行政院國家科學委員會專題研究計畫　期中進度報告

子計畫四：用於低功率處理器之暫存器組織及運算資料產生(1/3)

計畫類別：整合型計畫
計畫編號：NSC92-2220-E-009-027-
執行期間：92 年 11 月 01 日至 93 年 07 月 31 日
執行單位：國立交通大學電子工程學系

計畫主持人：劉志尉
共同主持人：李鎮宜
計畫參與人員：林泰吉　林宏曄　劉晉宏　歐士豪　趙至敏

報告類型：完整報告
報告附件：出席國際會議研究心得報告及發表論文
處理方式：本計畫可公開查詢

中　華　民　國 93 年 5 月 26 日

行政院國家科學委員會補助專題研究計畫 □ 成 果 報 告
■ 期中進度報告

低功率系統之設計及自動化
子計畫四：用於低功率處理器之暫存器組織及運算資料產生 (1/3)

計畫類別：□ 個別型計畫　　■ 整合型計畫
計畫編號：NSC 92-2220-E-009-027
執行期間：92 年 11 月 1 日 至 93 年 7 月 31 日

計畫主持人：劉志尉
計畫參與人員： 林泰吉 林宏曄 劉晉宏 歐士豪 趙至敏

成果報告類型(依經費核定清單規定繳交)：□精簡報告 ■完整報告

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
■涉及專利或其他智慧財產權，□一年■二年後可公開查詢

執行單位：國立交通大學 電子工程學系

中華民國 93 年 5 月 19 日

摘要 Abstract

針對多媒體 SoC 中日益增加的即時運算需求，目前常見的解決方式主要有兩種：強化單一處理器整合運算能力或整合一個或多個數位訊號處理器(digital signal processor; DSP)。然而無論是加強微處理器的多媒體能力(如 MMX 指令延伸)或強化數位訊號處理器的控制能力(如 stack、control flow 及高速 context switch 等功能)，控制系統與數位訊號處理中資料利用的先天差異使得設計能同時滿足兩者的 memory hierarchy 的難度非常高。另一方面，市面上 DSP 處理器的設計考量都是單獨(standalone)的應用，使得在整合一個或多個 DSP 處理器的系統中，與微處理器存在功能重疊的重複部分。基於上述特性，我們保留雙處理器(dual processor)的架構將控制及運算分開處理，並進一步針對上述問題將其中的 DSP 處理器純化為一顆能夠自動處理大量資料運算的微型運算加速器。

我們採用與 Analog Devices Inc.的 ADSP-2181 類似的運算單元(ALU、multiplier unit 及 barrel shifter)，針對數位訊號處理中常見的演算核心，如 DCT、FFT、biquad filter、lattice filter、IIR filter 等，設計能執行高效率運算的運算引擎、記憶體架構及介面，並去除與微處理器功能重複的累贅部分，以期能達到面積小及快速運算的目標。對外的資料輸出入方面，搭載了 ping-pong mode 的緩衝區及符合業界標準的 AMBA AHB 介面，可以簡單的機制控制此核心及讀寫資料。我們提供針對上述運算加速器的微指令編譯器及模擬器。微指令可將使用者提供的高階演算核心，針對上述加速器架構最佳化，並轉換成可於其執行的微指令。模擬器部分則提供使用者於驗證不同階層之演算核心。

我們完成的數位訊號處理器核心可達到 314 MHz 的運算時脈(採用 CIC 所提供的 UMC 0.18um CMOS 製程及 cell-based 設計)，其 core size 為 1.5×1.5mm^2，消耗功率約 52mW。我們完成各設計階層所需的設計模型，包括完整的 physical 設計檔案，synthesizable Verilog RTL、cycle-accurate 之 SystemC model、Seamless CVE model 等。並將此加速器整合進 CIC 提供的 EASY (Example AMBA System)平台，在 EASY 平台完成一完整的 JPEG encoder 開發及驗證。

