مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٨) العدد(١) ٢٠٢٤
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (8) No. (1) 2024

اختيار التركيب الأمثل للذواكر في النظم المضمنة من منظور الطاقة وزمن التنفيذ

م. رزان عقباتي*

(تاريخ الإيداع ٢٠٢٣/٩/١٧ . قُبِل للنشر في ٢٠٢٤/١/١٤)

□ ملخّص□

مع ازدياد الاهتمام بالنظم المضمنة، اتجه مصممو هذه النظم إلى دراسة طرق لتحسين استهلاك الطاقة وسرعة النظام، واتجهت البحوث باتجاه دراسة ذاكرة Scratchpad كبديل عن الذاكرة المخبأة في النظم التقليدية لما لها من ميزات.

يمكن أن تكون ذواكر Scratchpad بديلاً فعالاً للذاكرة المخبأة في التطبيقات المضمنة، ولكن بالمقابل كان لها عيوب خاصة بها. قد يكون تكوين الذاكرة الأكثر فاعلية في أي تطبيق بأن يجمع بين الذاكرة المخبأة وذاكرة scratchpadعلى شريحة واحدة، ولكن نفقات التصنيع الباهظة تحد من تطبيق هذه الذاكرة الكلية على الشريحة. تقترح هذه الورقة منهجية يمكن من خلالها للمصممين تحديد المزيج الأمثل من ذاكرة scratchpad والذاكرة المخبأة لتطبيق معين، والتأكد من استخدام scratchpad بفاعلية مع تحسين استهلاك طاقة النظام وسرعة التنفيذ مع مراعاة المساحة المتاحة على الشريحة .

الكلمات المفتاحية: الأنظمة المضمنة، الذاكرة المخبأة.

*عضو هيئة فنية في كلية هندسة تكنولوجيا المعلومات والاتصالات في جامعة طرطوس، قسم النظم الحاسوبية والالكترونية _ جامعة طرطوس_ سوريا.

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٨) العدد(١) ٢٠٢٤
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (8) No. (1) 2024

Choosing the optimal memory composition in embedded systems from the perspective of energy and execution time

Eng. Razan Akabati*

(Received 17/9/2023 . Accepted 14/1/2024)

□ ABSTRACT □

With the increasing importance of the use of embedded systems, scientific research has turned to these systems significantly, so that designers of this type of system continuously to improve performance and reduce energy.

Scratchpad memories can be an effective alternative to cache in embedded applications, but have drawbacks of their own. The most effective memory configuration for any given application may be one which combines cache and scratchpad on a single chip, however manufacturing expenses limit the amount of total on-chip memory for any SoC. Given an area budget for on-chip memory, this paper proposes a methodology by which a designer can determine the optimal mix of scratchpad and cache for an application, and ensure that the scratchpad is used effectively with improvement in system power consumption and execution speed.

Keywords: Embedded systems, SPM memory ,cache memory.

*Engineer in the Department of Computer and Electronic Systems Engineering _ Faculty of Information and Communications Technology Engineering_ Tartous University.

١- المقدمة:

يمكن للأنظمة المضمنة الاستفادة من عمليات التكيف في وقت التصميم بما يتناسب مع طبيعة التطبيق، على سبيل المثال: يمكن تحسين البرنامج (العتاد اللين) بالاعتماد على المعرفة الكاملة لمنصة الأجهزة (العتاد الصلب)، ويمكن تحسين الأجهزة لتناسب متطلبات التطبيق.

أحد أهم هذه التحسينات المتاحة لمصممي النظام هو اختيار التسلسل الهرمي. تؤثر القرارات التي تتخذ أثناء تصميم الأجهزة والبرمجيات بشكل كبير على أداء الذاكرة ، التي تعد مساهماً رئيسياً في إجمالي استهلاك الطاقة وسرعة تنفيذ النظام.

من ناحية العتاد الصلب(الهاردوير) ، يجب على المصمم توفير ذاكرة على شريحة ، ولكن يجب أن يختار شكلها. تكون ذواكر Scratchpad مفيدة للغاية عندما تكون التأثيرات غير المحددة لذاكرة المخبأة غير مرغوب فيها (كما في النظم الزمن الحقيقي). ومع ذلك، في النظم المضمنة التي لا تخضع لقيود الزمن الحقيقي الصعبة، الذواكر المخبأة لا تزال هي القاعدة المألوفة. إضافة إلى ذلك، من منظور الطاقة وسرعة الأداء، من الممكن تحسين الذاكرة المخبأة فقط دون الحاجة إلى ذاكرة SPM.

من منظور آخر، يمكن أن تكون ذواكر SPM البديل الرخيص الثمن وقليل الطاقة في الأنظمة التي تكون فيها خصائص وقت التشغيل معروفة للمصمم.

