تصميم معالج رياضي بأسلوب خط الأنابيب ومضاعفة السرعة وتنفيذه باستخدام FPGA

د. باسل شكر محمود قتيبة عبد الله حسن
كلية الهندسة / جامعة الموصل

الخلاصة

في هذا البحث تم تصميم معالج رياضي بأسلوب خط الأنابيب باستخدام لغة (VHDL)، حيث تم عمل محاكاة للمعالج باستخدام برنامج المحاكاة ISE10.1، بعدها نُفذ على رقاقة FPGA في لوحة SPARTAN3E حيث تم تسقيط التصميم باستخدام المنفذ USB. صُمّم كلٌّ من سجلَي جلب وتحليل الإيعاز وذاكرتيهما المؤقتتين بحيث تتم عملية القراءة والكتابة على الموقع نفسه خلال النبضة نفسها (استخدام إحدى حافتي النبضة للقراءة والأخرى للكتابة) لغرض زيادة سرعة المعالج، وقد تم استخدام منفذ JTAG لتحديث البيانات والايعازات المخزونة في الذاكرة الرئيسة للمعالج عن طريق استخدام دائرة مراقبة. تحتوي الذاكرة الرئيسة للمعالج على منفذين أحدهما مخصص لتحديث البيانات والايعازات المخزونة فيها والثاني مخصص لقراءة البيانات والايعازات منها. ولغرض زيادة سرعة التنفيذ تم بناء كلّ من وحدتي تحليل وتنفيذ الإيعاز بحيث يمكن تنفيذ جميع العمليات بشكل متوازٍ.

إن عدد العمليات التي يقوم المعالج بتنفيذها هي 30 عمليةً وتشمل الدوال المثلثية ودوال القطع الزائد والجذر التربيعي، وعملياتُ الحساب والمنطق ودوائر المقارنة وعملياتُ النقل وعمليتي التزحيف والتدوير المتوازيتين. أن عدد النبضات التي تحتاجها الدوال الرياضية تتراوح بين 17-23 نبضة للحصول على أول إخراج، ثم بعد كل نبضة هناك إخراج عندما تكون قيم إدخالات الدوال متسلسلة ومستمرة. أما العمليات الرياضية والمنطقية (مثل الجمع والطرح والضرب .... الخ) فتحتاج إلى نبضة واحدة للحصول على الإخراج. ان اعلى تردد لتشغيل الرقاقة المصممة هو 133,820 ميكا هرتز، اي ان انتاجية الرقاقة هي 133,820 MFlops.

ولغرض عرض جميع إخراجات المعالج على شاشة الحاسوب تم استخدام منفذ VGA، التصميم الكلي لهذا المعالج استغل 98% -عند ربطه بمنفذ VGA- من حجم الرقاقة الموجودة على لوحة SPARTAN3E.

Design a piplined math processor, doubling its speed and implementing it on FPGA

Basil Shukr Mahmood(Ph.D.) **Qutaiba Abdullah Hassan**
Prof. of microprocessors **Msc. of Computer Eng.**
University of Mosul / Mosul / Iraq
Basil_mahmood@yahoo.com Quteiba_qa@yahoo.com

Abstract

In this paper, a pipeline math processor is designed using VHDL, where doing simulation for processor by using simulation program ISE10.1. The processor is implemented on the FPGA chip in a panel SPARTAN3E, where it has been downloaded by using USB port. The register and buffer memory of each of the fetch and decoder units are designed such that reading and writing operations for the same location are performed during one clock cycle (each clock pulse edge is used for one operation). JTAG port is used to update the data and instructions stored in the main memory via monitor circuit. The main memory of the processor contains two ports, one of them is used to update the data and instructions and the other is used to read data and instructions. For the purpose of increasing the speed, decode and execute units are built so that all operations can be executed in parallel.

The number of operations that can be executed on the processor are 30 operations including triangular functions, hyperbolic functions, square root, ALU, comparison operations, move operation, and parallel shifting and rotation operations. The number of clocks that is required by the mathematical functions, ranges between 17-23 clocks for the first output and then each clock has it's output when the values of input functions are sequential and continuous. The remaining operations (such as addition, subtraction, multiplication and division) need only one clock for each output. The maximum operating frequency for the design chip was found to be 133.820 MHz, therefore its throughput is 133.820 MFlops. For the purpose of displaying all the processor outputs on a computer screen, the VGA Port is used. The overall design of this processor occupies 98% (when the processor is connected with VGA port) from the volume of the FPGA chip on board SPARTAN3E.

أستلم: 17 – 11 - 2013 قبل: 6 – 3 - 2013

1. المقدمة :

في هذا البحث تم تصميم معالج رياضي باسلوب خط الانابيب وتنفيذه باستخدام رقاقة FPGA كما تم استخدام الحافة الصاعدة والنازلة للنبضة لغرض مضاعفة سرعة المعالج، وحدة المعالجة المركزية Central Processing Unit ((CPU)) هي قلب الحاسوب الآلي والتي تحتوي على مجموعة من الدوائر المنطقية تقوم بتنفيذ مجموعة من ايعازات برامج الحاسوب أو عمليات حسابية معينة ثم تقوم بخزن ناتج التنفيذ في ذاكرة معينة أو إرسالها إلى أجزاء الحاسوب الأخرى[1]. إن أول معالج دقيق تم تصنيعه كان من قِبَل شركة Intel في العام 1971م، حيث أطلقت عليه الشركة اسم INTEL4004، وهذا المعالج يحتوي على أكثر من 2000 ترانسستر، وفي العام 1978م قامت الشركة نفسها بصناعة معالج آخر احتوى على 29000 ترانسستر ويتعامل هذا المعالج مع 16 بتاً، تتراوح سرعته بين (5 - 10) ميكا هيرتز، وقد أطلقت الشركة عليه اسم INTEL8088 والذي استخدمته شركة IBM لتصنيع أول حاسوب آلي [2].

في العام 1980م قام الباحثون John Bruno و John W.Jones وKimming SO [3] بدراسة المشكلة التي تحدث في أثناء القيام بعملية جدولة (Scheduling) مجموعة من المهام (Tasks)، إما لمعالج مصمم بأسلوب خط الأنابيب أو لأكثر، حيث افترضوا وجود جدولة مثالية لمهام معينة في أثناء القيام بعملية المطابقة، ثم اظهروا ثلاثة أصناف أو أنواع من المشاكل توضح كيفية حدوث المشكلة في أثناء قيام المعالج بتنفيذ مهام معينة، وعدّوا كل مشكلة من مشاكل عملية جدولة مجموعة من المهام، موجودة في النظام الذي يحتوي : إما على معالج مصمم بأسلوب خط الأنابيب أو أكثر.

في العام 1990م قام الباحثون Mitsuhisa Sato و Shuichi Ichikawa و Eiichi Goto [9] بتقديم تقييم لمعالج مصمم بأسلوب خط الأنابيب ذي السرعة الفائقة (Super Pipeline Processor)، ثم فحصوا أداء هذا المعالج باستخدام لغة فورتران (Fortran) (أي استخدموا هذه اللغة لعمل محاكاة لهذا المعالج)، وقارنوا نتائج الفحص مع معالج مُصمّم بأسلوب خط الأنابيب، حيث لاحظوا أن هذا المعالج يعطي أعلى أداء إذا ما قُورِن بالمعالج المصمم بأسلوب خط الأنابيب، حيث إن أسلوب خط الأنابيب المستخدم في هذا البحث، مَكّن المعالج من تنفيذ عدة ايعازات بشكل متوازٍ في آن واحد.

في العام 1993م قام الباحثان Antonio M. Gonzalez و Jose M. Llaberia [10] باستخدام آلية معينة لتقليل كلفة إيعاز القفز (reduce the cost of branch instruction) في معالج مُصَمّم بأسلوب خط الأنابيب. حيث قاما بدراستها ووصفها وتقييمها، وهذه الآلية تسمى بالكلفة المثالية للقفز (Cost Optimization of Branch)، إذ تعتمد على الاستخدام المتعدد لجلب الإيعاز المسبق والحساب الأولي لعنوان الهدف وزمن القفز والتنفيذ المتوازي لإيعاز القفز، واستخدما أنموذجاً معيناً لتنفيذ هذه الآلية مكّنتهما من قياس كفاءة هذه الآلية بكلفة حسابية منخفضة جداً. واستنتجا أن هذه الآلية تعطي أداءً عالياً للمعالج عند مقارنتها مع آليات أُخَر.

في العام 1996م قام الباحث باسل شكر[4] بتصميم معالج رسومي لتنفيذ مجموعة من الوظائف مثل رسم مضلع وخطوط ومنحنيات ودوائر وهذا المعالج يقوم بتنفيذ هذه العمليات عن طريق مجموعة من الايعازات، وهذه الايعازات تكون من نوع الايعازات المتقلصة (RISC)، ولغرض زيادة سرعة وحدة المعالجة التي تقاس بوحدة الايعازات التي يتم تنفيذها بوحدة الزمن، استخدم تقنية خط الأنابيب المتوازية حيث تتم وحدة المعالجة هذه عن طريق أربع قنوات متوازية : الأولى جلب الإيعاز، الثانية تحليل الإيعاز، الثالثة تنفيذ الإيعاز والأخيرة خزن الرسم في ذاكرة فيديوية.

في العام 2002م صمَّم الباحث Charles Brej [5] معالجاً متكاملاً من نوع (MIPS R3000) والذي مازال يستخدم حتى يومنا هذا والسبب في تصميم هذا المعالج هو كون عملية فك شفرات ايعازاته واضحة وسهلة، فضلاً عن أن هذا النوع من المعالجات يحتوي على ذاكرة رئيسة (RAM) وذاكرة وسيطة (Cache) ودائرة لإدارة الذاكرة (Control Memory Management) ومعالج مساعد بسيط (Coprocessor)، حيث قام بتنفيذ هذا المعالج على رقاقةFPGA وفحص لبعض المكونات الرئيسة لهذا المعالج.

