Introduction

* Contrarily to static binary translation (BT), and because translation is performed dynamically, it usually costs to a software-implemented DBT thousands of instructions to translate and optimize a source Instruction Set Architecture (ISA) instruction, which makes DBT a hard challenge to perform in low-end devices, which are commonly scarce in resources.
* The major costs associated with a DBT system comprises several translation and emulation overheads, along with other potential runtime costs.
* Borin et al. presented a strategy to identify the main sources of overhead involved in the process of DBT [3].
* Yao et al. identified that common DBT systems suffer performance loss because of architectural heterogeneity among ISAs, control flow and context switches [4].
* DBTIM is a hardware-assisted DBT architecture for full virtualization [5].
* The use of FPGA fabric to promote binary compatibility is advocated in [6] and in [7].
* This article presents a hardware-based mechanism to accelerate and extend the functionalities of DBT applied to the resource-constrained embedded devices, approaching hardware offloading to address some limitations of the DBT engine regarding the TCache management mechanism.

Translation Cache

* DBT 執行過程
  + DBT engine 用 translation module, execution module處理source binary，而每次處理都使已basic block(BB)為單位，用basic block的第一道指令的address來辨認不同的basic block
  + 經過translation後，每個source BB會產生對應的translated BB (TBB)並存在translation cache (TCache)中
  + translation process會去TCache中找source PC value
  + 如果BB已經轉好了，那它相對應的TBB會在TCache中，就可以拿到它在TCache上的address，DBT engine就會切換到execution process開始執行TBB
  + 執行完後就會回傳結果，並去找下一個source BB的TBB有沒有在TCache中
  + 如果有BB還沒轉好，那就會切到translation process做轉譯
  + 在TCache上會新增一個entry並將這個BB的TBB的起始address存到這個entry上
  + translation process開始做轉換直到遇到control flow instruction
  + 每道指令個別做解碼及轉譯, PC持續更新
* This article focus on the TCache component, which is part of the DBTOR, an in-house DBT system developed by authors that targets resource-constrained embedded systems.
* TCache is a translation buffer where the translated code is stored prior its execution and it must be allocated on a section of the host machine’s memory with write access and execution privileges.
* TCache 又稱為fragment cache，它讓BB跟TBB的位置有了關聯，記錄source PC跟target PC的對應關係, BB跟TBB的關係，它需要跟一般的source binary cache不同的feature
  + write and execution privileges

為了要能在TCache上儲存並執行轉換完的code，必須要有兩個條件

* + - TCache所占用的memory空間必須能重複寫入
    - TCache在執行上要有特權
  + Content Management

TCache需要一個data structure來管理及儲存TBB，有兩個目的

* + - 檢查被查詢的TBB是否已經轉譯
    - 回傳TBB所在的address

一旦hit，TBB的address必須要回傳給DBT engine並開始執行

若miss，則會傳miss給DBT engine來做轉譯

TCache的input為source PC，它作為unique key來找對應的TBB

* + Eviction Mechanism

當TCache滿了，它的content必須要被替換掉來儲存新的TBB，它可以部份或全部將TCache中的content替換掉，但部分替換需要考慮coherency enforcement，可能overhead會很大

* paper使用linked list跟hash table作為上述的data structure，並使用一個新的hardware-based的方法來一起做比較

Hardware-assisted Translation Cache

為了加速translated code management並得到具scalable的cache management的方法，會使用FPGA-based的hardware來做，另一篇paper已經有好的方法可以做，因為這個方法是non-intrusive且充分使用到硬體的好處，但它依賴software-based backup mechanism來處理missing translation，hash table要避免在resource-constrained embedded system上使用因為它的latency

TCache比較不像cache，比較像buffer，一般cache儲存已經確定的memory位置並且每個entry的大小都相同，TCache每個address存的source code並不是相同的表示法，每個轉換後的結果只有entry address可以存取且產生的大小不固定且無法知道，為了要在硬體上處理這個特徵，會需要複雜的方法去處理可用空間的管理，大部分TCache的overhead是在management上而不是data caching或memory access

approach4的修改版，改為不使用hash table，TBB caching, eviction policy, available space management用軟體的方法處理，新增TBB entry跟search TBB則是由輔助硬體處理，

硬體實作上是使用TCache memory space加上一個lookup table，在這個方法下CAM的output不會直接回傳給microprocessor，如果source address hit，那就將target address傳給microprocessor，如果miss則回傳0x00000000表示source address還沒做轉譯，???，使用一個valid bit來避免在第一次存取TCache或TCache reset後造成的false hit，雖然TBB的數量並不固定，但LUT entry卻是固定的，為了解決這個問題，會使用hash table去給超出LUT數量的TBB一個index，導致TCache miss都必須要計算hash key跟搜一遍hash table，即使LUT沒有滿

在這裡避免使用hash table的方法是調整新增entry進lookup table的機制，使用上有點像circular list，新的entry會依序寫進lookup table中直到index overflow後再從第一個entry重新寫入，造成那些在LUT中被覆蓋掉的entry會產生false miss，因為它們確實有存在TCache中但LUT卻找不到，!!!，此外舊的entry也較少會再重新使用，LUT的大小可以藉由一個參數做改動，但要能再對應的TCache size有充足的大小，