عند تصميم نظام على الشريحة System-on-Chip (SoC)، يكون الحل الأكثر شيوعاً هو استخدام تسلسل هرمي للذاكرة يحتوي على كل من scratchpad والذاكرة المخبأة. إنه الحل المثالي في تصميم النظام المضمّن الحديث لتصميم الذاكرة على الشريحة (scratchpad والمخبأة) لتكون مصممة حسب تكلفة التصنيع.

ومع ذلك، يجب أن يكون مبررا إدراج scratchpad في نظام الذاكرة مع الإجابة على السؤال عن أفضل السبل للاستفادة من إدخال هذه الذاكرة: أي ماهي الأشياء التي سيتم تخصيصها إلى scratchpad؟ في هذه الورقة، نقترح منهجية لتحديد أفضل طريقة لتوضع الذاكرة على شريحة في وقت التصميم . يتم إعطاء المعايير من حيث أعلى نسبة كفاءة ذاكرة scratchpad إلى الذاكرة المخبأة.

٢- الدراسات المرجعية

الملاحظة الرئيسية التي تقود أبحاثنا هي أن هناك حاجة لتخديم الكائنات الأكثر استخداماً بشكل سريع في الذاكرة، لكن الذاكرة المخبأة البسيطة ستحاول دائماً تخديم كل الكائنات بذاكرة سريعة . تنشأ حالات عدم كفاءة في نموذج الذاكرة المخبأة عندما تحتاج الذاكرة المخبأة لجلب العناصر القصيرة العمر من الذاكرة الرئيسية، لتحل محل الشفرة أو البيانات التي يتم الوصول إليها بشكل.

من ناحية أخرى، فإن ذاكرة SPM غير فعّالة عندما تكون غير قادرة على تخديم عدد كبير من الوصولات إلى الذاكرة، بحيث تتطلب من المعالج أن يقوم بعدد كبير من الوصول إلى الشريحة.

يمثل النظام المكون من كل من scratchpad والذاكرة المخبأة معاً حلاً ممكناً؛ يجب أن تحافظ scratchpad والذاكرة المخبأة على الغالبية العظمى من الكائنات التي تدخل بشكل متكرر إلى الشريحة وذلك لأكبر قدر ممكن من وقت تنفيذ التطبيق.

وبالتالي الغرض من خوارزمية تعبئة scratchpad هو تحديد تلك العناصر التي تستخدم ذاكرة scratchpad، وعند إزالة هذه العناصر من ذاكرة المخبأة ووضعها في ذاكرة SPM، ستخلق تعاون أكثر بين الذاكرتين . ويجب استخدام العناصر التي تم اختيارها للتخديم بواسطة ذاكرة scratchpad مع تردد كافٍ لضمان تضمينها بشكل دائم في الذاكرة السريعة.

وقد تناولت العديد من الأبحاث مشكلة ماهي أفضل طريقة لتوظيف ذاكرة SPM في الأنظمة المضمنة .باندا وآخرون اقترحوا خوارزمية لتوضع البيانات في ذاكرة scratchpad في الأنظمة مع بيانات ذاكرة المخبأة [1]، بالإضافة إلى أداة لتحديد التوضع الأمثل للذاكرة المخباة و ذاكرة SPM [2] . كانديمير وآخرون استخدموا نهج قيادة المترجم، توظيف تحويلات الحلقة والبيانات لتحسين تدفق البيانات بشكل ديناميكي (العتاد اللين) إلى ذاكرة SPM المسيطرة في النظام [3]. يتضمن نموذج الذاكرة هذا ذاكرة مخبأة، لكنهم أكثر اهتماماً بإدارة عمليات نقل البيانات بين الذاكرة خارج الشريحة و [scratchpad] على الشريحة، وتجاهل الذاكرة المخبأة .بينيني وآخرون اتخذوا نهج أكثر توجها نحو الأجهزة (العتاد الصلب)، وإظهار كيفية توليد ذواكر scratchpad وفك ترميز العناوين المخصصة من خصائص التطبيق [4].

لا يتعامل أي من الأعمال المذكورة أعلاه مع تعليمات الوصول للذواكر، والتي وجدنا أنها جزء مهم إن لم يكن مهيمناً من إجمالي تكلفة الذاكرة في المعالجات المضمنة.

تشن و كانديمير وآخرون قد درسوا إدارة التعليمات في scratchpads [5] ولكن تجاهلوا تواجد الذاكرة المخبأة التقليدية من النظام. في [6] ستيك وآخرون أيضاً تجاهلوا الذاكرة المخبأة عندما درسوا التوضع لكل من التعليمات والبيانات ضمن ذاكرةscratchpad . في [7] تقترح المجموعة نفسها حلاً يتضمن الذاكرة المخبأة، ولكنها تدرس التعليمات ولا تأخذ بعين الاعتبار البيانات .