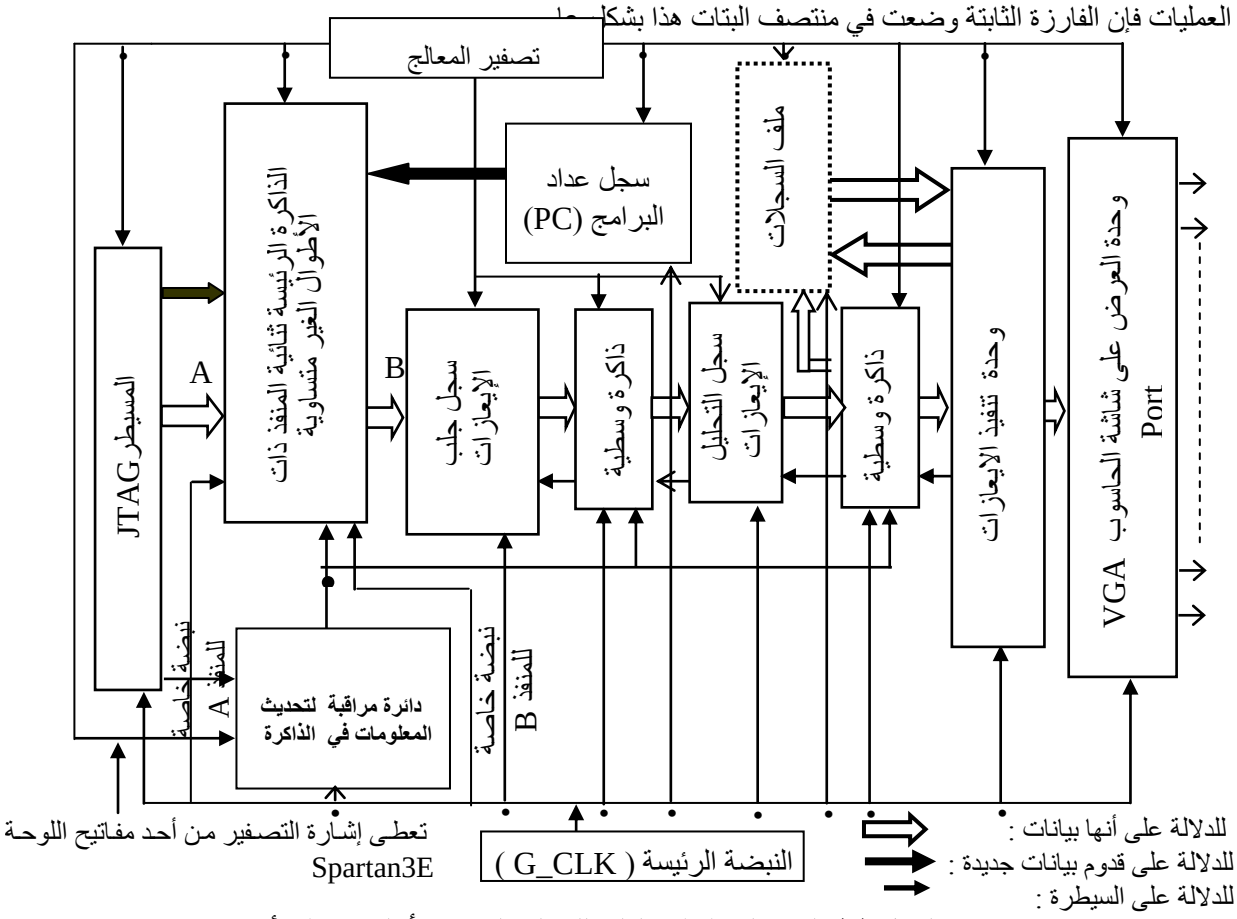
وفي العام 2006م صمَّم الباحث Jonathan Barre مع مجموعة من الباحثين [6] معالجاً بأسلوب خط الأنابيب ذي السرعة الفائقة، وهذا التصميم يزيد من سرعة تنفيذ مستوى الايعازات المتوازية وتسهيل عملية تحليله، ثم قاموا بعمل محاكاة لهذا المعالج بعد إجراء عملية التحسين عليه، ولاحظوا أنه أعطى نتائج أفضل حتى من معالج واقعي معتبر. وذكروا أن سبب استخدام هذه المعماريات في الوقت الحاضر هو أن بعض التطبيقات أصبحت من الصعب جداً تخمين أطول مدة زمنية لازمة لتنفيذها كون هذه التطبيقات أصبحت معقدة نتيجة التطور.

وفي العام 2008م قام الباحث أحسان عبد الستار [7] بتصميم وتنفيذ العديد من المعماريات المقترحة لحساب عدد من الدوال الرياضية المعقدة القابلة للاشتقاق، وذلك باستخدام خوارزميتي كوردك وجداول المقارنة. كما قام الباحث بتنفيذ تحويلات المحاور ("قطبي" إلى "ديكارتي") و ("ديكارتي" إلى "قطبي ") بوصفها تطبيقاً لمعماريات الدوال المنفذة. حيث تم تنفيذ الكيان المادي لهذه المعماريات على رقاقة FPGA. ولاحظ أن أفضل طريقة لتنفيذ خوارزمية كوردك هي المعمارية المتوازية بأسلوب خط الأنابيب.

2. تصميم المعالج :

إن الطريقة المستخدمة في تصميم المعالج هي طريقة معالجات Post RISC (أي تصميم وحدات المعالج بأسلوب خط الأنابيب وجعل وحدة التنفيذ تُنَفِذّ عملياتها بشكلٍ متوازٍ)، فضلاً عن جعل وحدة التحليل تعمل بشكلٍ متوازٍ. أما العمليات التي يقوم هذا المعالج بتنفيذها فهي : الدوال المثلثية (الجيب والجيب تمام والظل ومعكوس الظل) ودوال ألقطع الزائد (جيب وجيب تمام وظل ومعكوس ظل القطع الزائد) والجذر التربيعي، وعمليات الحساب والمنطق (ALU)(عملية القسمة والضرب والجمع والطرح وAND وNAND وOR وXOR وNOR وNOT) ودوائر المقارنة (حالة الأكبر والأصغر والمساواة) وعمليات النقل (من الذاكرة إلى السجل ومن سجل إلى سجل آخر) وعمليات التزحيف والتدوير المتوازيتين. لقد تم استخدام خوارزمية كوردك في تنفيذ الدوال المثلثية ودوال القطع الزائد ودالة الجذر التربيعي (عدا دالتي الظل وظل القطع الزائد، حيث تم تنفيذهما عن طريق قسمة الجيب على الجيب تمام للحصول على الظل وجيب على جيب تمام القطع الزائد للحصول على ظل القطع الزائد). والشكل (1) يوضح الرسم التخطيطي الكلي للمعالج المُصَمَّم بأسلوب خط الأنابيب.

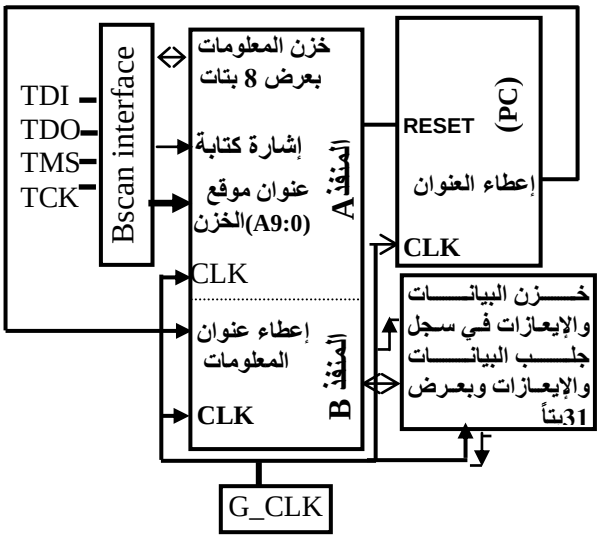
لقد استخدم تمثيل النقطة الثابتة (fixed point) في التصميم، وهي مرحلة وسطية بين النقطة العائمة (floating point) والأعداد الصحيحة (integers)، إن سبب استخدام النقطة الثابتة في تمثيل بيانات المعالج هو كونها سريعة في الحسابات إذا ما قورنت بالنقطة العائمة لكن عملية تمثيل العدد الذي فيه أرقام كثيرة بعد الفارزة يحتاج إلى عدد كبير من البتات لتمثيله بالدقة المطلوبة. استخدمت النقطة الثابتة في تمثيل جميع بيانات الإدخال والإخراج على المعالج، حيث وضعت في أماكن مختلفة على البيانات الداخلة والخارجة على عمليات المعالج، ففي عمليتي الجيب والجيب تمام وضعت النقطة الثابتة بين البت الثالث عشر والثاني عشر حيث يمثل البت الخامس عشر بت الإشارة، هذا بالنسبة لإدخال الجيب والجيب تمام، أما في إخراجهما فاستُخدِم بت واحدة لتمثيل العدد الصحيح والباقي لتمثيل القيمة بعد الفارزة (إن السبب الرئيس في وضع النقطة الثابتة في هذا الموضع هو كون أعلى قيمة إخراج يمثلها الجيب والجيب تمام هي واحد أي تحتاج إلى بت واحد لتمثيل العدد الصحيح وباقي البتات (0-13) استخدمت لتمثيل الأرقام التي هي بعد الفارزة)، أما دالة معكوس الظل ومعكوس ظل القطع الزائد فقد استخدم نفس تمثيل الجيب والجيب تمام إلا أن إدخالهما مُثِلّ ببت واحد فقط للعدد الصحيح وإخراجهما مُثِلّ ببتين للعدد الصحيح وبت آخر للإشارة وباقي البتات استخدمت لتمثيل القيمة بعد الفارزة، أما باقي العمليات فإن الفارزة الثابتة وضعت في منتصف البتات هذا بشكل عام.



