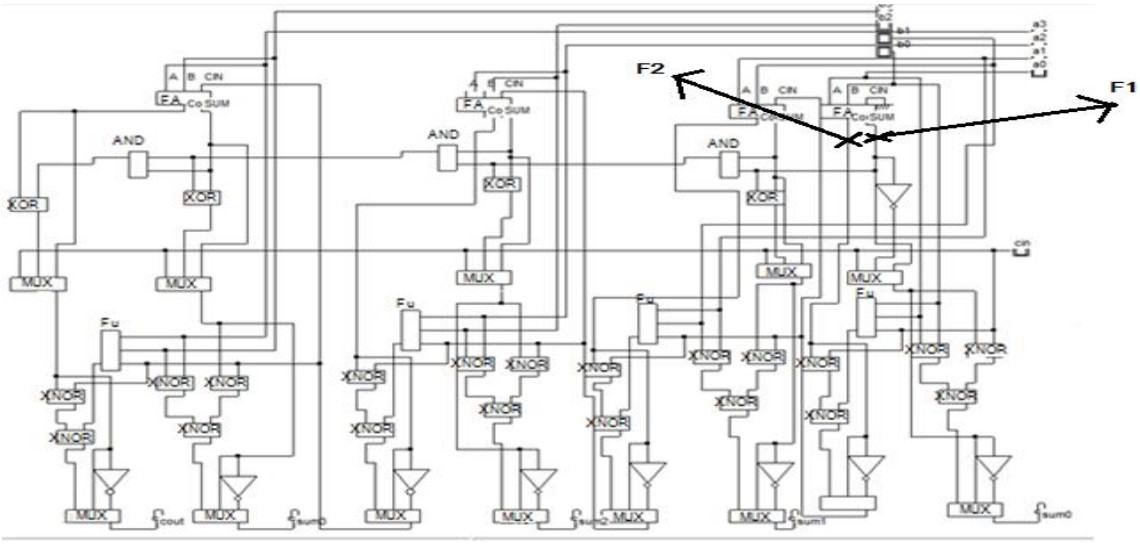
4 النتائج والمحاكاة

4-1.مرحلة محاكاة واختبار التصميم :

بعد أن استعرضنا بنية دارة جامع CSeA متسامح العطل و التي تعتبر البنية الاساسية لتصميم وحدة الحساب سيتم الاعتماد على برنامج Modelsim من أجل عملية المحاكاة و حشر مجموعة من الأخطاء المفردة والمزدوجة.

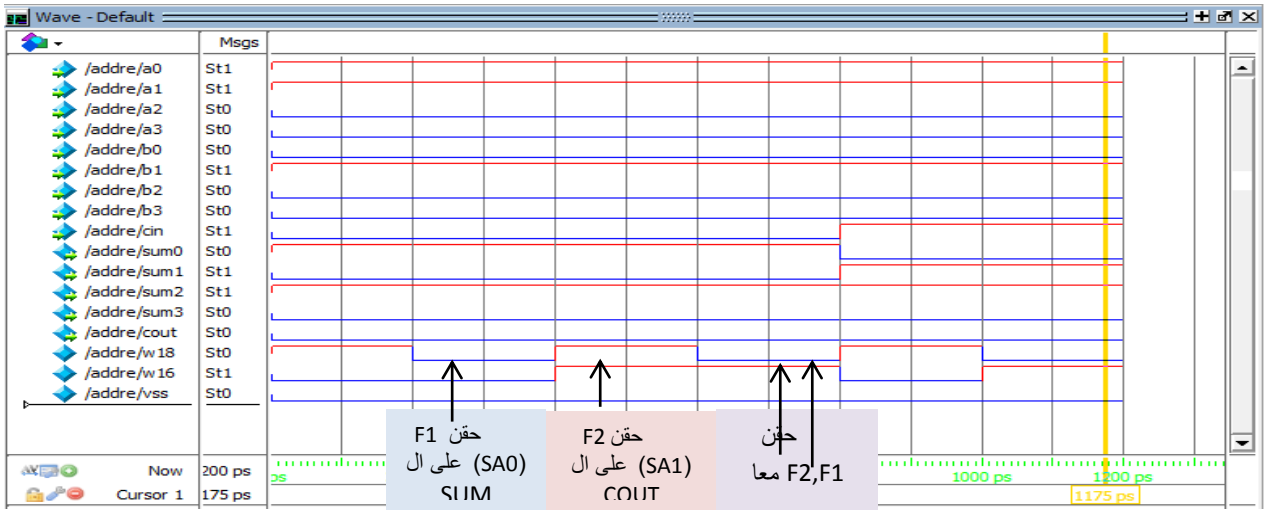
يبين الشكل (12) النموذج المقترح لدارة جامع CSeA متسامح العطل وفق تقنية CMOS مع حشر خطأين:

- الخطأ F1 على إشارة ناتج الجمع sum للجامع
- الخطأ F2 على إشارة الحمل Cout للجامع



شكل (12): حشر خطأين على ناتج الحمل والجمع

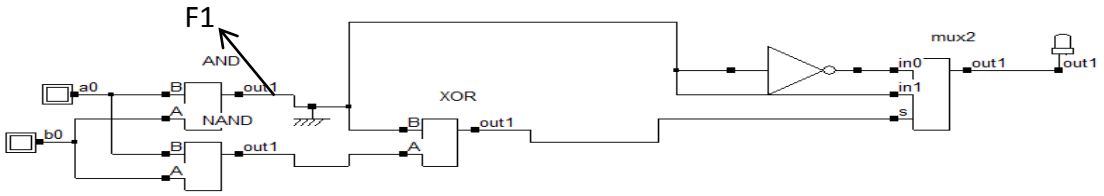
من أجل المحاكاة قمنا أولاً بمحاكاة الدارة السابقة مع وجود الخطأ F1 فقط (خطأ وحيد على ناتج الجمع) ثم محاكاة الدارة مع وجود الخطأ F2 فقط (خطأ وحيد على ناتج الحمل) ثم محاكاة الدارة مع وجود F1 و F2 (خطأ مزدوج) كما في الشكل (13)



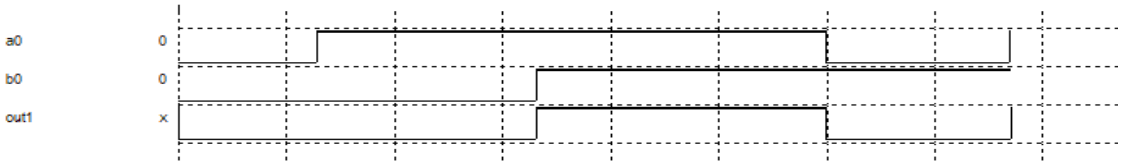
شكل (13): نتائج المحاكاة بحشر خطأ مفرد ومزدوج.

نلاحظ أنه بالرغم من وجود الخطأ المفرد و الخطأ المزدوج فإن التصميم المقترح قد أعطى نتائج صحيحة في كلا الحالتين أي أنه استطاع كشف الخطأ و تصحيحه. كذلك سنقوم باختبار عمل وحدة المنطق من خلال اختبار عمل دراتي AND ,OR والتأكد من قدرتها على الحفاظ على خرج صحيح عند حشر خطأ.

بين الشكل (14) النموذج المقترح لدارة AND متسامحة العطل مع حشر خطأF1 على خرجها ويبين الشكل (15) نتائج المحاكاة للتأكد من صحة عمل دارة AND .

