# A Cyclic Queue for Backward Branch Prediction

# 迴向分支預測的循環佇列機制

羅宏毅
逢甲大學電機工程系
M9405530@fcu.edu.tw

謝宇翔
逢甲大學資訊工程系
D9230810@fcu.edu.tw

王豐
逢甲大學電機工程系
leiwang@fcu.edu.tw

## 摘要

由管線技術的觀點而言，分支指令會造成管線的控制危障(Control Hazard)，導致管線延遲抓取到正確指令的時間。預先猜測分支指令是否發生跳躍，預測正確即可適時抓取分支後的指令，消除延遲。本論文接續既有爲嵌入式處理器所設計之迴向分支預測機制BBQ(Backward Branch prediction Queues)，針對其無法解決之複雜巢狀迴圈結構作研究，根據 BBQ 在處理迴向分支上的行為特徵，特別設計一個簡單而有效的預測機制以解決此類程式行為造成的效能破壞，此一機制將產生一名為 CBBQ(Cyclic Backward Branch prediction Queues)的設計。CBBQ 電路已於研究中完成，而根據模擬發現 CBBQ 對迴向分支的預測正確性，可達到平均 90.51%以上的比例，較原始 BBQ 在預測失誤率的降低上達到 25.15%的改進。

關鍵字：嵌入式處理器，控制危障，分支預測，BBQ

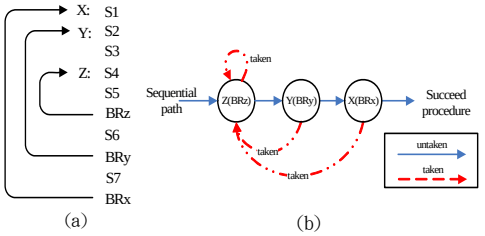
Keyword: Embedded Processor ， Control Hazard， Branch Prediction，BBQ

## 一、序論

由管線技術的觀點而言，分支指令會造成管線的控制危障(Control Hazard) [3]，使管線延遲抓取到正確指令的時間。因此在一般處理器採用的管線架構中，處理分支指令技術之好壞將直接影響處理器的執行效能。分支預測機制[4]可以在管線抓取指令時預測是否依序抓取指令或者是抓取跳躍後位址的指令，當預測正確即可適時抓取分支後的指令，消除上述的延遲。

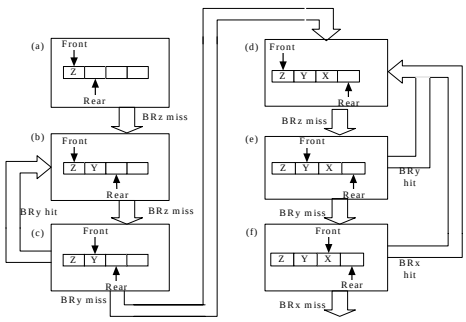
考量嵌入式處理器應達到較傳統高效能處理器更嚴格的成本/效能比，並避免因硬體複雜度的增加而加重電力的負擔，且針對特定的應用行為，以最少量的硬體成本，擴充處理器能力以提升其效能[5]。由於不同的分支指令種類將擁有不同之程式結構和特性，故對於不同類型的分支指令應發展出不同的策略，以尋找最符合其分支指令種類的預測機制。本研究過去曾提出針對迴向分支(Backward Branch)設計之預測機制 BBQ (Backward Branch prediction Queues)[1]，其運作機制乃基於對構成迴圈之迴向分支指令加以記錄並作為預測來源，而不需盲目儲存所有分支指令以進行預測。而程式中的巢狀迴圈行為可以圖一說明。

茲以圖一(a)所示之程式結構為例，X：Y：Z：別代表迴向分支之目的位址(Target Address)；$BR_Z$、$BR_Y$、$BR_X$ 則為迴向分支指令的代稱，S1 至 S7 則代表其它非分支指令。而在圖一(b)中，圓圈 Z、Y、X 代表各層迴圈，虛線表示迴向分支指令跳躍，實線則表示迴向




圖一: 三層巢狀迴圈的行為模式

分支指令不跳的循序流程。由於迴圈行為的規律，而且其巢狀結構亦易於由指令中獲知，因此 BBQ 即以如圖二所示的處理方式來完成此類行為的分支預測，其基本架構包含四組儲存欄位與兩組控制指標：讀出指標 front 用以讀出對應欄位的資料提供預測，而寫入指標 rear 則指向準備寫入新資料之儲存欄位。程式初次執行至 BRz 且發生分支行為時，BBQ 即儲存此 $BR_Z$ 之 PC 位址與目的位址如圖二(a)，並在 Z：迴圈重複執行時用以預測。當程式脫離 Z：迴圈向下執行，BBQ 並不立即清除舊有紀錄，而在判斷接續分支指令 $BR_Y$ 包含 $BR_Z$ 且構成巢狀迴圈後，選擇接續建構分支紀錄如圖二(b)。由於 $BR_Y$ 發生分支行為將使程式執行重新回到 Z：迴圈，故 Front 指標將調回欄位 Z 直到程式脫離 Z：迴圈後再指向欄位 Y，如圖二(b)、(c)所示。遵循上述對迴向分支指令的處理原則，即可推知 BBQ 遭遇巢狀迴圈分支指令 $BR_X$ 時之運作流程如圖二(d)、(e)、(f)。

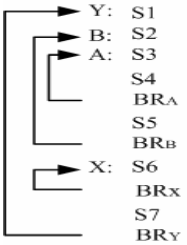



圖二：三層巢狀迴圈在 BBQ 中之處理過程

此一簡易的預測機制經由模擬程式實驗發現，可在極低之硬體成本基礎上對迴向分支的預測正確性，達到平均 87.32%以上的比例，並在整個程式執行上產生加速 8.62%的執行效果。

然而在之前的研究中，我們注意到由於 BBQ 僅針對巢狀迴圈結構加以考慮，以單一一組 BBQ 儲存最近執行的巢狀迴圈之迴向分支指令；而透過模擬程式的觀察，發現在以下特定的程式結構會導致部份模擬程式(如 bitcount、Rijndael)預測正確率下降：