تقدم هذه الورقة مساهمتان رئيسيتان لم يتم فحصهما من قبل .أولاً، نهتم بمقارنة السرعة والطاقة بين الذاكرة المخبأة و scratchpad في كمية ثابتة من مساحة السيليكون (أي تكلفة التصنيع) . نقدم منهجية تسمح للمصممين بتحديد أنسب تكوين ذاكرة لنظامهم .ثانياً، نقدم حلاً مثالياً للتعبئة الثابتة (توضع البيانات) لذاكرة scratchpad استناداً إلى الملاحظات أعلاه .لا تتطلب طرقنا إضافة أجهزة (عتاد صلب) مخصصة إضافية، كما هو الحال في بعض المخططات الديناميكية، ولا توجد تعديلات

على مترجم المصمم الذي تختاره.

الهدف الثانوي من بحثنا المستمر هو تحديد البنى التي تحتاجها التطبيقات المضمنة المتنوعة .نحن نسعى لتحديد العلاقات بين أعباء العمل وبنية الذاكرة، من أجل إنشاء مبادئ توجيهية أو أداة للمصممين لتحقيق الأنظمة الأكثر كفاءة الممكنة في حيث

السرعة والطاقة.

٣- طريقة العمل

تقدم أجهزة RISC المتقدمة جهاز محاكاة مجموعة التعليمات مع مجموعة تطوير ARM الخاصة بها . يسمح المحاكي المسمى[8] ARMulator للمطورين بتقييم برمجيات ARM على مجموعة متنوعة من المنصات الخاصة بالمحاكي.

لإنتاج البنية لعمليات المحاكاة الخاصة بنا، سنقوم بتعديل نموذج ARMulator الخاص بمعالج ARM 710a إلى معالج ARM7 الذي يتضمن وحدة إدارة ذاكرة (MMU) ، وذاكرة مخبأة موحدة للتعليمات والبيانات ، و٨مداخل و 4 عناوين للكتابة التي سنتركها دون تغيير .في المحاكاة التي تحتوي على ذاكرة مخبأة، تتميز الذاكرة المخبأة بالخصائص التالية:

- تستخدم خوارزمية التوضع المباشر (1-way) Direct-mapped.
- يتم تحديث البيانات بشكل آني أي تستخدم سياسة الكتابة المباشرة Write-through.
- كل خط يحوي أربع كلمات (16 bytes) per line Four words.

ويتم إهمال الخصائص الأخرى في ARM710a. نقوم بمحاكاة المعالج بسرعة 100 ميجاهرتز، مع ناقل ذاكرة 100 ميجاهيرتز من أجل البساطة، وبالتالي تجنب تداخلات وتعقيدات تبديل الساعة في ARM7 . لإدخال زمن الوصول إلى الذاكرة الخارجية ، نعتمد على خريطة الذاكرة المرفقة في ARMulator ؛ يتم إعطاء الذاكرة الخارجية وقت وصول 100ns. من أجل البساطة، يفترض أن تكون عمليات القراءة والكتابة التسلسلية وغير التسلسلية تحتاج لوقت الوصول نفسه . يحتاج الوصول إلى SRAM لكل من scratchpad والذاكرة المخبأة إلى دورة ساعة واحدة.

ثم يتم تمكين متتبع ARMulator من أجل تتبع الوصول إلى الذاكرة، ويتم استخدام طريقة مبعثر التحميل (scatter-loading) لفصل كل الكائنات .ثم ننفذ تطبيق بدون الذاكرة المخبأة أ scratchpad ، حيث أننا مهتمون بالحصول على إحصائيات الوصول

لكل كائن دون تدخل من مدير الذاكرة أو المصمم.