1. 簡介

Introduction

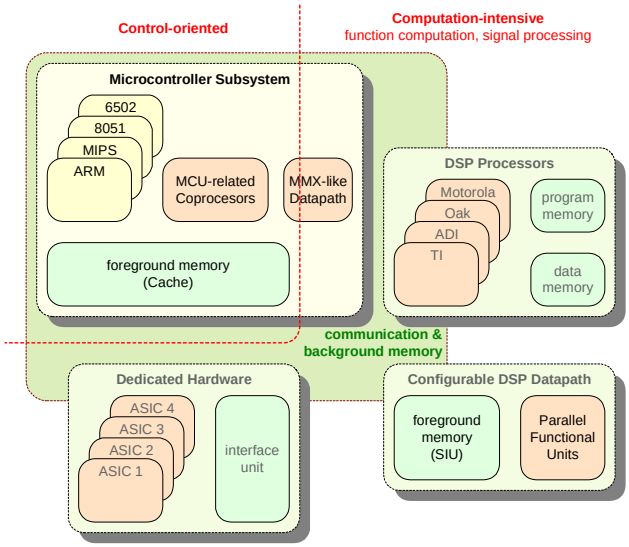


圖 一　異質性運算平台

現今的嵌入式系統擁有越來越豐富的多媒體功能及無線通訊支援，因此對於運算的需求也日益增加。在嵌入式系統的運算需求，可大致分為兩類：控制(control-oriented)的運算及資料計算(data-intensive)的運算。以常見的無線通訊手機為例，控制為主的運算包括有人機介面、作業系統及通信協定的處理等；資料處理為主的運算則包括有基頻處理(baseband processing)、數值轉換(transformation)，甚至聲音、影像的處理等等。對於上述日益增加的運算需求，目前常見加強運算的解決方案包括有使用單一加強處理器或整合一個或多個數位訊號處理器(DSP processor)。

單一加強處理器的解決方案包括有使用運算加強的 RISC 處理器及使用控制能力加強 DSP 處理器。運算加強的 RISC 處理器在原有的控制器上增強運算處理的功能，如單一時脈乘加器，SIMD (single instruction multiple data)指令支援等，但是實際上的數值運算效能仍遠不及 DSP 處理器。加強控制能力的 DSP 在原有的信號處理器上，再加上微處理器中常見的控制能力，如快速中斷處理、堆疊支援及 context switch 等功能，但使用上仍不及常見的微處理器有廣泛的作業系統支援。事實上由於控制及資料運算使用資

料的型態差異，設計能同時符合兩者的記憶體架構也是一大瓶頸。

整合 RISC 處理器及 DSP 處理器的解決方案將控制及資料運算兩種工作分開，各自分別處理，再以特定的機制彼此同步。但上述兩處理器的設計初衷都是單獨(standalone)的使用，兩者之前存在重複的部分。

基於上述特性，我們保留了雙處理器的架構，將控制部分留給 RISC 處理器，去除數位訊號處理器重複累贅的部分，純化為一個單純運算資料的數位信號處理器核心。並透過簡單的控制機制，迅速執行需要大量運算的演算法核心，如 DCT、FFT、linear filters 等。在此 DSP 處理器核心配備了 8KB 的 ping-pong mode 資料記憶體，可快速的輸出入資料並提供大量的運算資料給運算單元。並配備了業界標準的 AMBA Slave interface，資料輸出入、指令輸出入及控制訊號皆以存取記憶體方式控制，並便於系統整合。

為了進一步縮減 DSP 處理器核心，我們提出了靜態浮點運算(SFP; static floating point arithmetic)的概念，將數值以純小數的方式進行運算，類似浮點運算中的 mantissa，並以軟體靜態追蹤運算過程數值之指數，以近似定點運算單元的硬體複雜度，達到趨近浮點運算單元的精確度。