當程式迴圈結構中包含多個子迴圈如圖三所示，當執行由上半部的 B：A：迴圈進入下半部的 X：迴圈結構時，原先的 B：A：迴圈將因被判定脫離而使得其紀錄被清除，而通過外層迴圈（Z：迴圈）回復上半部執行時，不僅 B：A：迴圈將因此而產生預測失敗且重建，並因而又造成下半部的 X：迴圈被清除。最後導致外層迴圈每次都要透過 BBQ 的預測失敗以建立上/下半部巢狀迴圈的建立，如此重複的清除與建構動作使預測正確率降低。




圖三：包含多個子迴圈之多重巢狀迴圈

上述問題之特徵為無論 BBQ 儲存欄位是否裝滿，已存在之迴圈紀錄受到程式結構之干擾而被誤判清除，但在短時間內又會再次建構，從而對 BBQ 造成不必要的預測失誤。

爲達到以簡單的電路設計與少量的硬體成本對 BBQ 加以改良以解決此一干擾，使 BBQ 具有解決複雜的巢狀迴圈結構或副程序呼叫所造成之預測失誤之能力，本文提出一利用比較已儲存之迴圈記錄進行動態的預測指標調整，並採取循環方式管理有限的 BBQ 儲存欄位以充分保留有效的迴圈紀錄供預測使用之改良設計，稱之為 CBBQ(Circular Branch prediction Queues)。在以下的敘述中，將說明此 CBBQ 之設計理念與硬體結構，並根據模擬分析以證明其效能的改進。在下一章中，我們將以造成 BBQ 效能損失之巢狀迴圈結構為例，詳細解釋其在 BBQ 運作中之特性，並因應其特性發展出 CBBQ 的基本處理流程。詳細的 CBBQ 電路則在第三章中敘述。而第四

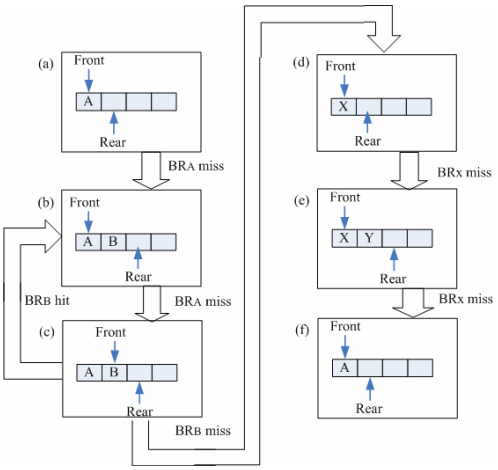
章的模擬測試，則透過應用程式的動/靜態模擬分析以驗證 CBBQ 對迴向分支預測的改進效果。在最後的結論中，本文將綜合 CBBQ 的硬體設計及效能分析，討論 CBBQ 在嵌入式處理器中的應用形式及未來可做進一步改進的研究方向。

## 二、 CBBQ 迴向分支預測機制

本文提出之 CBBQ 迴向分支預測機制乃自過去研究設計之 BBQ 機制改良而成，目的在於解決特定程式行為造成 BBQ 效能損失的問題。本章將先簡述此類行為於 BBQ 中之運作流程，並觀察其特性以發展對應策略，最後說明改良機制 CBBQ 的基本設計與處理流程。

由於 BBQ 運作策略係針對簡單之巢狀迴圈設計而成，其 Front 指標調整與分支記錄建構機制對於大型巢狀迴圈或多組巢狀迴圈的程式行為並不能達到最佳的效果。茲以圖三之大型巢狀迴圈為例，圖四為 BBQ 在此程式行為中之運作流程。BBQ 首先建構此大型巢狀迴圈中的 B：A：迴圈紀錄如圖四(a)、(b)、(c)，當程式執行脫離 B：A：迴圈並遭遇 $BR_X$ 時，由於 BRx 與 BBQ 內已存在之 B：A：迴圈紀錄並無構成巢狀迴圈關係，BBQ 將清除舊有記錄並重新建構 X：迴圈於最初欄位。當程式執行脫離 X：迴圈後，迴向分支 $BR_Y$ 將在 BBQ 建構 X：Y：迴圈且使程式執行再次回到 B：A：迴圈中如圖四(d)、(e)、(f)所示，由於 BBQ 已清除此 B：A：迴圈之舊有記錄且迴向分支 $BR_A$ 並未包含 $BR_Y$ 形成巢狀迴圈，導致 BBQ 須清除 X：Y：迴圈紀錄並重新由 $BR_A$ 開始建構 B：A：迴圈。在此例中，可以發現 BBQ 處理包含多組巢狀迴圈之大型迴圈結構時，容易因無法保存仍然具有使用價值之分支記錄而造成預測正確率降低。

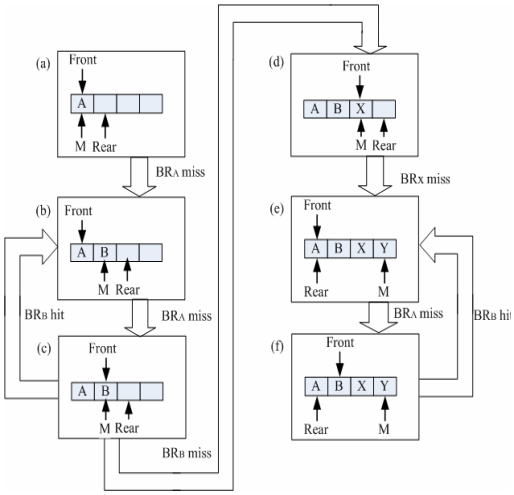
進一步觀察 BBQ 的行為流程後，我們注意到由於 BBQ 僅使用最外層之儲存欄位與所遭遇的迴向分支就是否構成巢狀迴圈進行判斷，故無法同時儲存多組不同之巢狀迴圈結