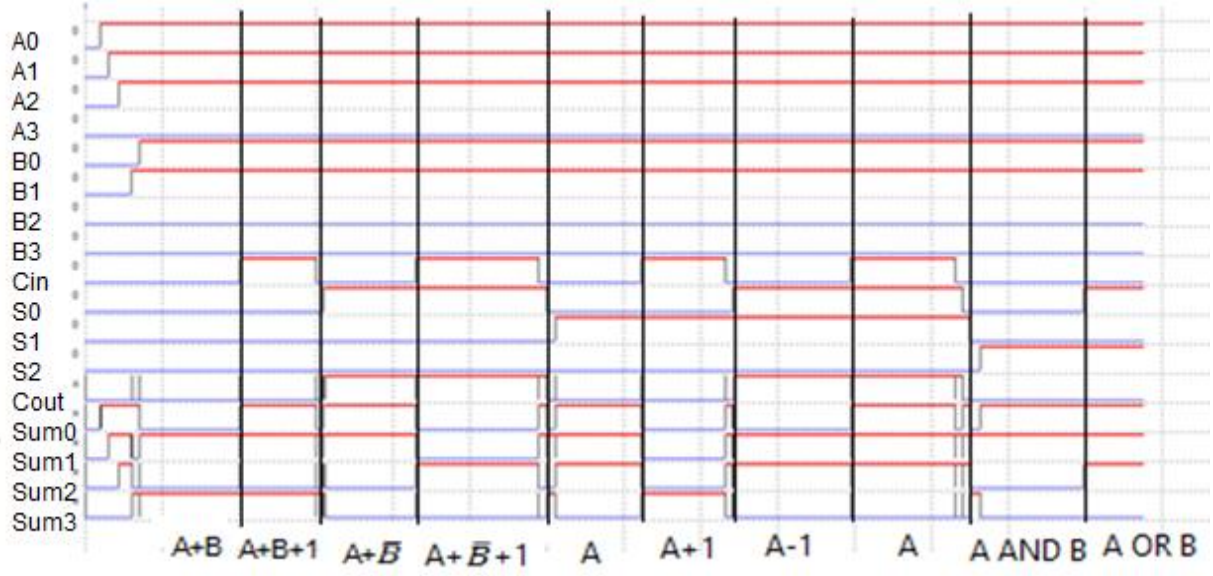


شكل(14): النموذج المقترح لدارة جامع AND متسامحة العطل مع حشر خطأ



الشكل(15):نتائج المحاكاة عند حشر خطأ على خرج بوابة AND

وبعد اختبار دارة OR بنفس الطريقة نلاحظ أن كل من بوابتي AND,OR ستعملان بشكل صحيح على الرغم من وجود خطأ وبالتالي وحدة المنطق ستعمل بالشكل الصحيح. يوضح الشكل (18) محاكاة واختبار وحدة الحساب والمنطق كاملة اعتمادا على جامع CSeA تم حقنه بخطأ مزدوج وعلى دارتي AND ,OR تم حقنهم بخطأ أيضا .



الشكل (18):نتائج محاكاة وحدة الحساب والمنطق

5- النتائج والمناقشة :

عادة ما تتم مقارنة المساحة بطرق مختلفة مثلا حسب عدد الترانزستورات، عدد البوابات أو التكنولوجيا المستخدمة. في هذا البحث سنقوم بمقارنة التصميم المقترح لجامع Self checking CSeA مع تصميم CSeA التقليدي وتصاميم self-checking CSeA التي تم اقتراحها في [12,13] اعتمادا على عدد الترانزستورات حيث يوضح الجدول (4) عدد الترانزستورات اللازمة للوحدات المستخدمة في التصميم، ويعرض الجدول (5) عدد الترانزستورات اللازمة لأحجام مختلفة من البتات للتصميم المقترح لاكتشاف الخطأ، أما الجدول (6) فسيقارن بين التصميم المقترح وتصاميم تم اقتراحها سابقا لجامع CSeA بالنسبة لأحجام مختلفة من البتات.

الجدول (4): عدد الترانزستورات اللازمة للوحدات المستخدمة في التصميم

الوحدة المستخدمة	عدد الترانزستورات
AND	2[16]
XOR	4[16]
MUX	2[16]
Full ADDER(FA)	9[15]
Function unit (Eq. (2))	30
Checker (5 XNOR)	20[16]
Module for final Cout (MOFC)	6
INV	2

الجدول (5): عدد الترانزستورات اللازمة لأحجام مختلفة من البتات للتصميم المقترح لاكتشاف الخطأ

	4Bit	6Bit	8Bit	16Bit
AND	3	5	7	15
XOR	3	5	7	15
MUX	4	6	8	16
FA	4	6	8	16
Function unit	4	6	8	16
Checker	20	3	40	80
(MOFC)	1	1	1	1
INV	1	1	1	1
المجموع الكلي للترانزستورات	270	386	538	1074

الجدول(6): المقارنة بين تصاميم مقترحة سابقا لجامع CSeA

عدد البتات	CSeA without self-checking[12]	CSeA design proposed by [12]	CSeA design proposed by [13]	تصميمنا المقترح
	عدد الترانزستورات	عدد الترانزستورات	عدد الترانزستورات	عدد الترانزستورات
4bit	284	328	286	270
6bit	420	490	426	386
8bit	556	652	566	538
16bit	1100	1300	1126	1074

يتضح من الجدول (6) ان CSeA المقترح للفحص الذاتي وكشف الاخطاء يتطلب مساحة أقل من تصميم CSeA التقليدي وكذلك عدد ترانزستورات اقل من التصاميم المقترحة في [12,13]، مثلا بالنسبة لجامع 4bit سيتم تقليل عدد الترانزستورات بنسبة 18% بالمقارنة مع التصميم المقترح في [12]، وبنسبة 6% بالمقارنة مع التصميم المقترح في [13]. بالإضافة لذلك فان التصميم المقترح قدم تحسينا ملحوظا بالمقارنة مع

التصميم المقترح في [12,13] حيث تمكن من اكتشاف الاخطاء المفردة والمزدوجة اينما حدثت ضمن بنية الجامع .