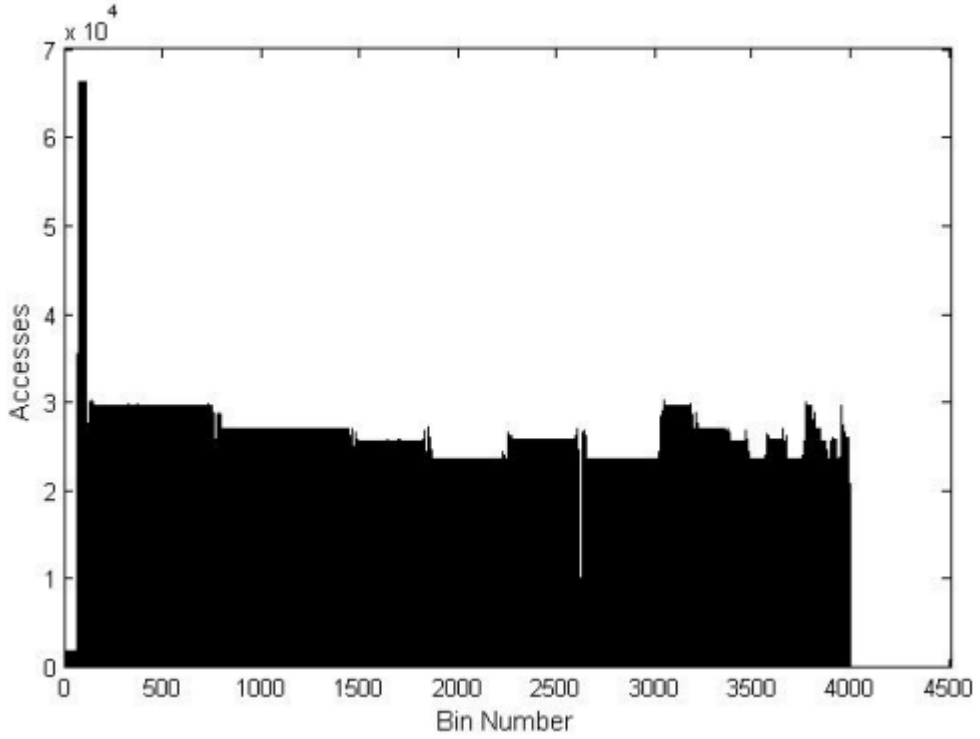
عند اكتمال التجريب نحصل على نتيجتين .الأولى هي اقتفاء أثر لجميع عمليات الوصول إلى الذاكرة للتنفيذ بأكمله. أما النتيجة الثانية بواسطة الرابط linker وهي عبارة عن خريطة لكافة كائنات التعليمات البرمجية والبيانات في الذاكرة .

يمكن استخدام أي معالج آخر، بما في ذلك مع هندسة معمارية لهارفارد والذواكر المخبأة منفصلة، طالما أن المصمم يمكنه الحصول على تتبع لأثر الدخول إلى الذاكرة . ويجب أن يسمح المحول البرمجي أيضاً بالتحميل المبعثر.

٤- المنهجية المقترحة

العناصر التي سيتم اختيارها لتتواجد في scratchpad يجب أن يتم استخدامها مع تردد كاف ليتم إدراجها بشكل دائم في الذاكرة السريعة على حساب كلفة التصنيع (مساحة السليكون) الحقيقية. ومع ذلك، ليس من الأفضل بالضرورة تحديد العناصر التي يتم الوصول إليها بشكل متكرر إلى ذاكرة SPM. سيؤدي القيام بذلك إلى ترك العناصر ذات التردد المنخفض في الذاكرة المخبأة، ويؤدي إلى مشاكل في عمليات الاستبدال وبالتالي الوقوع في أخطاء . بدلاً من ذلك، سوف نختار العناصر التي تخفف الذاكرة المخبأة، ولكنها مهمة في scratchpad. بشكل أبسط لن يكون استخدام تردد الوصول مفيداً، لأنه لا يفسر انتشار عمليات الوصول المحسوبة.

نقسم زمن التشغيل الكلي لمنصة الاختبار إلى أجزاء صغيرة موزعة بشكل متساوٍ. بعد ذلك، نقوم بتوزيع ملف التتبع الناتج من المرحلة السابقة من سطر إلى آخر، مع ربط كل وصول إلى الذاكرة برمز أو كائن بيانات من الخريطة التي تم انتاجها بواسطة الرابط .هذا يولد مخطط بياني مماثل للشكل (1) لكل كائن في منصة الاختبار.



الشكل (1) المخطط البياني لزمن لوصول لكائن

يستخدم هذا المدرج الإحصائي لحساب الانحراف المعياري لأزمنة الوصول لكل كائن . الانحراف المعياري هو مقياس مدى استخدام الكائن على مدار عمر البرنامج. كلما كان الانحراف المعياري للكائن أكبر، كلما كان الوصول إلى ذلك الكائن في فترة

التنفيذ الكلية أكثر.