圖四：多重巢狀迴圈在 BBQ 中之處理過程

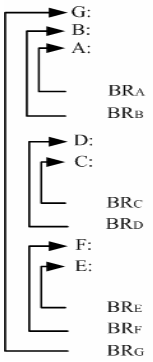
構。當程式執行至不同的巢狀迴圈結構時，即使 BBQ 尚有儲存欄位仍必須先清除舊有記錄再另行紀錄新的迴向分支，導致無益的儲存電路浪費。

本論文針對上述 BBQ 的缺陷加以改良，其想法是修改 BBQ 儲存管理策略與指標控制，使 BBQ 遭遇複雜的巢狀迴圈時不須立即清除舊有紀錄，而能充分利用儲存欄位以保留不同的巢狀迴圈結構並精確地選擇對應的分支紀錄進行預測，其效果在上例中將如圖五所示，主要修改之處為增加一組控制指標 M，用以讀取 BBQ 中巢狀迴圈最外層之分支紀錄。當程式執行脫離巢狀結構 B：A：迴圈如圖五(c)所示時，不同於原始 BBQ 直接清除非巢狀迴圈結構的處理方式，改良之 BBQ 設計在判斷當前分支並未與舊有紀錄構成巢狀迴圈時選擇將 X：迴圈接續建立於已有的 B：A：迴圈紀錄後並透過調整 Front、Rear 與 M 指標正確進行分支預測如圖五(d)。圖五(e)、(f)則顯示改良之 BBQ 在程式執行脫離 X：迴圈來到迴向分支 $BR_Y$ 時，因 $BR_Y$ 與已有紀錄構成巢狀迴圈故接續建立 Y：迴圈並在比較各分支紀錄與 $BR_Y$ 跳躍位址後調整 Front 指標至被保留的最內層迴圈 A，從而避免因複雜的巢狀結構導致 BBQ 預測正確率下降。



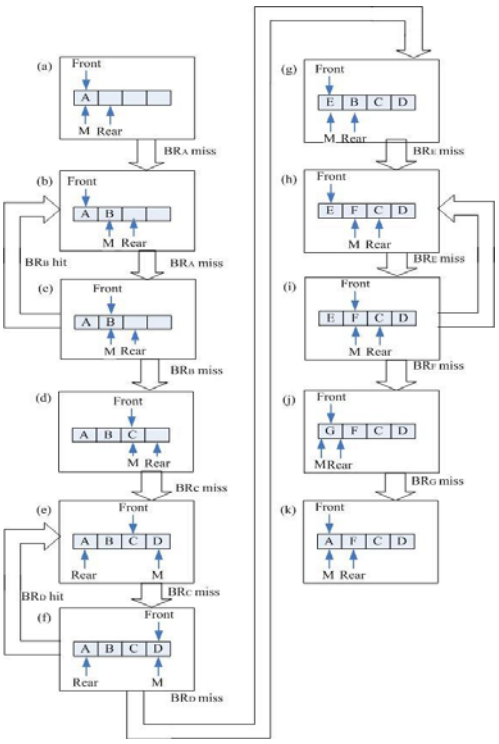

圖五：改良之 BBQ 設計，CBBQ

不主動清除分支紀錄的儲存策略將導致有限的 BBQ 欄位無法在存滿分支紀錄後繼續對新的迴向分支進行預測，故在迴向分支紀錄超過 BBQ 容量時，CBBQ 將以循環方式使新的迴向分支覆蓋至 BBQ 的開頭欄位。我們以圖六所示之程式結構為例，圖七則為該程式結構超過儲存容量時在 CBBQ 中的處理過程。




圖六：六組迴向分支構成之大型巢狀迴圈

當程式執行脫離 B：A：D：C：迴圈如圖七(f)所示並遇到迴向分支 $BR_E$ 時，由於發生的迴向分支能儲存於 CBBQ 中且可用以進行預測。但在程式執行脫離 E：F：迴圈來到迴向分支 $BR_G$ 時，由於 $BR_G$ 所構成的巢狀迴圈包含尚未被清除之 D：C：迴圈，此一情況將導致無法簡單地透過比較方式判斷 Front 指標應如何調整，此時 CBBQ 將在接續建構 $BR_G$ 紀錄並遭遇迴向分支 $BR_A$ 時從開端欄位以覆蓋方式建構 A：迴圈如圖七(k)。

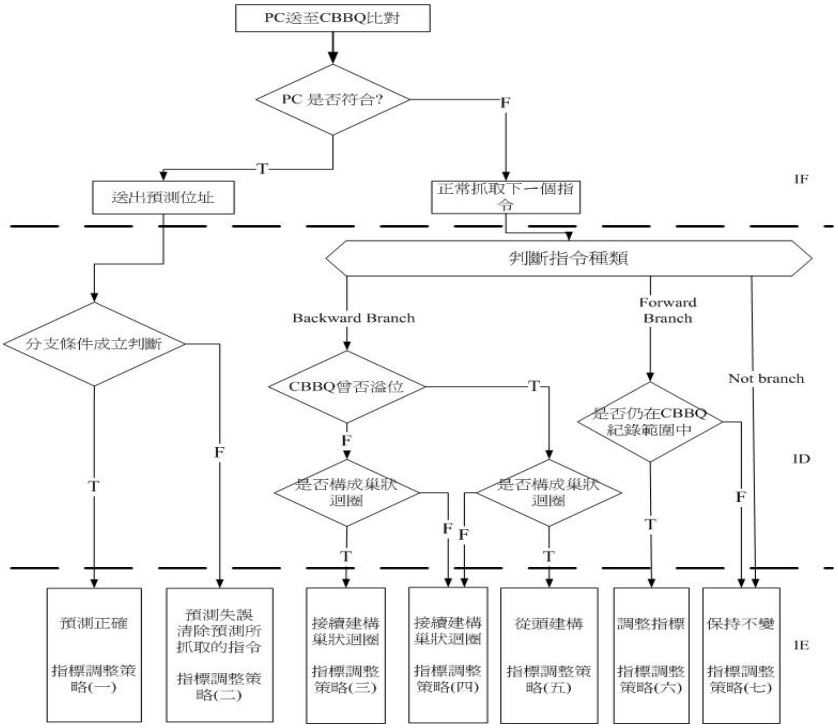



圖七： CBBQ 處理大型巢狀迴圈過程

## 三、CBBQ 之硬體設計

本研究以 ARM-9 的五階管線及微架構為基礎，將 CBBQ 作業模式融入處理器的指令管線流程中，其作業將如圖八所示在 IF、ID、IE（即 Fetch、Decode、Execution）三個階段產生如下所述之動作:

『IF Stage』：PC 送正欲抓取的指令位址到 CBBQ 中，與 CBBQ 中 Front 指標所指之紀錄比對決定是否為 CBBQ 紀錄過為目前預測的迴向分支指令，若經過比對結果符合，即送出預測分支指令的目的位址供下一指令抓取指令的位址；若不符合，則 CBBQ 不做特定動作，管線正常執行。『ID Stage』：此處需要分為兩部分討論－『左線流程』表示指令為以往執行過的迴向跳躍指令，並已在前一階段產生預測分支的效果，當其條件分支指令之條件成立時，則表示 CBBQ 預測正確；反之條件不




圖八：CBBQ 工作流程圖

成立則表示 CBBQ 的預測失誤，此時必須清除 CBBQ 預測所抓取的指令，並恢復管線抓取正確的指令位址。『右線流程』表示指令為非紀錄於 CBBQ 目前預測之迴向分支指令或能造成程式流程改變之前置分支指令。當執行指令經解碼判斷為分支指令時，若指令為迴向分支指令且發生跳躍時，會判斷其跳躍的目的位址與 CBBQ 中最外層巢狀紀錄 M 所對應之跳躍紀錄是否構成巢狀，而在構成巢狀迴圈的判斷上，其定義為:新的迴向分支指令其目的位址與 PC 包夾 M 指標對應的儲存欄位(圖一(a)為例判斷 BRz 與 BRy 是否構成巢狀迴圈以(Target(BRz) ≧ Target(BRy) && PC(BRz) < PC(BRy)))判定；若未構成巢狀則表示已脫離 CBBQ 已紀錄之巢狀迴路，兩者皆於 IE Stage 更新其對應之 CBBQ 欄位紀錄。CBBQ 曾否溢位用以判斷是否目前的迴圈紀錄是否已建構至欄位末端而循環回到欄位開端，為避免巢狀迴圈存在於不連續的儲存欄位上導致無法正確判定最內層迴圈所在位置，此時 CBBQ 將重新由開端欄位建構巢狀迴圈。若指令為前置分支指令且發生跳躍時，則根據其跳躍的目的位址與全 CBBQ 紀錄判斷控制指標是否需調整至正確的對應位置。

『IE Stage』：CBBQ 根據 ID Stage 的條件判斷後，在 IE 進行指標調整，其調整方式如下所述，同樣可分為左線流程與右線流程：

調整策略(一)：當 CBBQ 已記錄之迴向分支指令再一次被執行，預測其跳躍而的確發生跳躍，此時 CBBQ 預測正確，根據巢狀迴圈的特性，若未有其他分支指令更改程式流程，下次必然回到 CBBQ 所構建之巢狀迴圈最內層。CBBQ 將在比較該分支跳躍位址與舊有紀錄而判斷出巢狀迴圈的最內層後，更改讀出預測位址的 Front 指標令其指向巢狀迴圈的最內層，其餘指標保持不動。

調整策略(二)：當 CBBQ 已記錄之迴向分支指令再一次被執行，預測其跳躍但因不符合執行條件未發生跳躍時，因上述行為表示程式流程已脫離現在的迴圈而進行到下一層迴圈，CBBQ 會將讀出預測位址的 Front 指標指向下一個儲存欄位(即指向下一層迴圈)以

進行預測，其餘指標則保持不動。

調整策略(三)：當指令經過 ID Stage 的流程，確認此指令為一不存在 CBBQ 紀錄欄位中且發生跳躍的迴向分支指令，並經由位址比較發現該指令與 CBBQ 欄位所儲存指令構成巢狀迴圈，此時若 CBBQ 未曾儲存至末端欄位而發生循環，則此分支指令需根據 Rear 指標接續存入 CBBQ 欄位中，並將 Front 指標調至巢狀結構最內層，M 指標指向新建立之分支欄位而 Rear 指標向下調整一欄位。

調整策略(四)：類似策略三，當指令經過 ID Stage 的流程，確認此指令為一不存在 CBBQ 紀錄欄位中且發生跳躍的迴向分支指令，但經由位址比較發現該指令與 CBBQ 欄位所儲存指令未構成巢狀迴圈，此時無論 CBBQ 曾否儲存至末端欄位而發生循環，該分支指令皆可根據 Rear 指標接續存入 CBBQ 欄位中，並將 Front 指標、M 指標指向新建立之分支欄位而 Rear 指標向下調整一欄位。