أما بالنسبة لإصلاح الخطأ فإن التصميم المقترح قادر على اصلاح الخطأ في حال حدوثه على الـ Sum أو Cout أو كليهما.

يوضح الجدول (7) عدد الترانزستورات اللازمة لأحجام مختلفة من التصميم المقترح بينما يوضح الجدول(8) المقارنة بين تصميمنا المقترح والتصميم المقترح ب [14] من حيث عدد الترانزستورات لأحجام مختلفة من البتات .

الجدول (7): عدد الترانزستورات اللازمة لأحجام مختلفة من البتات للتصميم المقترح لإصلاح الخطأ

	4Bit	6Bit	8Bit	16Bit
and	3	5	7	15
xor	3	5	7	15
mux	12	18	8	16
FA	4	6	8	16
Function unit	4	6	8	16
Checker	20	30	40	80
(MOFC)	1	1	1	1
INV	9	13	17	33
المجموع الكلي للترانزستورات	302	452	602	1127

الجدول(8): المقارنة مع تصميم مقترح سابقا لجامع CSeA قادر على الاصلاح الذاتي

	CSeA design proposed by [14]	تصمينا المقترح
4bit	496	302
6bit	740	452
8bit	984	602
16bit	1960	1127

نلاحظ انه بالنسبة لجامع 4bit تمكن التصميم المقترح من تقليل عدد الترانزستورات بنسبة 39% بالمقارنة مع التصميم المقترح في[14] وتعتبر هذا النسبة عامل مهم جدا حيث أن تقليل عدد الترانزستورات سيؤدي الى تقليل مساحة التصميم وتقليل الطاقة المستهلكة .

بالإضافة لذلك فإن وثوقية الخرج بالنسبة للتصميم المقترح في [14] ستكون 100% في حال حدوث خطأ مفرد، أما في حال حدوث خطأين في نفس الوقت فإن الوثوقية ستنخفض الى 85.82% كما يظهر في الجدول (9) وذلك بسبب عدم قدرة التصميم على اكتشاف واصلاح الخطأ المزدوج اذا حدث في نفس الوقت، أما تصميمنا المقترح سيتمكن من حل المشكلة واصلاح الأخطاء المفردة والمزدوجة التي تحدث في نفس الوقت وبالتالي فإن وثوقية هذا التصميم ستكون 100%عند حدوث خطأ مزدوج .

الجدول(9):مقارنة الوثوقية مع تصميم مقترح سابقا لجامع CSeA قادر على الاصلاح الذاتي

	CSeA design proposed by [14]	تصمينا المقترح
وثوقية الخرج في حال حدوث خطأ وحيد	100%	100%
وثوقية الخرج في حال حدوث خطأين في نفس الوقت	85.82%	100%

اعتمادا على ما سبق فإن وحدة الحساب والمنطق المقترحة ستكون قادرة على احتواء الخطأ اينما حدث في وحدة الحساب أو وحدة المنطق وستحافظ على خرج صحيح في حال حدوث خطأ مفرد أو مزدوج في نفس الوقت ، وسيتم تقييم أداء وحدة الحساب والمنطق المقترحة اعتمادا على تكنولوجيا CMOS 0.90nm باستخدام برنامج Microwind 3.1 ، ويوضح الجدول (10) الطاقة المستهلكة وعدد الترانزستورات اللازمة للتصميم المقترح .

الجدول(10): الطاقة المستهلكة وعدد الترانزستورات اللازمة للتصميم المقترح

	الطاقة المستهلكة	عدد الترانزستورات
fault tolerance 4bit ALU	3.26 mw	448

6. الاستنتاجات والتوصيات

في هذا البحث تم تصميم دارة حساب ومنطق(ALU) متسامحة العطل اعتمادا على دارة جامع (CSeA) متسامحة العطل وقادرة على الاصلاح الذاتي باستخدام تقنية Static CMOS. وتمت عملية حقن مجموعة من الأخطاء المفردة و المزدوجة من النوع SA1 و SA0.