الشكل (1) الرسم التخطيطي الكلي للمعالج المصمم بأسلوب خط الأنابيب

1.2 الذاكرة الرئيسة (Main Memory) :

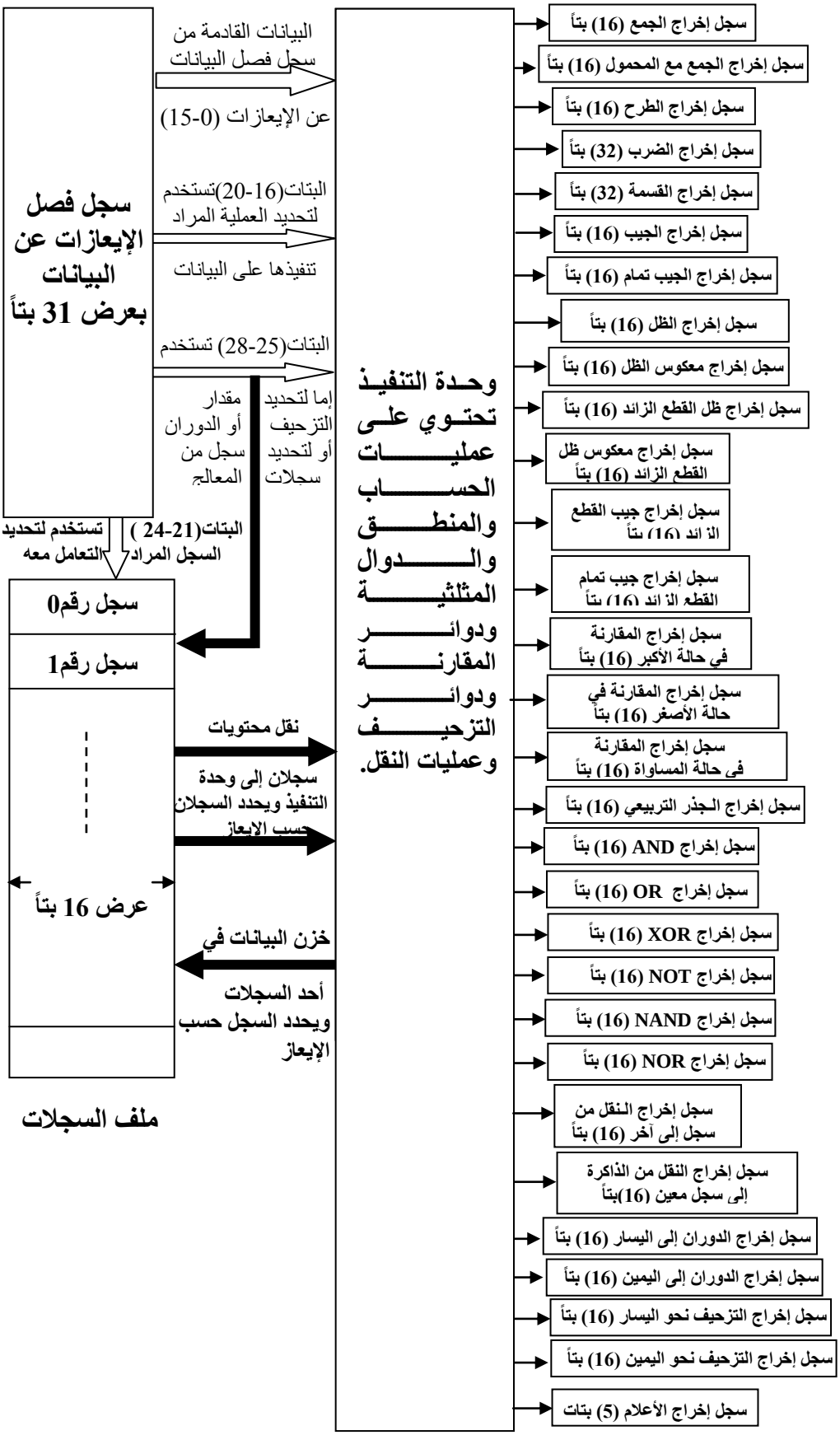
إن الذاكرة الرئيسة المستخدمة في هذا البحث هي من نوع ثنائية المنفذ غير متساوية الأطوال (mismatch length) ، إذ استُخدم أحد المنافذ (المنفذ A) لتحديث البيانات والإيعازات المخزونة فيها عن طريق استخدام برنامج JTag Loader، والمنفذ الآخر (المنفذ B) استُخدِم لقراءة البيانات والإيعازات منها عن طريق سجل عداد البرامج (PC)، حيث يعطي هذا السجل عنوان البيانات والإيعازات المخزونة في الذاكرة لكي تظهر على المنفذ (B) ثم يقوم سجل جلب الإيعازات بخزنها في ذاكرته المؤقتة، والشكل (2) يوضح ربط المعالج بالذاكرة الرئيسة.



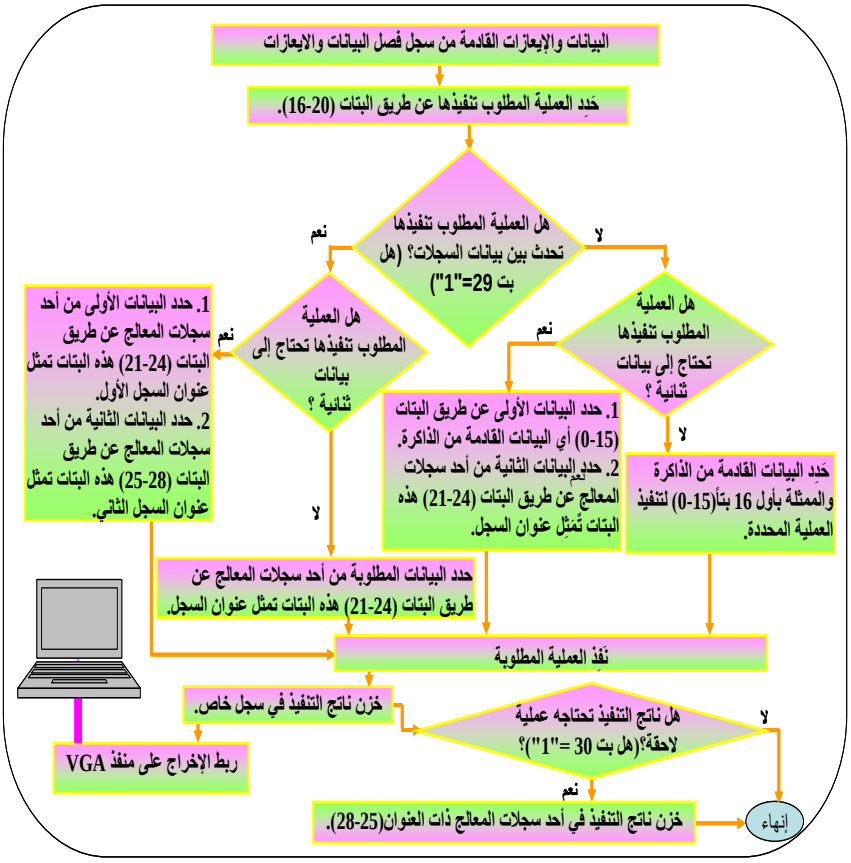
الشكل (2) ربط المعالج مع الذاكرة الرئيسة وJTAG

2.2هيئة الإيعاز :

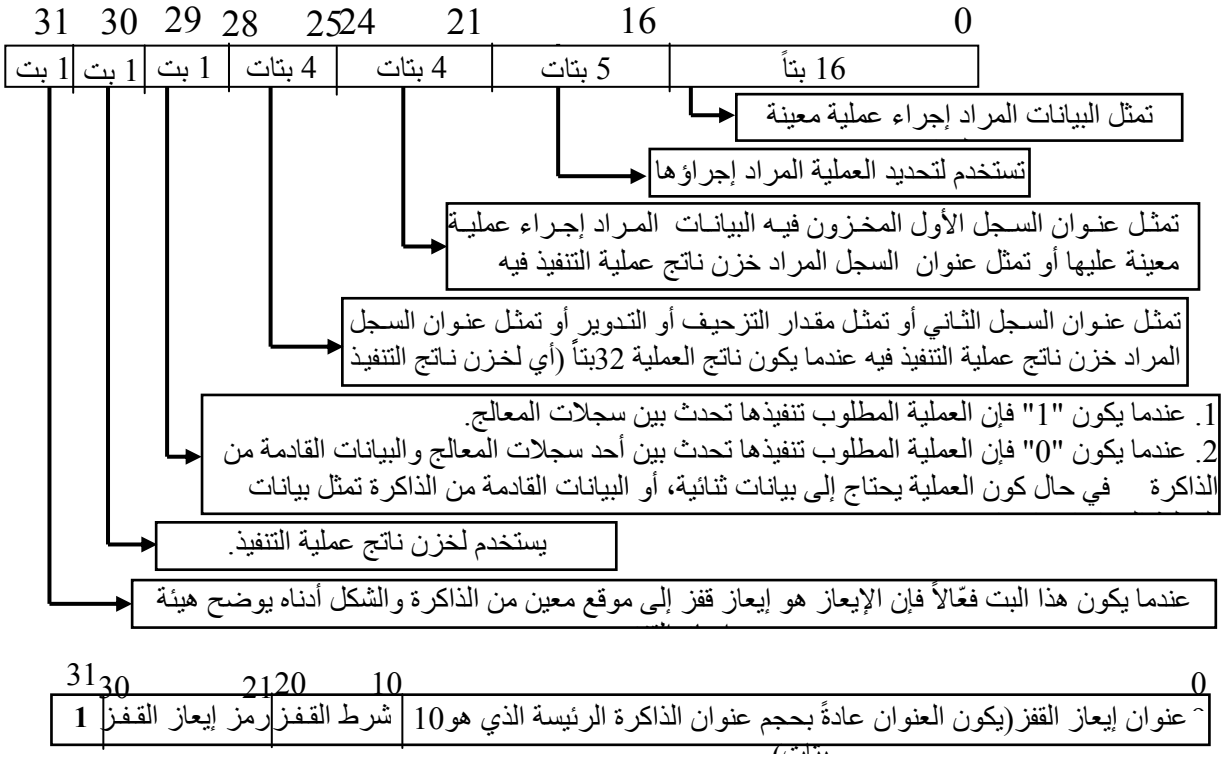
البيانات والإيعازات القادمة من الذاكرة مُتَكونة من 32 بتاً أول 16 بتاً منها تمثل قيمة البيانات المراد إجراء عملية معينة عليها والبتات (16-20) تحدد العملية المراد إجراؤها والبتات (21-24) تمثل عنوان السجل المراد التعامل معه إما قراءة البيانات منه أو كتابة البيانات عليه هذا إذا كانت العملية المراد إجراؤها تحتاج إلى بيانات أحادية، أما إذا كانت العملية تحتاج إلى بيانات ثنائية فإن هذه البتات تمثل عنوان السجل الأول المخزون فيه البيانات الأول المراد التعامل معها أو تمثل عنوان السجل المراد خزن ناتج عملية التنفيذ فيه هذا إذا كان ناتج التنفيذ متكون من 16 بتاً، أما إذا كان متكون من 32 بتاً فإن هذه البتات يمثل عنوان السجل المراد خزن البتات (0-15) الخاصة بناتج التنفيذ، والبتات (25-28) تستخدم إما لتحديد مقدار تزحيف أو تدوير البيانات الداخلة على العملية المحددة أو لتحديد عنوان السجل الثاني في حالة كون العملية المطلوب تنفيذها تحتاج إلى بيانات ثنائية أو لتحديد عنوان السجل المراد خزن ناتج التنفيذ فيه عندما يكون ناتج التنفيذ مُتكون من 32 بتاً (أي يمثل عنوان السجل المراد خزن البتات (16-31) الخاصة بناتج التنفيذ)، والبت 29 يستخدم لتحديد البيانات الداخلة على عملية معينة، فعِندما يكون هذا البت فعالاً فإن البيانات الداخلة تمثل البيانات المخزونة في أحد سجلات المعالج هذا إذا كانت العملية المطلوب تنفيذها تحتاج إلى بيانات أحادية، أما إذا كانت تحتاج إلى بيانات ثنائية فسيتم تحديد بياناتها من سجلين من سجلات المعالج (البيانات الأولى تحدد من أحد السجلات إذ يحدد عنوان هذا السجل عن طريق البتات (21-24) والبيانات الثانية تحدد من سجل آخر من سجلات المعالج عنوانه يحدد عن طريق البتات (24-28))، وفي حالة كون هذا البت غير فعال فالبيانات الداخلة ستمثل البيانات القادمة من الذاكرة الرئيسة والممثلة بأول 16 بتاً (0-15) أما إذا كانت العملية تحتاج إلى بيانات ثنائية فإن البيانات القادمة من الذاكرة ستمثل البيانات الأولى والبيانات الثانية تحدد من أحد سجلات المعالج إذ يحدد السجل عن طريق العنوان(21-24)، أما البت 30 فيستخدم لغرض خزن ناتج عملية معينة تم تنفيذها، هذا البت (بت 30) مهم لمعالجة مشكلة البيانات المعتمدة، فعندما يكون هناك إيعاز يحتاج ناتج تنفيذ إيعاز معين سابق يتم تفعيل هذا البت ويُخزَن ناتج تنفيذ الإيعاز المطلوب في سجل معين عنوانه يحدد عن طريق البتات (25-28). يوضح الشكل (3) الرسم التخطيطي لوحدة تنفيذ المعالج مع ملف السجلات ووحدة التحليل (حيث أن البتات (16-20) القادمة من الذاكرة الرئيسة تستخدم لتحديد الايعازات وحسب الشكل (3) ادناه، فان اول عملية (عملية الجمع) تحدد عندما تكون قيم البتات (00000) وهكذا حتى اخر عملية تحدد عندما تكون قيم البتات (11101) حيث تمثل الاعلام)، ويوضح الشكل (4) المخطط الانسيابي لعملية التحليل الكاملة للإيعاز وكيفية تحديد كُل عملية وبياناتها، بينما يوضح الشكل (5) الهيئة أو البنية العامة للإيعازات المسيطرة على البرنامج.