我們並完成了配合此精簡 DSP 處理器核心的工具程式組，包括有同步資料流圖(SDFG; synchronous data flow graph)模擬器、靜態浮點運算轉換、排程器(scheduler)及指令產生器。使用者可將欲執行的演算法以同步資料流圖表示，以上述工具進行模擬驗證的工作、轉換靜態浮點運算、排程，最後產生可以在此 DSP 處理器的執行的指令。

2. 硬體架構

Hardware Architecture

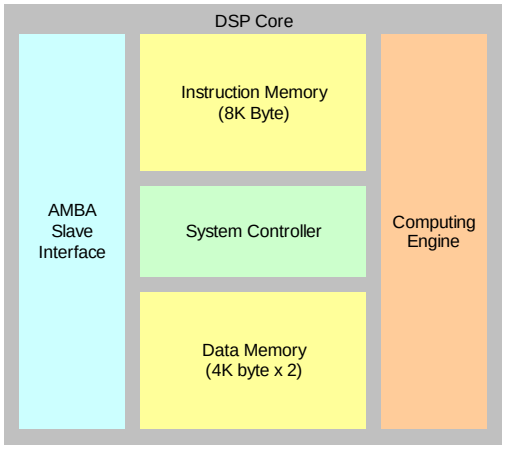


圖 二　硬體架構

如圖 二，提出的數位信號處理器核心包括下列五個主要部分：

- 運算單元 (Computing engine)
- 資料記憶體 (Data memory)
- 系統控制器 (System controller)
- 指令記憶體 (Instruction memory)
- AMBA 介面 (AMBA Slave interface)

分別以下列各節描述其特色及架構。

2.1. 運算單元 (Computing engine)

一般的數位信號處理器的設計理想是追求達到 dataflow rate，即處理器在執行時，所有功能單元(functional unit)都能處理數值運算。然而無法達到此理想的瓶頸在於：程式執行時的流程控制指令，如 branch、loop 等。在此運算單元中，我們在功能單元的輸

入端使用儲列來提供功能單元的運算來源；控制指令部分則整合至微指令中，降低因為執行控制指令需付出的資料運算代價。

在以儲列為基礎設計的運算核心中，主要包括兩種設計架構：output-queue system 及 input-queue system。Output-queue system 需要將 queue 中每個位置的資料回饋給功能單元；Input-queue system 則是需要運算輸出回饋給 queue 中的每個位置。無論是 input/output-queue system，都會有大量的控制訊號或是資料回饋造成佈局繞線上的困難。為了解決這個問題，我們的核心採用了混合式儲列架構，將每個功能單元之儲列改設置為可同時讀寫之 two-port memory，運算來源及運算結果由產生存取位址，並且利用軟體靜態分析方式來避免讀寫衝突的情況，達到近似 dataflow-rate 同時減低硬體佈局繞線的複雜度。