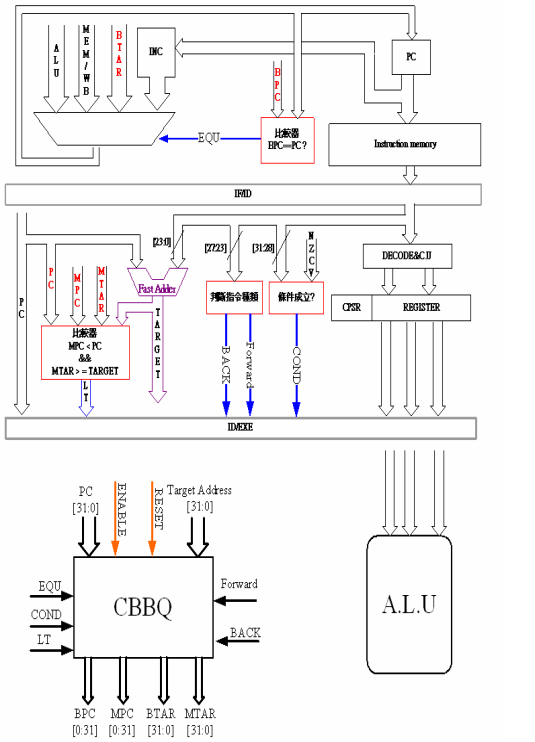
調整策略(五)：當指令經過 ID Stage 的流程，確認此指令為一不存在 CBBQ 紀錄欄位中且發生跳躍的迴向分支指令，並經由位址比較發現該指令與 CBBQ 欄位所儲存指令構成巢狀迴圈，此時若 CBBQ 已儲存至末端欄位而發生循環，為避免比較機制誤判最內層迴圈所在欄位，CBBQ 將由最初欄位開始覆蓋舊有紀錄。該分支指令將直接存入 CBBQ 開端欄位，並將 Front 指標、M 指標指向新建立之分支欄位，而 Rear 指標調向次一欄位。

調整策略(六)：若指令為前置分支指令且跳躍位址未超出 CBBQ 中已存迴向分支的 PC 範圍，在比較各儲存欄位與此前置跳躍位址後 CBBQ 將控制 Front、Rear、M 指標調整至對應位址。

調整策略(七)：若指令為跳躍位址超出 CBBQ 中已存迴向分支的 PC 範圍之前置分支指令或其他非分支指令(包含條件失敗而未跳躍之條件分支指令)，由於此類行為不會對程式流程造成改變，此時 CBBQ 將保持不變。

本文所提出之 CBBQ 預測機制乃由舊有 BBQ 預測機制改良設計而成，藉由部份修改 BBQ 原始電路架構並增加控制指標調整電路，CBBQ 可以在不增加過多硬體成本負擔的條件下有效改良複雜巢狀迴圈結構對分支預測正確性的傷害。

由於 CBBQ 僅針對 BBQ 電路中控制指標及儲存管理進行修改，其主體架構與原始 BBQ 並無太大差異。圖九為 CBBQ 於處理器管線電路中的結構，以下亦以 IF、ID、IE 三個階段分別說明有關之硬體設計。




圖九：CBBQ 於 IF、ID、IE 管線階段之電路架

『ID Stage』：當指令進入解碼階段後，首先我們將抓取指令之[27:23]BIT，判斷指令是否為分支指令並分辨分支指令其為前置跳躍或迴向跳躍，利用 BACK 與 Forward 兩條 1-BIT 控制訊號線輸出供下一階段 BBQ 電路使用，取[31:28]BIT 條件欄位與 NZCV 旗標值判定指令條件是否成立，並將判斷結果利用 1-BIT 訊號線 COND 輸出供下一階段 BBQ 控制電路。快

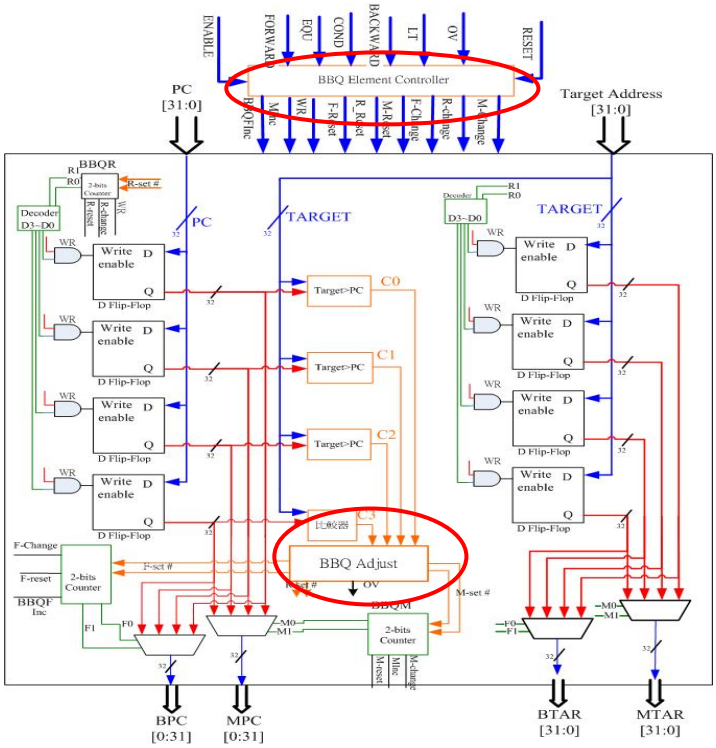
速加法電路用以提早於 ID 階段取得分支指令的目的位址，之後即可提早在解碼階段判斷 BBQ 儲存的迴向分支跳躍紀錄與新的迴向分支指令是否構成巢狀迴圈，或是否造成破壞 BBQ 預測機制的錯誤。而電路上利用比較器將 MTAR(巢狀迴圈最外層之目的位址)、MPC(巢狀迴圈最外層之 PC 值)與新發生的分支指令之目的位址及 PC 值作比較，判斷的結果利用 1-BIT 訊號線 LT 輸出供下一階段 BBQ 控制電路辨別。

『IE Stage』: CBBQ 電路於執行階段主要是根據 CBBQ 預測機制選擇讀出預測指令並更新儲存欄位，整個設計將分成儲存電路、控制電路、指標調整電路(如圖十)等部份分項說明。

儲存電路：在 CBBQ 電路中，儲存欄位主要是由兩組 32-BIT 的 D-TYPE 正反器所組成，分別儲存分支指令跳躍記錄所需的 PC 值和目的位址，欄位的多寡決定 BBQ 可以處理的巢狀迴圈層數之大小。Front 指標與 Rear 指標分別由 BBQF、BBQR 二組計數器控制並用以選擇 CBBQ 欄位之讀出與寫入，透過 BBQM 計數器可選擇 BBQ 欄位中巢狀迴圈最外層之對應欄位讀出。