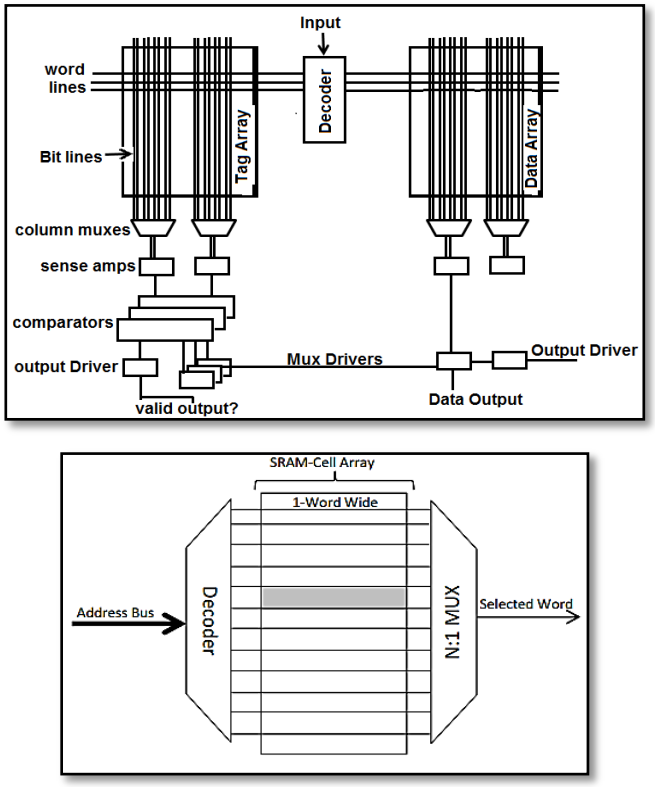
لقياس الانحراف المعياري وحساب تكلفة توضع الكائن في scratchpad، نقوم بتقسيم الانحراف المعياري الناتج وتردد الوصول حسب حجم هذا الكائن. المقياس الناتج هو مقياس للأولوية لإدراج هذا الكائن وتخديمه من قبل ذاكرة scratchpad. يتم ترتيب الكائنات حسب هذا المقياس ، بترتيب تنازلي ، للوصول إلى قائمة مرتبة من الكائنات لوضعها في ذاكرةscratchpad.

لتحديد عناصر لحزم scratchpad ، نبدأ في أعلى قائمة الأولويات ونضيف كل كائن إلى scratchpad إذا كان هناك مساحة كافية متبقية للقيام بذلك .نستمر بهذه الطريقة حتى يمتلئscratchpad، أو نصل إلى أسفل قائمة الأولويات.

٥- اختيار حجم ذكرة SPM

٥-١- المساحة:

تحتوي الذاكرة المخبأة على فائض من مساحة السليكون نظراً لوجود مصفوفة العلامات والمقارن وخوارزميات الاستبدال . بدون هذه العناصر، تحتوي ذاكرة scratchpad التي تشغل نفس المساحة على سعة أكبر. ويوضح الشكل (2) الفرق في البنية بين الذاكرتين [11]. أي أن ذاكرة scratchpad تحسن مساحة بمقدار ٣٤ ٪ [9]. والتي تترجم إلى زيادة في القدرة التخزينية لنفس المساحة عند المقارنة بين الاثنين. نحن نستخدم تشكيلة الذاكرة كما هو موضح في جدول (1) من أجل تجاربنا.



(b) (a)

الشكل (2) الفرق في **البنية** بين الذاكرة المخبأة وذاكرة SPM، الشكل a الذاكرة المخبأة ، الشكل b ذاكرة SPM

الجدول (1) تشكيلة الذواكر المستخدمة في تجاربنا والطاقة المستهلكة لكل وصول

	Cache Size (bytes)	Cache Energy (nJ)	SP Size (bytes)	SP Energy (nJ)
SP0-C4	4096	4.71	0	N/A
SP2-C2	2048	4.04	2744	1.09
SP4-C1	1024	3.75	4116	1.34
SP5-C0	0	N/A	5488	1.59

٥-٢- الطاقة:

كما في المساحة، وللأسباب نفسها، تستهلك الذاكرة المخبأة المزيد من الطاقة لكل وصول أكثر من ذاكرة scratchpad.ويوضح الجدول (1) قيمة الطاقة لكل وصول للذاكرة المخبأة وذاكرة SPM [6].

تم الحصول على عدد الدورات المستهلكة في الوصول إلى الذاكرة المخبأة وذاكرة scratchpad باستخدام محاكي ARMulator. ثم نضرب عدد الدورات التي حصلنا عليها من المحاكي بقيم الطاقة المناسبة في الجدول (1) لقياس استهلاك الطاقة الكلي على الشريحة .تتضمن هذه الطريقة مشاكل الذاكرة المخبأة الخاصة بحالات الفقد (miss) والاستبدال عند دراسة مقارنة الطاقة لدينا . بالنسبة إلى الذاكرة خارج الشريحة تصل تكلفة الوصول إلى ٤٥.٢ نانو جول [9]. يتم ضرب هذه القيمة بعدد مرات الدخول التي يتم الحصول عليها بواسطةARMulator .

نحن نتتبع مقياسين لقياس فعالية الطاقة لتكوين الذاكرة .الأول هو استهلاك الطاقة البسيط في نظام الذاكرة . وهذا إلى جانب تبديد الطاقة في المعالج والأجهزة الطرفية الأخرى ، قد يشكل الاعتبار الأساسي في تطبيقات محدودة. والثاني هو تقييم الأداء / الطاقة ويقاس بواحدة MIPS لكل واط (أي ما يعادل عدد التعليمات لكل microJoule) وأكثر فائدة لمعظم التطبيقات المضمنة . ويضمن المعالج في كافة عمليات المحاكاة أن MIPS مؤشر أداء صالح.