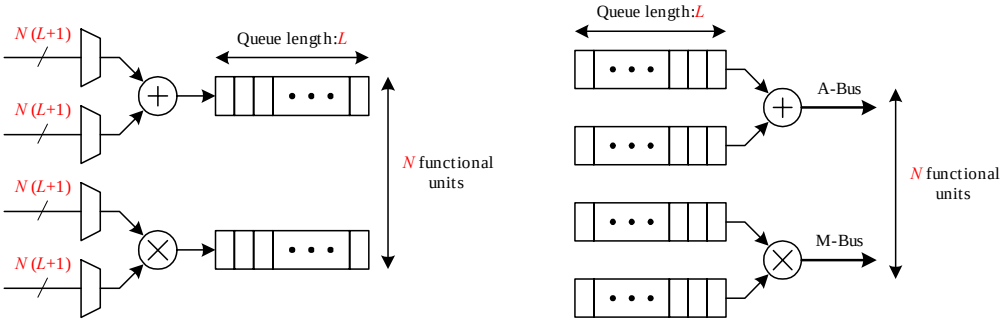
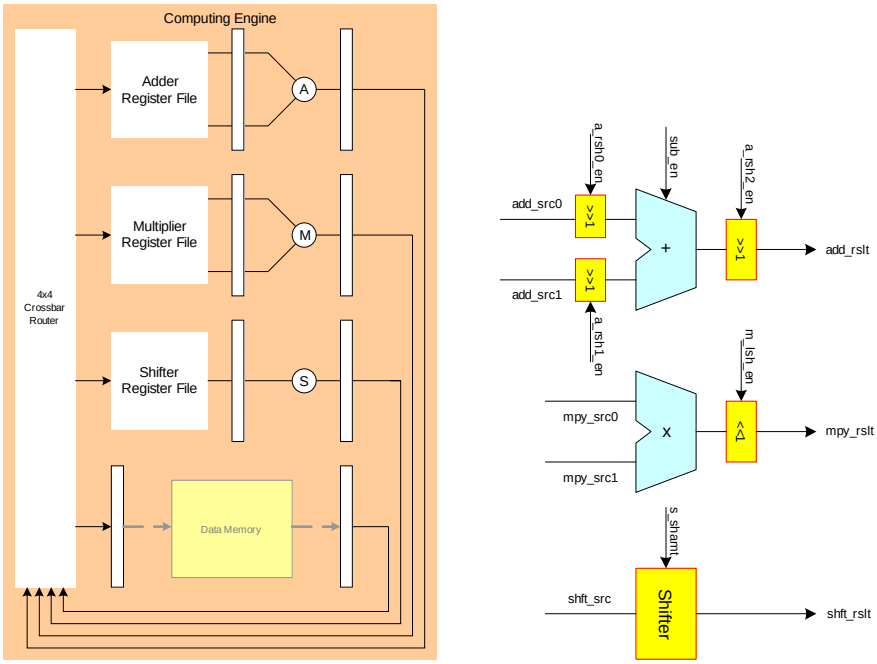


圖 三　(a) Output-queue;　(b) input-queue-based data generator

針對線性(linear)運算，運算單元包括四個主要的功能單元：加法器、乘法器、移位器以及輸出入單元。加法器負責執行加法動作，乘法器負責執行乘法動作，移位器執行數值的左移右移並支援正負號延伸(sign extension)，輸出入單元將運算來源從資料記憶體讀入並將結果輸出至資料記憶體。每一個功能單元配置了獨立的暫存器檔案(register file)，運算的結果透過 4x4 的 crossbar router 送至其他的功能單元進行接續之運算或輸出至資料記憶體，如圖 四(a)。為了進一步降低硬體複雜度，我們採用了靜態浮點運算單元，希望能夠以接近定點運算單元的硬體複雜度來達到近似浮點運算的精確度。因此，我們額外在加法器輸入端各配置了一位元之小數點對準裝置(aligner)及在輸出端配置了

一位元之正規化裝置(normalizer)，乘法器則是在輸出端配置了一位元之正規化裝置，相當於浮點單元中執行對準及正規的移位器的縮減形式，如圖 三(b)。其中所有的移位控制訊號，以及各個運算單元之控制、暫存器檔案(register file)位址、crossbar router 控制訊號等，由提供的軟體產生，並儲存至指令記憶體。



(a) (b)

圖 四 (a)運算單元架構; (b)靜態浮點運算單元

2.2. 資料記憶體 (Data Memory)