الشكل(3) الرسم التخطيطي لوحدة تنفيذ المعالج مع ملف السجلات ووحدة التحليل



الشكل (4) المخطط الانسيابي لعملية التحليل الكاملة

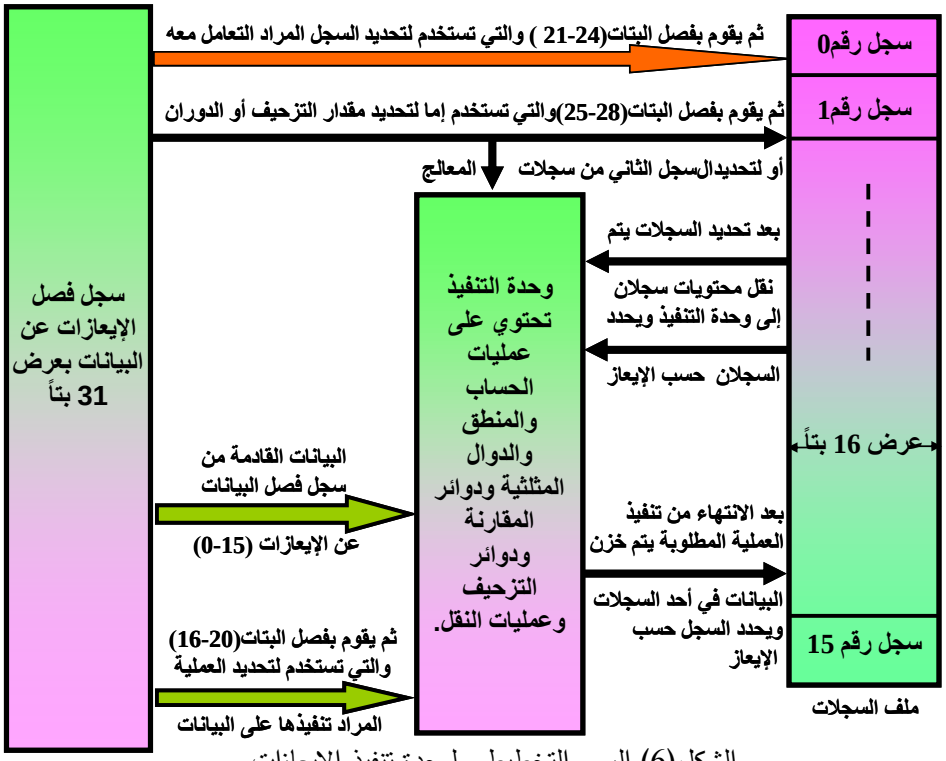


الشكل (5) الهيئة العامة للإيعازات المسيطرة على البرنامج

3.2 وحدة تنفيذ الإيعازات :

تُعَدُّ وحدة تنفيذ الإيعازات من أعقد وحدات المعالج، ففي هذه الوحدة تتم كل العمليات المراد إجراؤها[8]. تم تصميم وتنفيذ 30 عملية، حيث تُنَفَّذ هذه العمليات حسب الإيعازات القادمة من الذاكرة الرئيسة عبر سجلي جلب وتحليل الإيعاز وذاكرتيهما المؤقتتين وصولا إلى وحدة التنفيذ مع البيانات المراد إجراء العمليات اللازمة عليها.

الشكل (6) يوضح الرسم التخطيطي لوحدة تنفيذ الإيعازات، إن كل عملية لها سجلاً خاصاً يخزن فيه ناتج تنفيذها، ويمكن خزن ناتج التنفيذ في أحد سجلات المعالج، عادةً يخزن ناتج تنفيذ عمليات معينة في أحد سجلات المعالج عندما تكون هناك إيعازات أخرى تحتاج ناتج تنفيذ العمليات السابقة وهذا ما يطلق عليه اسم البيانات المعتمدة (مثلاً عند جمع قيمتين معينتين ثم ضرب ناتج الجمع في قيمة أخرى).



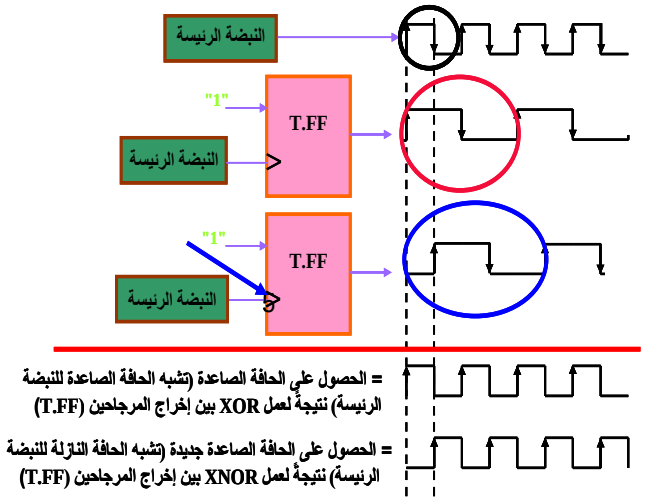
الشكل(6) الرسم التخطيطي لوحدة تنفيذ الايعازات

- **مضاعفة عرض النبضة :**

إن عملية استخدام الحافة النازلة والصاعدة بشكل مباشر في تصميم معين باستخدام لغة وصف الكيان المادي لا يمكن تركيبه (أي إن عملية استخدام الحافة الصاعدة والنازلة للنبضة في تنفيذ عمليات معينة داخل تصميم معين لا يمكن تركيبه بسبب تفعيل حافتي النبضة وإنما تستخدم الحافتان فقط لعمل محاكاة للتصميم)[9]، لذلك تم بناء عملية (Process) خاصة تقوم بتوليد الحافة الصاعدة والنازلة للنبضة وإعطاء حالاتها (حالات النبضة المُوّلدة الصفر والواحد) لمتغيرات معينة. هذه المتغيرات هي التي تمثل حافتي النبضة الصاعدة والنازلة وبذلك استخدمت حافتا النبضة في بناء وتنفيذ وعمل محاكاة لوحدات المعالج الرياضي المقترح المصمم بأسلوب خط الأنابيب.

لم تعطِ شركة Xilinx الطريقة المستخدمة في بناء الذاكرات المطمورة داخل لوحاتها التي من أهم مزاياها القراءة والكتابة في آن واحد على مواقع مختلفة أو متشابهة من الذاكرة، وبعد عدة محاولات نُفّذت عدة أفكار لاستخدام الحافة الصاعدة والنازلة للنبضة لقراءة وكتابة البيانات على النبضة نفسها، إلا أن هذه المحاولات تنجح فقط في محاكاة للتصميم وليس عند التركيب، لذلك استُخدمت فكرة ذاكرة نسبة البيانات المضاعفة (DDR) Double Data Rate)) لبناء الذاكرات المؤقتة [10]، والشكل (7) يوضح كيفية توليد حافتي النبضة المستخدمة في بناء ذاكرتي سجلي جلب وتحليل الإيعاز، إذ استخدمت الحافة الصاعدة للنبضة لكتابة البيانات والإيعازات، بينما استخدمت الحافة النازلة للنبضة لقراءة

البيانات والإيعازات، وعُملت مصفوفة من نوع ذاكرة وصول عشوائي (RAM) ضمن العملية المستخدمة في التركيب، حيث تبدأ هذه الذاكرة بقراءة وكتابة الإيعازات والبيانات عند شرط حافة النبضة (عند الحافة الصاعدة للنبضة كتابة وعند الحافة النازلة للنبضة قراءة).



الشكل (7) توليد حافتي النبضة المستخدمتين في بناء ذاكرتي سجلي جلب وتحليل الإيعاز

ولغرض زيادة سرعة التحليل التي تؤدي إلى زيادة سرعة المعالج المصمم، تم تنفيذ وحدتي التحليل والتنفيذ بشكل متوازٍ أي إن كل إيعاز عند تحليله يصل إلى جميع العمليات الموجودة في وحدات التنفيذ في آن واحد، وكل عملية من العمليات الموجودة في وحدة التنفيذ تعمل بشكل مستقل عن غيرها من العمليات (أي تعمل بشكل متوازٍ)، إن الأسلوب المستخدم في تنفيذ هاتين الوحدتين بشكل متوازٍ هو استخدام عدد من العمليات (Processes)، إذ من مبادئ لغة VHDL أن أجزاء التصميم بين العمليات (Process) تنفذ بشكل متوازٍ [11].