控制電路：主要的功能是控制 CBBQ 欄位的讀出與寫入，以及依據解碼階段判別結果與儲存欄位曾否溢位旗標 OV 決定當前 BBQF、BBQR，BBQM 三個計數器是否需透過調整電路進行修改以符合流程圖之要求，此一部分完全只是一個組合電路的設計。指標調整電路：爲完成 CBBQ 指標動態調整與循環式儲存管理之改良目標，我們設計一指標調整電路(圖十之 BBQ Adjust)，利用分支指令之目的位址與目前 BBQ 儲存欄位中各個 PC 值比較其大小關係，經比較器判斷後結果分別輸出為 C0、C1、C2、C3 並在控制電路判斷須進行指標調整時透過比較結果與組合邏輯電路，產生 BBQF、BBQR、BBQM 計數器應更改的數值以進行修改，此一部分亦僅是一個組合電路之實作。

由以上之電路設計可知，CBBQ 的電路修改部分僅在控制電路與旗標調整電路，其增加的硬體成本僅為一組額外的比較器與一些




圖十：CBBQ 儲存電路與控制電路

條件判斷的組合邏輯電路，與 BBQ 相比並未造成過多的硬體成本增加而能充分利用有限的儲存欄位進行分支預測。因此 CBBQ 之硬體成本同樣遠低於複雜的 BTB 或類似的分支預測機制，且其反應速度亦遠比其他預測機制快速。

## 四、效能評估和模擬

為了驗證CBBQ預測機制是否能比BBQ更有效的解決控制危障，並評估其針對迴向分支指令預測的正確性，我們利用 Mibench[6]作為標準效能測試程式並以 Simplescalar[7]模擬程式作為模擬評估的測試平台，最後將獲得的模擬數據整理並分析，證明 CBBQ 預測機制的價值。

(一)測試平台及模擬程式

本研究之測試平台採用 SimpleScalar LLC[8]提供的 SimpleScalar Version 4.0 test released － SimpleScalar/ARM simulator ，SimpleScalar模擬環境最初是由Todd Austin所開發，免費提供學術上的研究與應用以及教學目的之用途，並提供所有的原始碼(source code)供SimpleScalar使用者針對所需的硬體架構修改程式以符合使用者需求。其處理器模型為 Intel SA-1 五階管線、兩層記憶體階層，類似於 ARM-9 的五階管線微架構[2]，具有支援 ARM 指令集的能力。SimpleScalar 為 Execution-driven 的模擬器，能實際執行 Benchmark 程式並同時產生執行紀錄(Trace)。

本研究使用的測試程式集(Benchmarks)為 Mibench(音譯 My bench )，該程式集是由密西根大學電機工程計算機實驗室（The University of Michigan Electrical Engineering and Computer Science）所發表，是以 EEMBC (EDN Embedded Microprocessor Benchmark Consortium)[9]為基礎並針對嵌入式系統及專為提供學術研究上所發展的 Benchmarks。Mibench 共將 benchmarks 分為 Automotive、Industrial Control、 Consumer Devices、Office Automation 、 Networking 、 Security 、 Telecommunications 六大類。

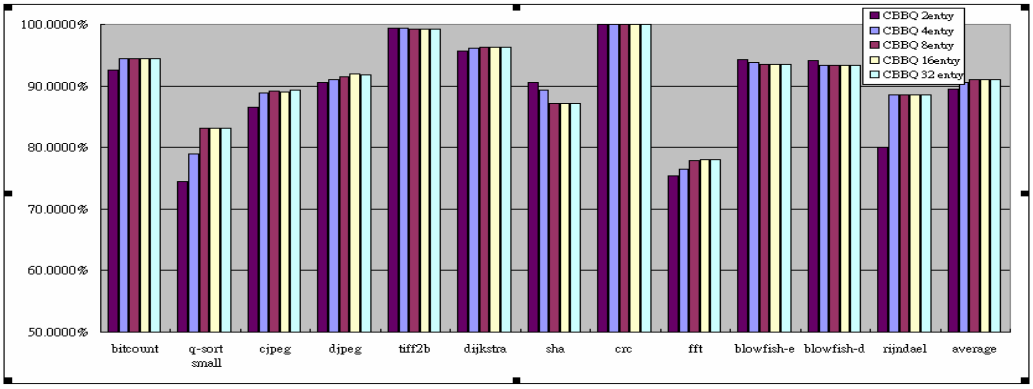
本研究的模擬在 Benchmark 的選擇上選用較常見的 Benchmark 並優先選用同時符合兩種用途的程式，因此挑選 Bitcount、Quicksort、Jpeg encode/decode、Tiff2bw、Dijkstra、SHA、Blowfish encrypt/decrypt、Rijndael encrypt/decrypt、CRC32、FFT 等共 12 個程式做模擬以分析 CBBQ 的預測效率。

(二)效能分析與統計結果

本文首先利用 Simplescalar 架構產生各 Benchmark 實際執行所得到的包含指令碼與 PC 值之全程式 Trace 檔，再將此 Trace 檔案輸入描述 CBBQ 行為的模擬程式以觀察 CBBQ 的分支預測正確性。為確保模擬評估的正確性與可信度，在模擬的過程中我們並未修改 Benchmark 或是抽取 Benchmark 程式片段做模擬評估，且在模擬程式的參數設定上不限制 Benchmark 指令執行之數目，因此在動態指令執行數目上相當可觀，而其模擬的結果亦較能真實的評估架構之效能。