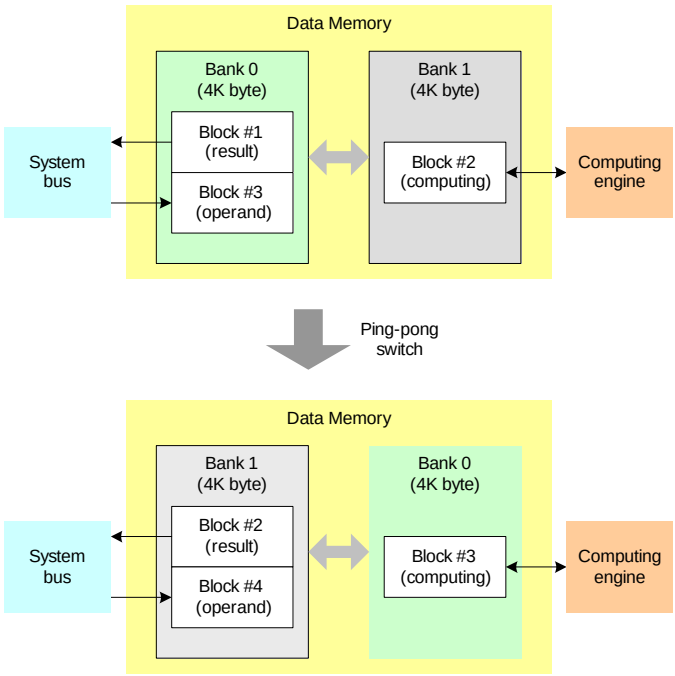


圖 五 Ping-pong mode 資料記憶體

處理器核心中另外搭載了 8K byte 資料記憶體，並分成兩組 4K byte ping-pong mode。兩組記憶體分別由外部匯流排(system bus)及內部的運算單元分別同時存取。透過控制機制，兩個 bank 之記憶體可交換彼此之存取對象。以圖 五為例，當運算單元使用 bank 1 資料記憶體運算 block #2 時, 外部 bus 同時讀出 bank 0 已運算完成的 block #1, 並寫入要即將運算的 block #3，完成運算以後，將 bank 0/1 交換，此時的 block #3 可由運算引擎存取並執行運算，而運算完成的 block #2 則由外部匯流排讀取，並寫入再下筆資料 block #4, 對於數位信號處理常見的 block processing 或 stream processing, ping-pong mode 可有效率的同時進行資料運算及存取的動作。

2.3. 系統控制器 (System Controller)

系統控制器主要為一個 FSM (finite state machine)，負責控制整個處理器核心之運作模式，包括五種模式：

- **IDLE**

  DSP 處於閒置狀態。此時外部匯流排可寫入指令，讀寫資料及控制記憶體。

- **PINGPONG_SWITCH**

  Ping-pong mode 交換狀態。此時產生資料記憶體對外及對內兩個 bank 的交換控制訊號，停留一時脈後，進入其他狀態。

- **RUNNING**

  DSP 處於執行運算的狀態。此時運算單元正執行運算，外部匯流排可以讀取已運算完成的資料，及寫入後續要計算的資料。

- **WAIT**

  DSP 處於等待外部存取的狀態。運算單元已完成預算，但外部匯流排仍在讀寫資料記憶體，等待外部匯流排讀寫完成後進入 PINGPONG_SWITCH 狀態。

- **EXCEPTION**

  DSP 的例外狀態。匯流排嘗試進行不合法的存取時，會跳至此狀態，此時需由外部重置(reset)數位信號處理器核心。

系統控制器包括兩組記憶體位址產生器，透過表 一所列之控制暫存器，系統可自動產生指令記憶體位址及資料記憶體位址，並執行所需的運算。其中位址記憶體特別針對 2D 轉換(TMODE)，提供 4X4、8X8 及 16X16 2D 轉換的位址產生。

表 一　控制暫存器

Control Register	Bit width	Comment
CL	9	Instruction length to execute
CSA	9	Instruction starting address
OBL	8	Number of operand per iteration
IOBSA	11	Operand starting address in data memory
IOBSAINC	5	Operand starting address increment per iteration
RBL	8	Number of result per iteration
IRBSA	11	Result starting address in data memory
IRBSAINC	5	Result starting address increment per iteration
ITRN	8	Number of iteration per block
BN	8	Number of block per data memory bank
DATA_VALID	1	Data valid signal
ACCESS	1	Bus accessing signal
TMODE	2	2D transpose mode signal
BUSY	1	Computing engine busy signal
EXCEPTION	1	DSP core exception signal

2.4. 指令記憶體 (Instruction Memory)