4.2 بناء وحدة التنفيذ :

تقوم هذه الوحدة بتنفيذ جميع العمليات، كل عملية من العمليات التي تنفذها هذه الوحدة تم بناؤها بأسلوب معين. يتحكم كلّ من البتّين 29 و30 في عملية القراءة والكتابة على ملف السجلات على التوالي. إن بيانات الإيعاز المراد إجراء عملية معينة عليها إما تحدد من أحد سجلات المعالج عندما يكون البت 29 فعالاً أو البيانات القادمة من الذاكرة الرئيسة هي التي تمثل بيانات الإيعاز المراد إجراء عملية معينة عليها كما تم ذكرها سابقاً، أما البت 30 فيستخدم لخزن ناتج تنفيذ عملية معينة في أحد سجلات المعالج. عادةً يستخدم هذا البت عندما يكون هناك إيعاز يحتاج إلى ناتج تنفيذ عملية سابقة معينة (وفي الدوال المثلثية توضع إشارة على الإخراج وهي إشارة (Ready)، فعندما تكون هذه الإشارة فعالة والبت 30 فعالاً يتم خزن ناتج الدالة المثلثية في أحد سجلات المعالج).

وفيما يأتي شرح عن بعض عمليات المعالج المصمم :

1.4.2 دالتا الجيب والجيب تمام :

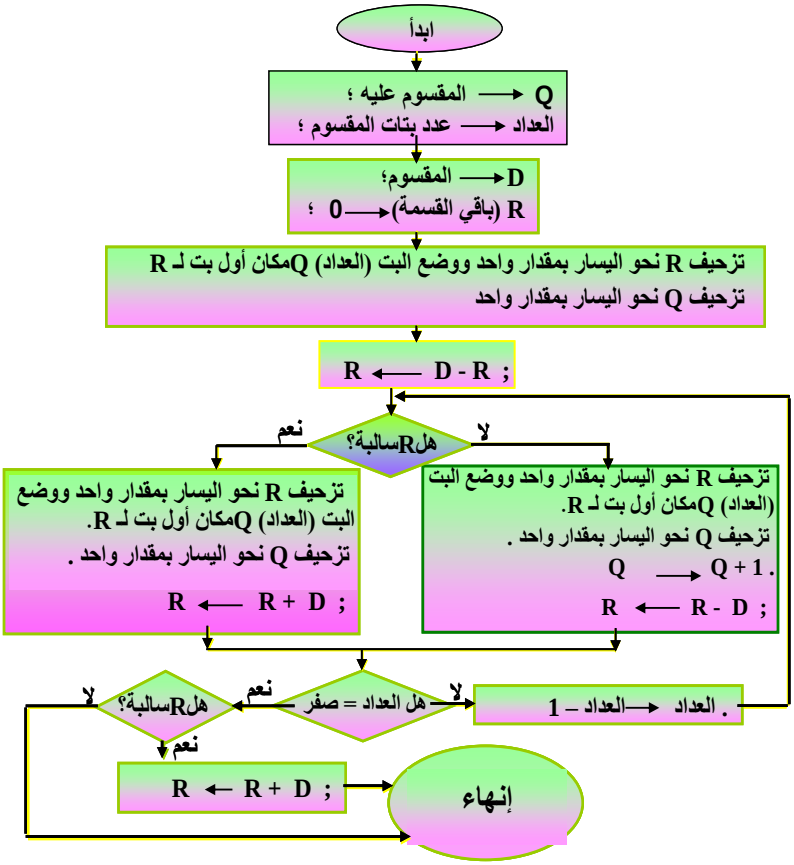
تم تنفيذ كل من عمليتي الجيب والجيب تمام بأسلوب خط الأنابيب باستخدام خوارزمية كوردك، حيث تم استخدام لغة VHDL وهي أداة للتصميم، ولوحظ أن IP Core يسرع من الكيان المادي (Hardware) لذلك تم بناء الدوال الأخرى (جيب وجيب تمام ومعكوس ظل القطع الزائد ومعكوس الظل) باستخدام IP Core generation المدعوم من قِبَل شركة Xilinx، حيث تمت عملية استدعاء هذه الدوال المصممة إلى البرنامج الرئيس باستخدام إيعاز Port Map. إن السبب الرئيس في تنفيذ دالتي الجيب والجيب تمام بأسلوب خط الأنابيب هو كونهما يحتاجان إلى مدة زمنية طويلة لإنهاء التنفيذ، إذ تحتاج كلّ من عمليتي الجيب والجيب تمام إلى 17 نبضة لإنهاء عملية التنفيذ والحصول على أول الإخراج ثم بعد كل نبضة هناك إخراج لكل منهما عندما تكون قيم إدخالات الدالتين متسلسلة ومستمرة، وهذا هو السبب الرئيس في بناء وتنفيذ وحدة التنفيذ بحيث تعمل بشكل متوازٍ. يتم استدعاء دالة الجيب والجيب تمام في حالتين هما:

الأولى: عند القيام بتنفيذ دالتي الجيب والجيب تمام في الحالة الاعتيادية.
الثانية: عند القيام بتنفيذ دالة الظل، حيث تُنَفَذ دالة الظل عن طريق قسمة دالة الجيب على دالة الجيب تمام.

2.4.2 عملية القسمة ودالتا الظل والظل تمام :

عند استخدام IP Core Generation في توليد وتنفيذ عملية القسمة بأسلوب خط الأنابيب على لوحة Spartan3E، لوحظ أنها تحتاج إلى مساحة كبيرة ومدة زمنية طويلة للحصول على أول إخراج لعملية القسمة مع إظهار باقي القسمة، فعند القيام بقسمة 32 بتاً على 16 بتاً، لوحظ أنه بعد 32 نبضة يعطي أول قيمة من ناتج القسمة ثم بعد كل نبضة يعطي قيمة وعند القيام بعملية تركيبها لوحظ أنها تحتاج إلى ما يقارب 38% من حجم رقاقة Spartan3E، وبعد عدة محاولات تم تنفيذ خوارزمية القسمة مع عدم إعادة الخزن، حيث تم تركيبها وتنفيذها على رقاقة FPGA في لوحة Spartan3E والحصول على نتائج جيدة (تحتاج هذه العملية إلى ما يقارب 4% من حجم الرقاقة وبعد كل نبضة تعطي قيمة لناتج عملية القسمة) وهذه الخوارزمية هي خوارزمية التزحيف مع إجراء بعض الإضافات عليها لزيادة سرعة التنفيذ، فضلاً عن إمكانية الحصول على باقي القسمة منها.

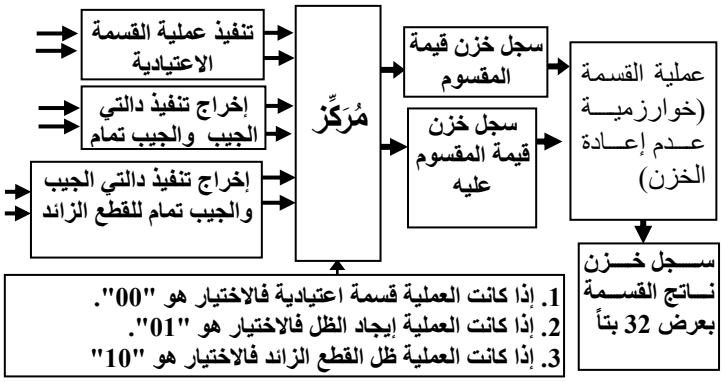
إن الأسلوب المستخدم في تنفيذ هذه الخوارزمية هو استخدام المتغيرات (Variables) ضمن العملية المستخدمة في التركيب، إذ أُجريت جميع العمليات عند حافة النبضة والحصول على الناتج بعدها بنبضة. إن خوارزمية القسمة عدم إعادة الخزن المنفذة، الموضحة في الشكل (8) [12] تم تنفيذها بحيث تقوم بقسمة 32 بتاً على 16 بتاً، إن الغرض من قسمة 32 بتاً هو لزيادة حدود ناتج القسمة، ففي بعض الأحيان نحتاج إلى قسمة عدد معين على عدد صغير نسبياً ومن ثم فإن ناتج القسمة سيكون كبيراً. والشكل (9) يوضح كيفية تنفيذ عملية القسمة.



الشكل (8) خوارزمية القسمة مع عدم إعادة الخزن

إن دالتي الظل وظل القطع الزائد تم تنفيذهما عن طريق تنفيذ دالتي الجيب والجيب تمام ثم قسمة ناتج الجيب على الجيب تمام للحصول على ناتج عملية الظل (أي أن هناك ثلاث عمليات تحتاج إلى استخدام خوارزمية القسمة وهي عملية القسمة الاعتيادية (إما قسمة البيانات المخزونة في سجلين من سجلات المعالج على البيانات القادمة من الذاكرة الرئيسة أو قسمة البيانات المخزونة في سجلين من سجلات المعالج على البيانات المخزونة في سجل آخر من سجلات المعالج) وعملية قسمة الجيب على الجيب تمام للحصول على الظل وعملية قسمة جيب على جيب تمام القطع الزائد للحصول على ظل القطع الزائد) كما موضح في الشكل (9). في حالة تنفيذ عملية القسمة الاعتيادية هناك احتمالان، إما أن يكون المقسوم عليه هو البيانات القادمة من الذاكرة (0-15) والمقسوم هو البيانات المخزونة في سجلين من سجلات المعالج ذات العنوانين (21-

24) و (25-28)، أو أن يكون المقسوم عليه هو البيانات المخزونة في أحد السجلات ذي العنوان (0-3) والمقسوم هو البيانات المخزونة في سجلين من سجلات المعالج ذي العنوانين (21-24) و(25-28).