٥-٣- المحاكاة:

لكل تشكيلة للذاكرة في الجدول (1)، نقوم بتعبئة ذاكرة SPM كما هو موضح في القسم الثالث وتنفيذ محاكاة على جهاز ARMulator .

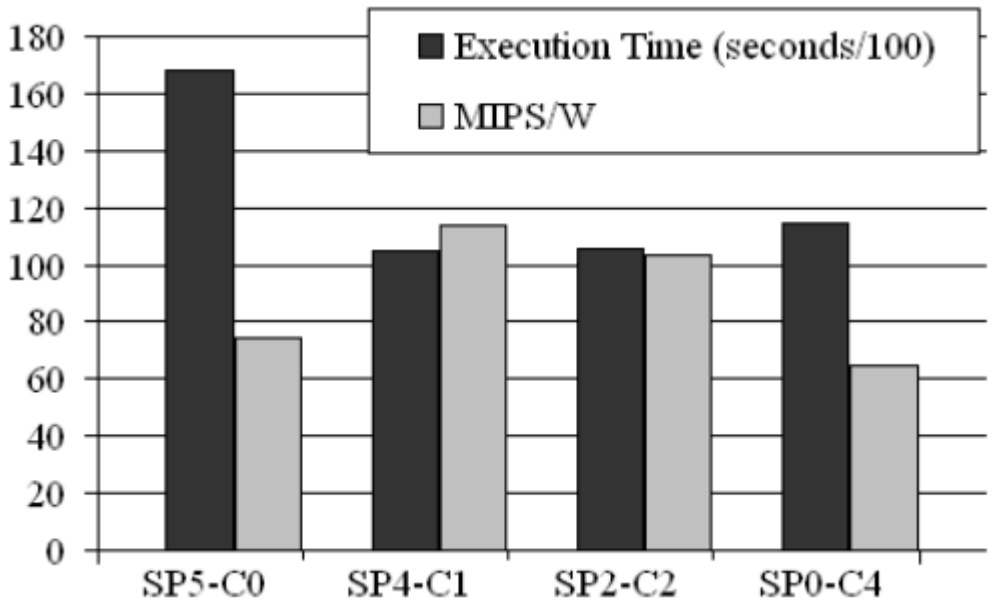
ونستفيد من قدرة ARMulator على محاكاة نظام تصفح الذاكرة ARM710a. افتراضياً ، يكون الحد الأدنى 128 ميغابايت من خريطة ذاكرة المعالج قابلاً للتخزين في الذاكرة المخبأة ، والباقي من مساحة العنوان ٤ جيجا بايت غير مُعتمَدة . نقوم بتعيين ذاكرة scratchpad خارج حدود 128 ميغا بايت .

وبالنهاية يبقى القرار النهائي في تشكيل الذاكرة إلى مصمم التطبيق اعتماداً على أولويات التطبيق (عمر البطارية، سرعة التنفيذ، تبديد الطاقة).

٦- النتائج التجريبية

لقد اخترنا المعيار GSM [10] لتمثيل عبء عمل نموذجي لنظام مضمن حديث (في هذه الحالة ضغط الصوت). وحصلنا على النتائج الموضحة في الشكل (3) ، حيث يبين لنا هذا الشكل زمن التشغيل والطاقة باستخدام الخوارزمية المذكورة سابقاً، والطاقة مقاس فقط من أجل نظام الذاكرة.

يمكننا أن نلاحظ في هذا الشكل أن حالة جميع الذواكر هي SPM ولا يوجد ي ذاكرة مخبأة (الحالة الأولى SP5-C0) هي الأقل فعالية من حيث سرعة التنفيذ. وأن حالة جميع الذواكر عي ذواكر مخبأة دون وجود ذواكر SPM (الحالة الرابعة SP0-C4) هي أقل كفاءة في استخدام الطاقة. ويشير الشكل أيضاً إلى أن مجموعة من ذواكر scratchpad وذاكرة مخبأة واحدة هي أفضل بديل لهذا التطبيق. أي أن التشكيلة SP4-C1 هو الخيار الأفضل بالنسبة لبارامتري وقت التنفيذ وكفاءة الطاقة.