考慮到本研究以最少量的硬體擴充處理器功能以提升其效能的基本精神，首要的目標即為合理的選擇 CBBQ 之儲存欄位大小。圖十一為不同大小之 CBBQ 效能比較，我們發現由 2 entry 增加至 4 entry 時整體平均預測正確率即已由 89.45%增加至 90.51%，而再行增加至 8 entry 則能提升至 90.98%。因當初原始 BBQ 的模擬發現 4 entry 的配置已十分接近 8 entry 的效果，因此許多模擬觀察均以 4 entry 為主，故而本研究亦以 4 entry 之 CBBQ 作為比較基本。整體而言 CBBQ 當 entry 數增加預測正確率亦應隨之提升；但此點在 sha 與 Blowfish-D、Blowfish-E 上會出現 entry 數多反而效能降低的異常現象，此點將在下一段落說明。

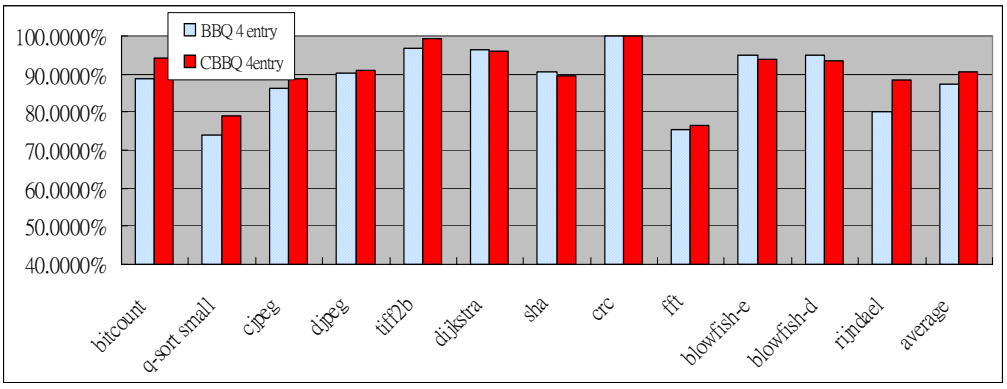
圖十二為我們比較原始 BBQ 預測機制對迴向



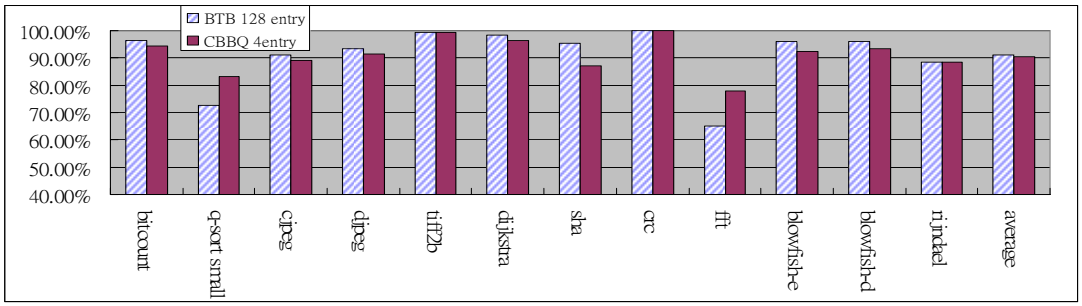

圖十一：不同大小之 CBBQ 的預測正確率

分支指令預測正確率與 CBBQ 之預測正確率，兩者的儲存欄位大小均為 4 entry，由模擬後的數據發現 CBBQ 預測正確率較原始 BBQ 在大部分之 Benchmark 中均有理想的預測正確率提升，僅 sha、Blowfish-E、Blowfish-D 這三個 Benchmarks 預測正確率略低於原始 BBQ，因此我們針對 sha 與 Blowfish 討論其命中率較低原因。此處以 sha 為例說明，在觀察此 Benchmarks 程式結構後，發現其程式結構中含有包含 BL 呼叫指令之巢狀迴圈且 BL 呼叫的副程序中具有多組迴圈結構，當程式執行 BL 指令呼叫副程序後容易發生 CBBQ 儲存溢位卻未覆蓋主程式之巢狀迴圈紀錄，此時將因巢狀迴圈的分支紀錄分布在不連續且順序錯亂之欄位上(若能完全覆蓋則 CBBQ 方可發現已脫離所執行之副程式區塊並進而重新清除 CBBQ 以重建新的迴圈結構)，導致 CBBQ 由副程序返回主程序時無法透過比較的方式判斷巢狀迴圈最內層之對應欄位，因而使後續的預測欄位(Front 指標)發生短暫的錯亂而使預測失誤。此外，在小容量之 CBBQ 中，此類行為容易因副程序之分支紀錄完全覆蓋主程式之巢狀迴圈紀錄而未受到，反而是較大容量的 CBBQ 會產生未完全覆蓋而誤判的情況。但除了上述三個 Benchmarks 出現較不理想的命中率外，其餘都有理想的命中率改良並以 bitcount、Rijndael 提升幅度最高，Rijndael 之預測命中率甚至可由原本 79.99%增加至 88.50%以上，而平均命中率則由 87.32%提升至 90.51%，減少了 25.15%的預測失敗率。

目前的嵌入式處理器對分支預測的作法一般可分為 2 類，較低階(如 ARM7TDMI[10])無分支預測的設計，較高階的(如 INTEL Xscale[11])則採用以大容量低關聯式設計的 BTB(Branch Target Buffer)[12]作為分支預測機制，如 Intel Xscale 即使用 128entry 直接映