الشكل (9) تحديد عملية القسمة

3.4.2 الجيب والجيب تمام للقطع الزائد والجذر التربيعي ومعكوسي الظل وظل القطع الزائد:

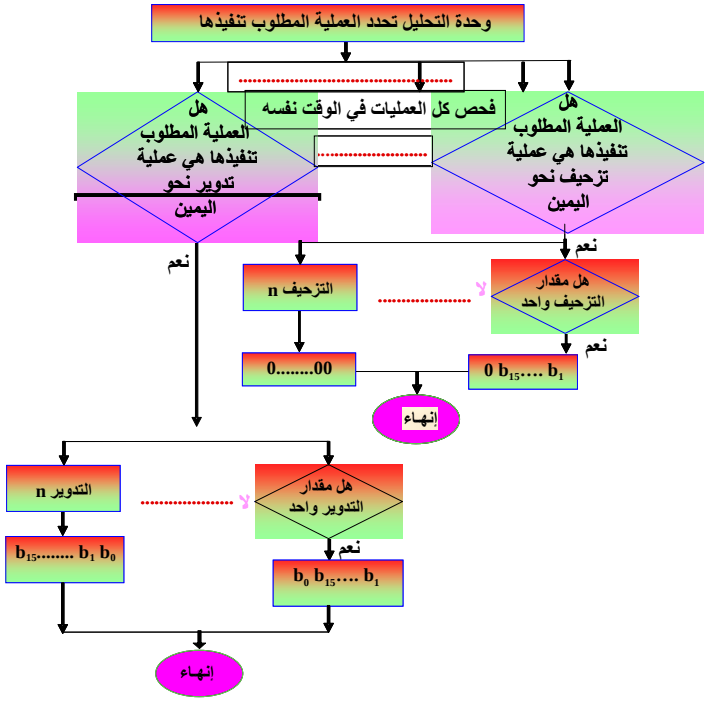
تم تنفيذ كل من هذه الدوال بأسلوب خط الأنابيب باستخدام خوارزمية كوردك باستخدام IP Core Generation [13] ثم القيام باستدعائها في البرنامج الرئيس باستخدام إيعاز Port Map المدعوم من قِبَل شركة Xilinx. إن تنفيذ هذه الدوال بأسلوب خط الأنابيب يزيد من الأداء والسرعة والشكل (10) يوضح تنفيذ دالة معكوس الظل.

لا يستخدم البت 30 مباشرةً لخزن ناتج التنفيذ لأنه قد يكون هناك إيعازات أخرى يتم تنفيذها لاتحتاج إلى خزن ناتج تنفيذها وبالتالي فإن قيمة البت 30 ستتغير، لذلك تخزن قيمة البت 30 في متغير خاص لكل دالة، وقد تم وضع إشارة (Ready) على إخراج كل دالة، إذ تُفَعّل هذه الإشارة عندما يتم تنفيذ احدى هذه الدوال والحصول على الاخراج، حيث يتم عمل ضرب منطقي بين إشارة Ready والمتغير الخاص بالدالة المعنية، فإذا كان الناتج واحداً فيخزن ناتج التنفيذ في سجل خاص لكل دالة من هذه الدوال.

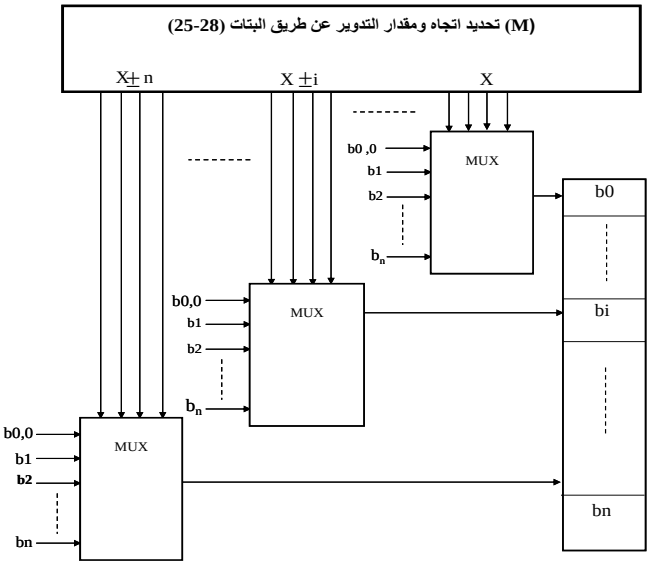
الشكل (10) عملية معكوس الظل المنفذ بأسلوب خط الأنابيب

4.4.2 عمليتا التدوير والتزحيف :

تم تنفيذ عمليتي التزحيف والتدوير بشكلٍ متوازٍ بحيث تحتاج كل منهما إلى نبضة واحدة فقط لتزحيف أو تدوير السجل بمقدار عدد بتاتها، والشكلان (11) و (12) يوضحان عمليتي التزحيف والتدوير نحو اليمين وهو عكس عمليتي التزحيف والتدوير نحو اليسار.



الشكل (11) المخطط الانسيابي لتنفيذ عمليتي التزحيف والتدوير نحو اليمين



الشكل (12) بناء عمليتي التزحيف والتدوير

5.4.2 أعلام (Flags) :

تم وضع خمسة أعلام، الأول يُفعّل عندما يكون هناك محمول (carry) نتيجة لإجراء عملية معينة والثاني يُفَعّل عندما يكون ناتج عملية معينة سالب والثالث يُفعّل عندما يكون المقسوم عليه في عملية القسمة صفر والرابع يُفعّل عندما يكون هناك طفحان نتيجة إجراء عملية معينة والأخير يُفعّل عندما يكون ناتج أي عملية صفراً.

3. النتائج :

تتضمن هذه النتائج، عرض نتائج المحاكاة لبعض العمليات التي يقوم المعالج بتنفيذها باستخدام الأداة Xilinx ISE Simulator (Test Bench)، فضلاً عن استخدام لوحة Spartan3E لغرض تنفيذ المعالج وعرض نتائج تنفيذ العمليات على شاشة الحاسوب.

1.3 كميات الموارد المستخدمة في تنفيذ المعالج :

بلغت النسبة المئوية لعدد الشرائح المستخدمة في تركيب المعالج نسبة إلى عدد الشرائح الكلية 67%، أما الدائرة المسوقة للـ VGA فتحتاج إلى 45% من عدد الشرائح الكلية لعرض جميع إخراجات المعالج على شاشة الحاسوب، إن عملية ربط المعالج مع VGA تحتاج إلى ما يقارب 117% من عدد الشرائح الكلية (67% + 45% + 5% (الخمسة بالمئة لإيصال إخراجات المعالج بمنفذ VGA))، ولغرض ربط المعالج مع VGA وعرض جميع نتائج وحدة التنفيذ على شاشة الحاسوب تم إلغاء البتين 29 و30 من الهيئة العامة للإيعاز ليصبح عدد الشرائح الكلية 98% (ان السبب الرئيس لالغاء البتين هو كون ان عملية ربط المعالج بمنفذ VGA يحتاج الى اكثر من عدد شرائح رقاقة FPGA بمقدار 17% اي نحتاج الى 117% من اصل 100% وهذا غير ممكن لذلك تم الغاء تفعيل البتين اثناء التسقيط فقط). والجدول (1) يوضح كمية الموارد المستخدمة في تنفيذ المعالج على رقاقة FPGA بدون VGA. بينما الجدول (2) يوضح عدد الموارد الكلية المستخدمة في تنفيذ المعالج مع VGA بعد إلغاء البتين 29 و 30 كما يوضح الانحدار الكبير في قيمة أعلى تردد يعمل به المعالج، والسبب الرئيس في ذلك (في انحدار أعلى تردد) هو كون وحدة التنفيذ صُمّمت لتُنفّذ جميع عملياتها بشكل متوازٍ إذ بلغت عدد الإخراجات الكلية للمعالج 533 إخراجاً بينما يقوم منفذ VGA بعرض إخراجات المعالج بشكل متسلسل.

الجدول (1) الموارد المستخدمة في الكيان المادي لتنفيذ المعالج بدون VGA.

Type Resources	Utilized Resources	Total Resources	Ratio
N. of Slices	3010	4656	%64
N. F.Flops	5354	9312	%57
N. of 4 i/p LUTs	5408	9312	%58
N. of IOBs	2	232	%0
N. of Block RAMS	3	20	15%
N. of GCLKs	2	24	8%
Op. Freq.	133.820MHz		

الجدول (2) الموارد المستخدمة في الكيان المادي لتنفيذ المعالج مع VGA.

Type Resources	Utilized Resources	Total Resources	Ratio
N. of Slices	4587	4656	%98
N. F.Flops	5955	9312	%63
N. of 4 i/p LUTs	8345	9312	%89
N. of IOBs	16	232	%6
N. of Block RAMS	4	20	%20
N. of GCLKs	3	24	%12
Op. Freq.	11.301MHz		

عند القيام بتركيب المعالج، لوحظ أن ذاكرتي جلب وتحليل الإيعازات بنيتا بشكل ذاكرتين كتليتين، علماً أن كلاً منهما قد تم تصميمها باستخدام لغة وصف الكيان المادي VHDL، ويوضح الجدول (2) عدد الذاكرات الكتلية المستخدمة في تركيب المعالج وهي أربعة (الأولى : ذاكرة ثنائية المنفذ غير متساوية الأطوال تمثل الذاكرة الرئيسة والأخيرة : ذاكرة ثنائية المنفذ استخدمت لعرض نتائج التنفيذ على شاشة الحاسوب، أما الذاكرتان الثانية والثالثة فهما ذاكرتا جلب وتحليل الإيعازات اللتان صممتا باستخدام لغة وصف الكيان المادي).

2.3 حساب زمن عدد النبضات التي تحتاجها كل عملية :

إن كل سجل من سجلي جلب وتحليل الإيعازات وكلاً من ذاكرتهما تحتاج إلى نصف نبضة لقراءة أو كتابة البيانات والإيعازات منه أو عليه على التوالي، بينما تحتاج عملية قراءة الإيعازات والبيانات من الذاكرة الرئيسة إلى نصف نبضة (أي لوصول البيانات والإيعازات القادمة من الذاكرة الرئيسة إلى وحدة التنفيذ نحتاج إلى نبضتين ونصف). ويوضح الجدول (3) عدد النبضات التي تستغرقها كل عملية في الحالة الابتدائية لإنهاء تنفيذها، وتحتاج كل عملية إلى نبضة واحدة لإنهاء التنفيذ والحصول على الإخراج بعد الحصول على أول إخراج لها، عندما تكون قيم إدخالات الدوال متسلسلة ومستمرة.