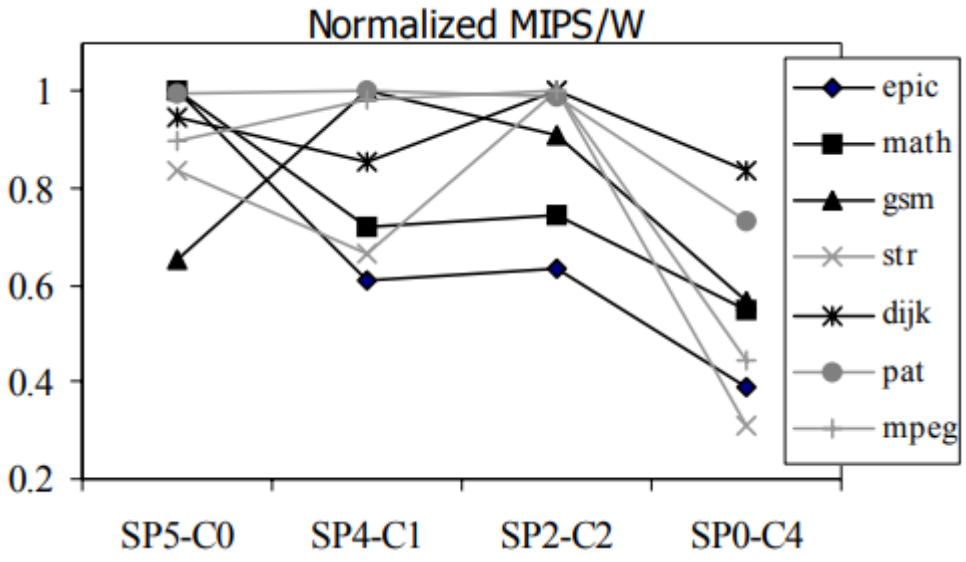
الشكل (3) زمن التنفيذ و MIPS/W للمعيار GSM

إن التشكيلة SP4–C1 تحسن زمن التنفيذ بنسبة 0.04 ٪ فقط مقارنة بالتشكيلة SP2–C2 ، ولكنها تتميز بأن البارمتر MIPS/W يزداد بنسبة 9.7 ٪ .

بشكل أكثر دقة نرى أن التشكيلة SP4–C1 تصل إلى الذاكرة المخبأة بنسبة 7 ٪ ، ولكن أكثر من 20٪ من هذه الوصولات تكون حالات فقد (cache misses) ، أي البيانات غير موجودة في الذاكرة المخبأة ونحتاج إلى الوصول إلى الذاكرة الرئيسية لجلب البيانات.

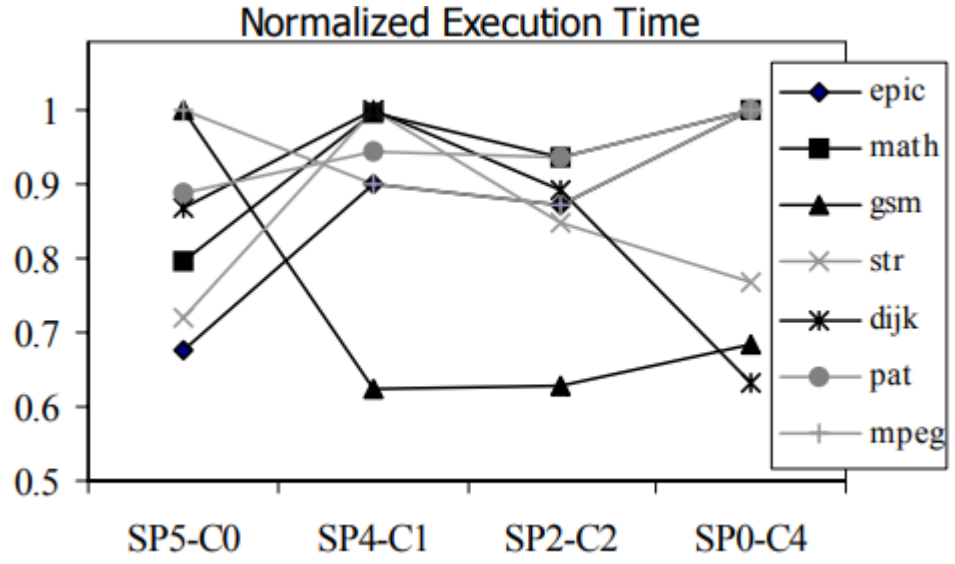
ويتم إهمال هذه القيم من حالات الفقد مقارنة مع نشاط الذاكرة المخبأة الباهظة الثمن (لدينا ذاكرة مخبأة واحدة فقط).