-1

وبمقارنة نتائج المحاكاة نلاحظ أن هذا النموذج أفضل من حيث المساحة على الرقاقة من التصاميم السابقة التي درست بنفس المجال. و من ناحية أخرى نلاحظ ان هذا التصميم قادر على كشف الأخطاء الفردية و المزدوجة بنفس الوقت بوثوقيه 100%.

2- يعتبر هذا النموذج المصمم ذو وثوقية عالية لاستخدامه في معالجات الإشارة الرقيمة وأنظمة السلامة والأنظمة الفضائية، فالحصول على نتائج دقيقة ولا تحوي أخطاء في أنظمة كهذه أمر مهم، بالإضافة إلى أنه من أجل المعالجات المعقدة، يظهر التأثير الواضح لانخفاض عدد الترانزستورات مما يخفض من استطاعة الشريحة ويقلل من حجم الشريحة المتسامحة العطل.

المراجع:

[1] Essa,S.(2019).*The power consumed and the number of transistors required for the proposed design*, Tartous University Journal of Research and Learning Studies, Engineering Science Series , Vol 3 (4).

[2]] Essa,S.(2020). *Diagnosis of faults in digital circuits based on artificial immune systems*, Tartous University Journal of Research and Learning Studies, Engineering Science Series , Vol 4 (9).

[3] Saleh,S.(2019). *Analysis and performance evaluation of 1-bit Full Adder circuit based on static-CMOS and GDI technology, and the impact of the technology used in manufacturing on the performance*, Tartous University Journal of Research and Learning Studies, Engineering Science Series , Vol 3 (4).

[4] Saleh,S.(2020). *Design and performance evaluation of fault tolerant full Adder circuit based on CMOS and GDI technology* , Tartous University Journal of Research and Learning Studies, Engineering Science Series ,Vol 4 (4).

[5] Akabati ,R .(2020). *Splitting data between SPM memory and main memory in an embedded system* , Tartous University Journal of Research and Learning Studies, Engineering Science Series , Vol 4 (6).

[6] Junming, L., Yan, S., Zhenghui, L., & Ling, W. (2001, October*). A novel 10-transistor low-power high-speed full adder cell.* In *2001 6th International Conference on Solid-State and Integrated Circuit Technology. Proceedings (Cat. No. 01EX443)* (Vol. 2, pp. 1155-1158). IEEE.

[7] Jiang, Y., Al-Sheraidah, A., Wang, Y., Sha, E., & Chung, J. G. (2004*). A novel multiplexer-based low-power full adder*. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *51*(7), 345-348.

[8] Chang, C. H., Zhang, M., & Gu, J. (2003, September). *A novel low power low voltage full adder cell*. In *3rd International Symposium on Image and Signal Processing and Analysis, 2003. ISPA 2003. Proceedings of the* (Vol. 1, pp. 454-458). IEEE.

[9] Weste, N. H., & Harris, D. (2015). *CMOS VLSI design: a circuits and systems perspective*. Pearson Education India.

[10] Rabaey, J. M., Chandrakasan, A. P., & Nikolić, B. (2003). *Digital integrated circuits: a design perspective* (Vol. 7). Upper Saddle River, NJ: Pearson Education.

[11] Kaur, S., Singh, B., & Jain, D. K. (2015). *Design and performance analysis of various adders and multipliers using GDI technique*. *International Journal of VLSI design & Communication Systems (VLSICS)*, *6*(5), 45-56

[12] Vasudevan, D. P., Lala, P. K., & Parkerson, J. P. (2007*). Self-checking carry-select adder design based on two-rail encoding*. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(12), 2696-2705.

[13] Akbar, M. A., & Lee, J. A. (2013, September). *Self-checking carry select adder with fault Localization*. In *2013 Euromicro Conference on Digital System Design* (pp. 863-869). IEEE

[14] Akbar, M. A., & Lee, J. A. (2014). *Self-repairing adder using fault localization. Microelectronics Reliability*, *54*(6-7), 1443-1451.

[15]Gotam, S., Ahmed, I., Ramola, V., & Kumar, R. (2014*). A New design of 1-bit full adder based on XOR-XNOR gate*. *International Journal of Enhanced Research in Science Technology & Engineering*, *3*(6), 81-85.

[16] Akbar, M. A., Wang, B., & Bermak, A. (2020). *Self-Repairing Hybrid Adder With Hot-Standby Topology Using Fault-Localization*. *IEEE Access*, *8*, 150051-150058.