الجدول (3) عدد النبضات التي تحتاجها كل عملية لإنهاء التنفيذ في الحالة الابتدائية حيث M :هي عدد النبضات

العمليات	عدد النبضات لتنفيذ N من المهام
1. عمليات الحساب والمنطق وعمليات التدوير والتزحيف وعمليات النقل.	M = N
2. عمليات الجيب والجيب تمام وجيب وجيب تمام القطع الزائد و معكوس الظل ومعكوس ظل القطع الزائد والجذر التربيعي	M = N + 17
3. عمليتي الظل وظل القطع الزائد	M = N + 23

- **محتويات سجلات المعالج في الحالة الابتدائية :**

لقد تم إدخال البيانات في سجلات المعالج والموضحة في الجدول (4) لتنفيذ جميع عمليات المعالج.

الجدول (4) القيم الابتدائية المخزونة في ملف السجلات

7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**السجل**
FF01	**0AA0**	**9999**	**A016**	**FFF1**	**011F**	**C893**	**5CC5**	**محتوياته**
F	**E**	**D**	**C**	**B**	**A**	**9**	**8**	**السجل**
376C	**D2BE**	**2D41**	**0000**	**E000**	**376C**	**00FA**	**1FFF**	**محتوياته**

3.3 عرض نتائج المحاكاة لبعض العمليات :

فيما يأتي عرض بعض نتائج المحاكاة للمعالج الرياضي المصمم بأسلوب خط الأنابيب :

*** مثال لحساب المعادلة الآتية :** $(\sqrt{0.25}+5)(5-SIN\frac{\pi}{3})$

تمت عملية حساب المعادلة أعلاه وحساب عدد النبضات التي تحتاجها لإنهاء التنفيذ إذ تم تحويل وإيجاد ناتج كل عملية في المعادلة أعلاه وكما يلي:

$(0.25)_D = (2000)_H$, $(\frac{\pi}{3})_D = (2182)_H$

$(0.5)_D = (4000)_H$

$(5)_D=(0005)_H$ ➔ 00310005➔ MOV R1,0005

$(SIN\frac{\pi}{3})_D=(376D)_H$ ➔ 40452182➔SIN R2,2182

$(5-SIN\frac{\pi}{3})_D=(C898)_H$ ➔ 64220000➔SUB R1,R2

$(\sqrt{0.25}=0.5)$ ➔ 40692000➔SR R3,2000

$(\sqrt{0.25}+5)_D=(4005)_H$ → 62610000→ADD R3,R1

$(4005)_H*(C898)_H=(3229EAF8)_H$ → 22630000→MUL R3,R1

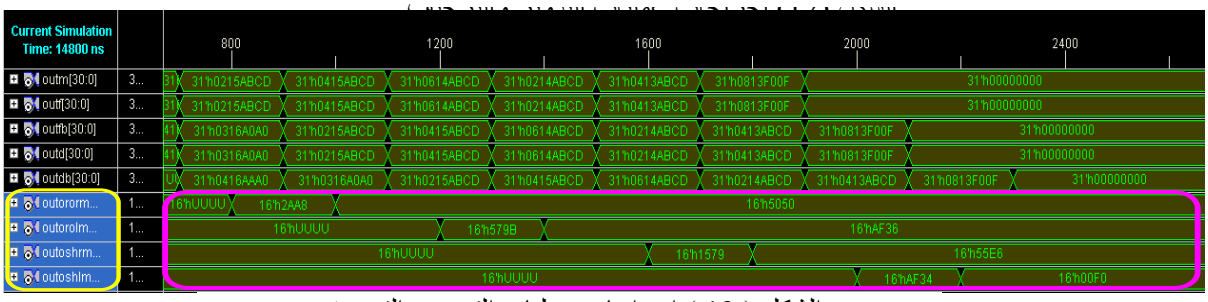
بلغت عدد النبضات اللازمة لإنهاء التنفيذ والحصول على الإخراج النهائي للمعادلة أعلاه 23 نبضة حيث كان الناتج النهائي للحصول على القيمة النهائية للمعادلة هو (3229EAF8)

1.3.3 محاكاة عمليتي التزحيف والتدوير :

لقد تم إدخال البيانات الآتية في الذاكرة الرئيسة والموضحة في الجدول (5) والشكل (13) يوضح نتائج تنفيذ عمليتي التزحيف والتدوير:

الجدول (5) نتائج حساب عمليتي التدوير والتزحيف

العملية	الادخال	مقدار التزحيف او التدوير	الناتج
SHL	F00F	4	00F0
SHR	ABCD	1	55E6
ROL	ABCD	2	AF36
ROR	A0A0	1	5050



الشكل (13) إخراجات عمليات التدوير والتزحيف

2.3.3 محاكاة عمليات الحسب :

لقد تم إدخال البيانات الآتية في الذاكرة الرئيسة : ← **اتجاه الخزن في الذاكرة الرئيسة**

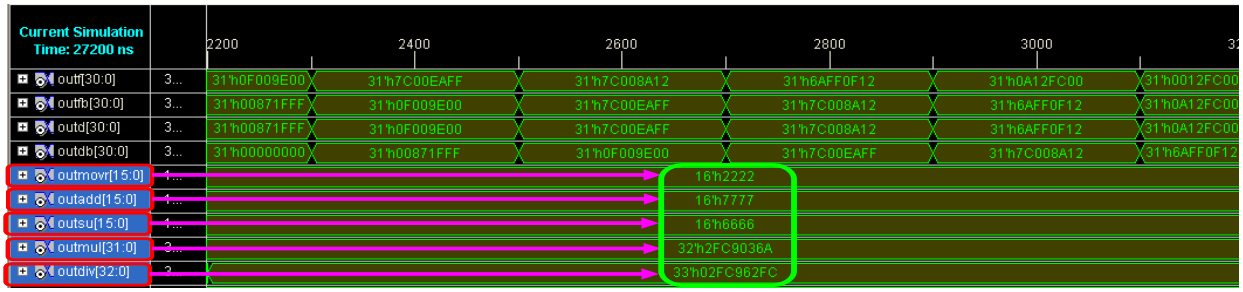
← 00712222 (MOV R3,2222) | 4422aaaa (SUB aaaa,R1,R2) | 00314444 (MOV R1,4444)

0484fffe (DIV R2,R4,FFFE) | 68430000 (MUL R2,R4,R2,R4) | 48615555 (ADD 5555,R3,R4) ←

البيانات والإيعازات أعلاه استخدمت لحساب المعادلة أدناه والشكل (14) يوضح ناتج محاكاة المعادلة أعلاه.

ناتج القسمة/باقي القسمة

$$\frac{(aaaa-4444)*(2222+5555)}{fffe}=2\,fc9\,|\,62\,fc$$



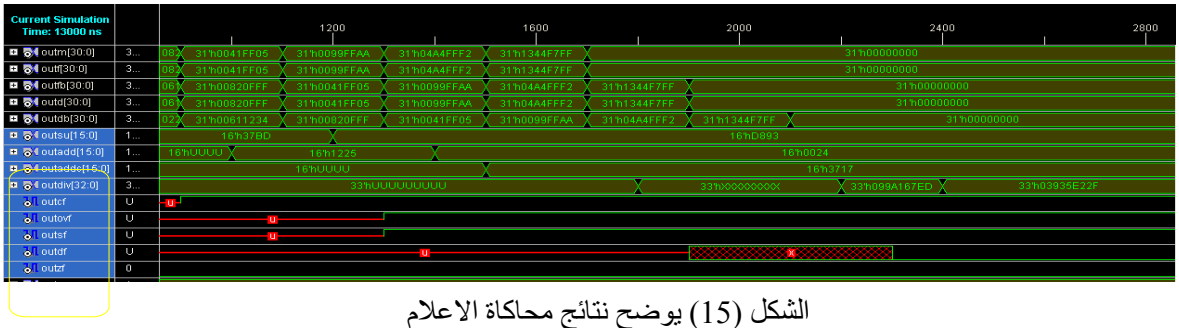
الشكل (14) ناتج المحاكاة لعمليات الجمع والطرح والضرب والقسمة

3.3.3 محاكاة الاعلام :

لقد تم إدخال البيانات في الذاكرة الرئيسة كما موضح في الجدول (6) والشكل (15) يوضح نتائج تنفيذ عمليات الجمع والطرح والقسمة وأعلامها والجمع مع المحمول:

جدول (6) عملية الجمع والطرح والقسمة

العملية	الادخال 1	الادخال 2	الناتج
ADD	1234	FFF1	1225
SUB	0050	C893	37BD
DIV	9999011F	FFF2	99A167ED
ADDC	376C	FFAA	3717



الشكل (15) يوضح نتائج محاكاة الاعلام

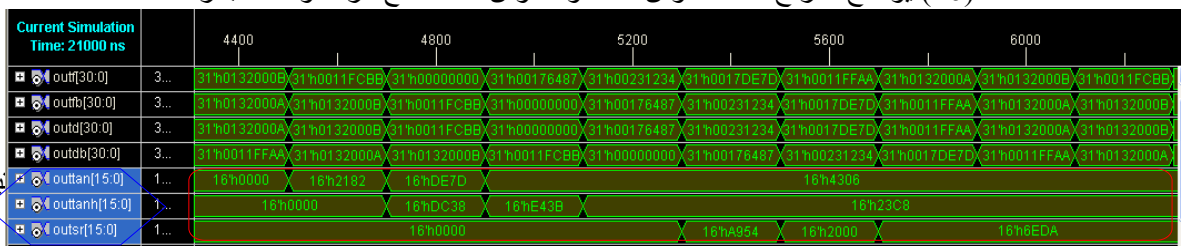
4.3.3 محاكاة دالة معكوس الظل ومعكوس ظل القطع الزائد ودالة الجذر التربيعي:

تم إدخال البيانات المدرجة في الجدول أدناه لفحص دالة معكوس الظل ودالة معكوس ظل القطع الزائد ودالة الجذر التربيعي، اذ يوضح الجدول (7) نتائج حساب كل دالة من هذه الدوال، بينما الشكل (16) يوضح نتائج محاكاة كلٍّ منها.