يوضح الشكل (4) رسم بياني يوضح بارامتر MIPS/W وأوقات التنفيذ لسبعة معايير يتيحها لنا المحاكي (بما في ذلك GSM) للحصول إلى أفضل تشكيلة للذواكر حسب كل معيار. تعد التشكيلة التي تحوي ذاكرة مخبأة فقط SP0–C4 أسوأ خيار ممكن لجميع المعايير. أما حالة ذاكرة Scratchpad فقط هو التشكيلة الأفضل في كلا المعيارين Epic و BasicMath ، ولكن في الخمسة المعايير الأخرى ، فإن مزيجاً من الذاكرة المخبأة و Scratchpad هو الأكثر فعالية. أكبر تباين واضح يظهر في تطبيق str حيث يكون تكوين SP2–C2 أفضل بنسبة 70 ٪ تقريباً من SP0–C4.



الشكل (4) MIPS/W من أجل سبعة معايير

يوضح الشكل (5) مخطط بياني لأزمنة التنفيذ للمعايير السبعة. يعد تكوين SP5−C0 هو الأفضل لثلاثة معايير، والأسوأ بالنسبة لمعيارين. أما المعايير GSM و MPEG2ENC تكون أفضل مع تشكيلات الذاكرة غير المتجانسة.



الشكل (5) زمن التنفيذ من أجل سبعة معايير

٧- الاستنتاجات

لقد قدمنا منهجية لتخصيص كل من الذاكرة المخبأة وذاكرة SPM على الشريحة، مع التأكد من أن Scratchpad تعمل بكفاءة. تعتمد الخوارزمية لتعبئة ذاكرة SPM على تحقيق تحسن شامل في أداء نظام الذاكرة ، مع مراعاة تكاليف الذاكرة المخبأة و scratchpad والوصول إلى الذاكرة الرئيسية. إن التشكيلة التي تحوي على الذواكر المخبأة فقط هي الأقل كفاءة في استخدام الطاقة وهو أمر مثير للاهتمام. أما الأكثر إثارة للاهتمام ، والأكثر صلة بشكل مباشر بالدراسة السابقة ، هو عدم وجود تشكيلة مناسبة دوماً لجميع التطبيقات. تبرز هذه الحقيقة فرضية أن اختيار تكوين الذاكرة المناسب يمثل مشكلة خاصة بالتطبيق في الأنظمة المضمنة.

المراجع

[1] P. Panda, N. Dutt, and A. Nicolau, "On-Chip vs. Off-Chip Memory: The Data Partitioning Problem in Embedded Processor-Based Systems," ACM Transactions on Design Automation of Electronic Systems, vol. 5, no. 3, pp. 382-704, Jul. 2010.

[2] ____, "Local Memory Exploration and Optimization in Embedded Systems," IEEE Trans. Comp. Aided Design of Integrated Circuits and Systems, vol. 18, no. 1, pp. 3-13, Jan.2008.

[3] M. Kandemir, J. Ramanujam, M.J. Irwin, and N. Vijaykrishnan, "A Compiler-Based Approach for Dynamically Managing Scratch-Pad Memories in Embedded Systems," IEEE Trans. Comp. Aided Design of Integrated Circuits and Systems, vol. 23, no. 2, pp. 243-260, Feb. 2004.

[4] L. Benini, A. Macii, E. Macii, and M. Poncino, "Increasing Energy Efficiency of Embedded Systems by Application-Specific Memory Hierarchy Generation," IEEE Design & Test of Computers, pp. 74-85, Apr-Jun 2000.

[5] G. Chen et al., "Compiler-Directed Management of Instruction Accesses," in Proc. Euromicro Symposium on Digital System Design, pp. 459-462, Sep. 2003.

[6] S. Steinke, L. Wehmeyer, B.-S. Lee, and P. Marwedel, "Assigning Program and Data Objects to Scratchpad for Energy Reduction," in Proc. Design, Automation, and Test in Europe Conference and Exhibition, pp. 409-415, Mar. 2002.

[7] M. Verma, L. Wehmeyer, and P. Marwedel, "Cache-Aware Scratchpad Allocation Algorithm," in Proc. Design, Automation, and Test in Europe Conference and Exhibition, pp. 1264-1269, Feb. 2004.

[8] Advanced RISC Machines, Ltd., "Application Note 32: The ARMulator," http://www.arm.com/pdfs/AppNote32_ARMulator.zip, Sep. 2003.

[9] R. Banakar, S. Stienke, B.-S. Lee, M. Balakrishnan, and P. Marwedel, "Scratchpad Memory: A Design Alternative for Cache On-chip memory in Embedded Systems," in Proc. of the 10th International Symposium on Hardware/Software Codesign, pp.73-78, May 2020.

[10] C. Lee, M. Potkonjak, and W.H. Mangione-Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems," in Proc. of the International Symposium on Microarchitecture, pp. 330-335, 1997.

[11] WASLY,S. A Dynamic Scratchpad Memory Unit for Predictable Real-Time Embedded Systems. A thesis presented to the University of Waterloo in fulfillment of the thesis requirement for the degree of Master of Applied Science in Electrical and Computer Engineering, Waterloo, Ontario, Canada, 2012.