圖十二：CBBQ 與原始 BBQ 之預測正確率比較

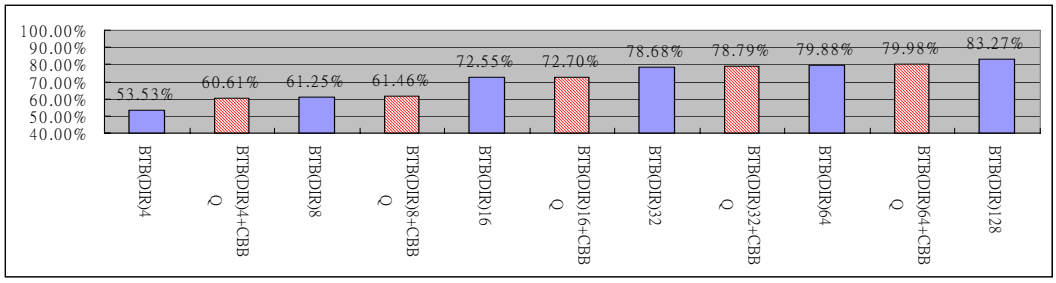



圖十三： CBBQ 與 4-way 集合關聯式 BTB 之預測正確率比較

射式(direct map)之 BTB 作為分支預測電路。圖十三為 CBBQ 與 128entry 4-way 集合關聯式(set associatvie) BTB 的迴向分支預測命中率比較，其中 BTB 對迴向分支的預測正確率平均為 91.77%。由圖中數據可以觀察到使用 4 個儲存欄位之 CBBQ 與具有 128 個欄位之 BTB 相較其預測正確率僅相差 1.26%，可發現 CBBQ 能以簡單的硬體(僅該 BTB 的 4%左右)達到接近於 BTB 的效能達 98.62%以上，證明 CBBQ 的設計符合以極低的成本提高效能成本比之理念，適合嵌入式處理器使用。

雖然 CBBQ 預測機制在單獨使用上即可對於迴向預測帶來不錯的預測正確性，但無法對前置分支進行任何的預測。因此在整體的分支預測改善上，將 CBBQ 與 BTB 進行整合並利用 BTB 處理前置分支指令不失為一有效的分支預測機制。圖十四為整合 CBBQ 與 BTB 之預測機制與單獨使用 BTB 預測機制的分支預測正確率比較，前者的預測機制為迴向分支採用固定 4 個欄位的 CBBQ 進行預測，而前置分支則以直接映射式 BTB 加以處理；後者則無視分支種類全部採用直接映射式 BTB 進行預測。我們首先觀察到，當 BTB 欄位過小時因 BTB 不足以儲存有效分支指令而導致其預測正確性偏低，但隨著 BTB 欄位的增加其預測正確性亦快速的成長直至 32 欄位後提升方趨於和緩。而在整合預測機制與單純 BTB 的比較方面，當 BTB 增加至 64 欄位時，CBBQ 與一半容量之 BTB 的組合機制預測正確性即超過單純使用 BTB 進行預測的正確性，而 CBBQ 配合 64 欄位 BTB 之預測正確率亦超過 128 欄位的 BTB。

根據以上的模擬評估，我們可以發現 CBBQ 的結構不僅構造簡單，並能在大部份的 Benchmark 中表現出超過 90%以上的預測正確率；而由這些模擬數據中，本研究也發現出 CBBQ 可進一步再度改良的方向，由 sha 與 Blowfish 的程式行為分析，我們可以發現，只要 CBBQ 能再有效的排除程式呼叫/回返所造成的儲存干擾，即可避免主副程式間的預測污染效應，使整體預測正確率再度提升。




圖十四： CBBQ 配合直接映射式 BTB 之預測正確率

## 五、結論

由於嵌入式處理器其強調結構單純、支援特定應用、低成本、低耗電的特性，為以少量的硬體成本，擴充處理器能力以提升其效能/成本比，CBBQ 迴向分支預測機制在對 BBQ 電路進行少量修改後，能產生對迴向分支平均達 90.51%的預測正確率而 CBBQ 改良之循環儲存管理機制，更可有效解決複雜巢狀迴圈對 BBQ 預測機制的干擾問題且充分利用所有儲存欄位進行分支預測，較原始 BBQ 在預測失誤率的降低上達到 25.15%的改善。

在未來 CBBQ 的後續研究上，對於程式呼叫/回返的行為所造成 CBBQ 之預測污染效應，我們將強化其分支管理策略使 CBBQ 預測機制更臻完美，同時透過硬體實作進一步的評估其電路成本，尋求 CBBQ 最佳硬體成本與效能比。

## 六、致謝

本研究接受國科會計畫編號：NSC 95-2221-E-035-009 之部分經費補助。

## 七、參考文獻

[1] 曾瓊賢，王壘，”BBQ_一種簡易有效的迴路預測電路”，中華民國九十四年全國計算機會議(NCS2005),12月,2005(NSC 93-2213-E-035-020)

[2] 羅宏毅，王壘，”SmartARM處理器微架構之設計與改良”，中華民國九十四年全國計算機會議(NCS2005),12月,2005(NSC 93-2213-E-035-020)

[3] J.L. Hennessy and D.A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann, San Francisco, CA, 1996.

[4] J.E. Smith, “A study of branch prediction strategies”, in 8th Int. Symp. Computer Architecture,p.135-148,1981

[5] Peter Petrov, Alex Orailoglu, "Low-power Branch Target Buffer for Application-Specific Embedded Processors," dsd, p. 158, Euromicro Symposium on Digital Systems Design (DSD'03), 2003

[6] Matthew R Guthaus, Jeffrey S Ringenberg, Dan Ernst. MiBench: A free, commercially representative embedded benchmark suite. In IEEE 4th Annual Workshop on Workload Characterization, Austin, 2001, p.1-12

[7] Doug Burger and Todd M. Austin. Evaluating Future Microprocessors: the SimpleScalar Tool Set. Technical Report 1300, Computer Sciences Department, University of Wisconsin, Madison, WI, April 1996.

[8] D.C. Burger and T.M. Austin. The SimpleScalar Tool Set, Version 2.0. Technical Report CS-TR-97-1342, University of Wisconsin-Madison, June 1997.

[9] EDN Embedded Microprocessor Benchmark Consortium , http://www.eembc.org.

[10] ARM7TDMI Technical Reference Manual, Document Number: ARM DDI0210B, ARM Limited 2001.

[11] Intel Corporation, “The Intel XScale Microarchitecture Technical Summary,” ftp://download.intel.com/design/intelxscale/XScaleDatasheet4.pdf.

[12] H. Perleberg and A. J. Smith, "Branch target buffer design and optimization. IEEE Transactions on Computers,Vol.42 NO.4 p.396--412, April 1993