الجدول (7) نتائج حساب دالة معكوس الظل ومعكوس ظل القطع الزائد ودالة الجذر

القيم المخزونة في الذاكرة	جيب القطع الزائد	جيب تمام القطع الزائد	الزاوية
00871fff	376C	1fff	2182
00281fff	C893	1fff	DC38
0009E000	E000		A954

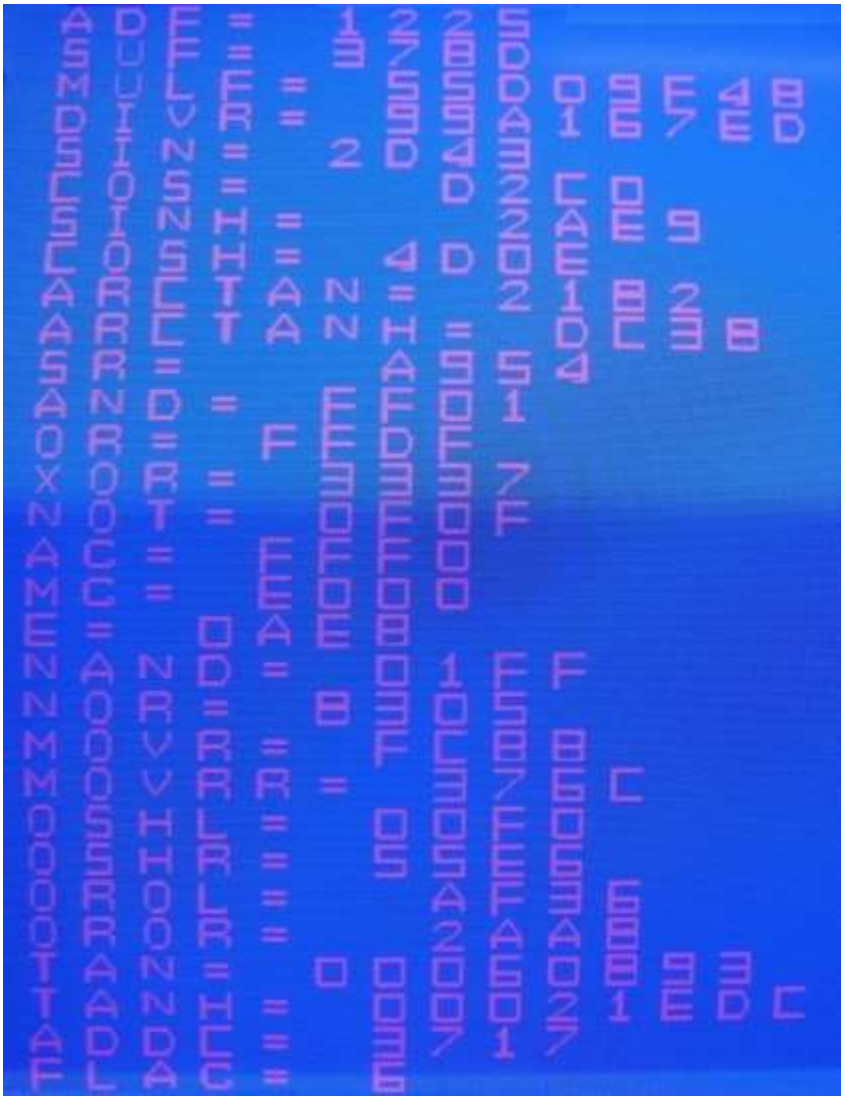
الشكل (16) يوضح اخراج دالة معكوس الظل ومعكوس ظل القطع الزائد ودالة الجذر



4.3 عرض النتائج على شاشة الحاسوب :

تم تحميل المعالج على رقاقة FPGA في لوحة Spartan3E بعد إلغاء تفعيل البتين 29 و 30 وعرض جميع نتائج التنفيذ على شاشة الحاسوب عن طريق استخدام منفذ VGA، إذ تم تحميل البيانات والإيعازات على الذاكرة الرئيسة للمعالج عن طريق Jtag، ويوضح الشكل (17) نتائج تنفيذ عينة لكل من العمليات الثلاثين.

من الشكل (17) نلاحظ ان عملية محاكاة المعالج بكل عملياته مطابقة لعملية التنفيذ على شاشة الحاسوب فمثلا عملية الجمع مع المحمول (ADF)= 1225 و عملية الطرح (SUF)= 37BD وعملية القسمة (DIVF)= 99A167ED كما موضح في الشكل ادناه هو مطابق لنتائج المحاكاه الموجودة في الجدول (6) والشكل (15) وكذلك الحال بالنسبة لباقي العمليات فكل عملية من عمليات المعالج قمنا بمحاكاتها حصلنا على نفس الناتج اثناء القيام بتنفيذ المعالج على رقاقة FPGA.



الشكل (17) نتائج تنفيذ عينة لكل من العمليات الثلاثين على شاشة

5. الاستنتاجات : لقد تم من خلال هذا البحث استنتاج ما يأتي:-

1. إن التصميم بأسلوب خط الأنابيب يزيد من مقدار العطاء (Throughput) للتصميم، أي يزيد من سرعة وحدة المعالجة. و قد بلغ اكبر عطاء او انتاجية للرقاقة المصممة 133,820 MFlops، و بذلك تعتبر من اعلى الرقائق انتاجية نسبة الى كمية الدوائر المنطقية المستخدمة و المحددة برقاقة FPGA Spartan 3E

2. إن بناء وحدتي التحليل والتنفيذ بشكل متوازٍ عن طريق استخدام عدد من العمليات (processes) يقلل من احتمالية انتظار البيانات والإيعازات في الذاكرات المؤقتة لسجلي جلب وتحليل الإيعازات، أي إن كل إيعاز يصل إلى جميع العمليات الموجودة في وحدة التنفيذ في آن واحد فضلاً عن كون كل عملية من هذه العمليات تعمل بشكل مستقل عن الأخريات، إلا أن عملية التوازي هذه تقلل من مقدار أعلى تردد يعمل به التصميم (Maximum Operating Freq).

3. عند القيام بتركيب المعالج باستخدام إحدى أدوات البرنامج ISE، لوحظ أن ذاكرتي جلب وتحليل الإيعازات كُوِّنت بشكل ذاكرة كتلية (Block RAM). إن الأسلوب المستخدم في تنفيذ كل من هاتين الذاكرتين، يتم بعمل مصفوفة من نوع ذاكرة وصول عشوائي (RAM) ضمن العملية (Process) المستخدمة مع التركيب واستخدام حافتي النبضة الصاعدة والنازلة المولّدتين لكتابة وقراءة البيانات فيها ومنها على التوالي.

4. ان عملية استخدام المتغير (Signal) داخل شرط الحافة لبناء السجلات تحتاج إلى عدد أقل من الشرائح فضلاً عن كونها سريعة جدا في تكوين السجلات، فبمجرد استدعائه (داخل شرط الحافة) تُكَوَّن سجلاً إلا أن عملية تتبُّع البرنامج - عندما يكون البرنامج طويلاً جداً - صعبٌ عند حدوث مشكلة فيه (أي صعوبة كشف الأخطاء) إذا ما قورنت بعملية استخدام إيعاز PORT MAP في تكوين السجلات عن طريق عمل برنامج خاص لبناء سجل واحد (وهذا البرنامج هو فرع من فروع البرنامج الرئيس) ثم استدعاء هذا السجل بأسماء متغيرات جديدة في البرنامج الرئيس لتكوين سجلات بعدد الاستدعاءات.

5. عملية تحديث البيانات والإيعازات عن طريق عمل إعادة تهيئة جزئية أو كلية للذاكرة الرئيسة عن طريق المنفذ JTAG، تجعل المعالج المصمم بأسلوب خط الأنابيب معالجاً حقيقياً (أي تُمكِّن المستخدم من القيام بتنفيذ البيانات والإيعازات في أي وقت وبشكل آني).

6. خوارزمية القسمة - عدم إعادة الخزن - المُنفذّة في هذا البحث تحتاج إلى عدد قليل من الشرائح (Slices) فضلاً عن كونها تعطي إخراجاً بعد كل نبضة بينما تحتاج عملية القسمة المنفذة باستخدام مولد اللب (الصميم) المنطقي IP Core Generation إلى ما يقارب 40% من عدد الشرائح الكلية للوحة Spartan3E، إلا ان هذا المولد يزيد من مقدار أعلى تردد تعمل به عملية القسمة.

References :

[1] Joanna Armeni, "Glossary Terms Basic Concepts", http://schoolnet.gov.mt/ictsec/Resources/SharingZone/ResNo033.doc.

[2] Santa Clara, Calif.,"60 years of continued transistor shrinkage, innovation Intel 45 Nanometer Technology Gives the Transistor Dramatic Make-Over", www.intel.com/pressroom, Jan. 29, 2007.

[3] John Bruno, John W. Jones, Kimming SO,"Deterministic Scheduling with Pipelined Processors", IEEE Transactions On Computers, Vol. c-29, NO. 4, April 1980.

[4] Basil Shukr Mahmood. " Design of a pipeline Graphic Processor", Raf.jour.Sci.,Vol.7,No.1,pp.50-68 , 1996.

[5] Charles Brej."A MIPS R3000 microprocessor on an FPGA", 13 February 2002. www.citeulike.org/user/rgrubisic/article/2227725.

[6] Jonathan Barre, C. Landet, Christine Rochange, Pascal Sainrat, "Modeling Instruction-Level Parallelism for WCET Evaluation", Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06), 0-7695-2676-4/06. 2006 IEEE.

[7] Ehsan Abdul-Sattar Ali, "Design and Testing A Reconfigurable Calculation Unit using FPGA" Master thesis, Computer Department, College of Engineering, Mosul University, 2008.

[8] Jari Nurmi, "Processor Design : System-on-Chip Computing for ASICs and FPGAs", Finland Tampere University of Technology, Published by Springer, P.O. Box 17, 3300 AA Dordrecht, The Netherlands. Copyright © 2007 Springer , www.Springer.com.
[9] Kris Gaj, "VHDL Coding for Synthesis", ECE 448 – FPGA and ASIC Design with VHDL, George Mason University, Spring 2009.
[10] "Spartan-3E FPGA Family : Complete Data Sheet", Copyright © 2005-2008 Xilinx, Inc. URL : www.xilinx.com/spartan3e. DS312 April 18, 2008.
[11] Petru Eles, Krzysztof Kuchcinski, Zebo Peng, Marius Minea, "Two Methods for Synthesizing VHDL Concurrent Processes", The Swedish National Board for Technical Development (NUTEK), and TEMPUS JEP-2754, Linköping University, 1993. ftp://ftp.ida.liu.se/pub/publications/techrep/1993/r-93-22.ps.gz.
[12] Sherif Galal and Dung Pham, "Division Algorithms and Hardware Implementations", EE 213A: Advanced DSP Circuit Design, The University of California, August 2007. ww.ee.ucla.edu/ingrid/ee213a/lectures/division_presentV2.pdf.
[13] "CORDIC v3.0," DS249 May 21, 2004 ©Xilinx,Inc. http://www.xilinx.com/legal.htm.

تم اجراء البحث في كلية ألهندسة = جامعة ألموصل