在處理器核心中搭載了 8K byte 的指令記憶體，對外可以經由 AMBA bus 直接將欲執行之指令寫入此記憶體，執行時由系統控制器產生執行位址輸出 128 位元寬之指令至運算單元執行。其中外部存取的資料寬度為 32 位元，但輸出的指令為 128 位元寬，故採用了四個 bank 分別儲存指令。對外部存取而言，每個 bank 都對應自己的位址，透過內建的位址解碼器，將 32 位元的資料分別儲存至四個不同 bank。使用者可經由我們所提供的軟體產生工具將所需要之演算法轉換成對應的指令。

2.5. AMBA 介面 (AMBA Slave Interface)

此處理器核心對外的介面採用業界標準的 AMBA Slave interface，如表 二，其中所有的存取及控制動作皆以讀寫記憶體的方式執行。

表 二　輸出入訊號

Pin name	Pin width	Direction	Comment
HCLK	1	Input	AMBA system clock
HRESETn	1	Input	AMBA reset signal active low
HADDR	14	Input	AMBA address
HTRANS	2	Input	AMBA type of transfer
HWRITE	1	Input	AMBA transfer direction signal
HWDARA	32	Input	AMBA write data
HRDATA	32	Output	AMBA read data
HREADYin	1	Input	AMBA transfer ready signal
HREADYout	1	Output	AMBA transfer ready signal
HRESP	2	Output	AMBA transfer response
HSEL_DSPlite	1	Input	AMBA slave select signal
im_MemGroupSel	1	Input	Instruction memory BIST signals
im_BistMode	1	Input	
im_BistFail	4	Output	
im_ErrMap	4	output	
im_Finish	1	output	
iob_MemGroupSel	1	input	Data memory BIST signals
iob_BistMode	1	input	
iob_BistFail	4	output	
iob_ErrMap	4	output	
iob_Finish	1	output	
SCI	1	input	Scan chain signals
SCO	1	output	
SCTM	1	input	
TM	1	input	

圖 六表示三大區塊：指令區塊、資料區塊及控制訊號區塊，各自對應的記憶體位置。指令區塊佔據 8KB 記憶體位址，直接由外部匯流排寫入將執行之指令。資料區塊佔據 4KB 記憶體位址，利用 Ping-pong 交換機制，將 8KB 記憶體分成兩塊，分別由外部匯流排及內部運算引擎存取。控制訊號暫存器均對應至特定位址，將外部寫入的數值轉換成控制資訊傳給系統控制器。

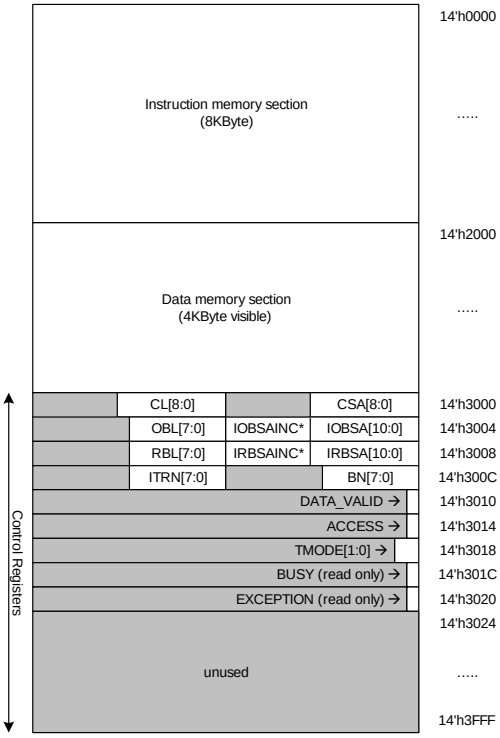

14'h0000
Instruction memory section
(8KByte)
.....
14'h2000
Data memory section
(4KByte visible)
.....
CL[8:0]
CSA[8:0]
14'h3000
OBL[7:0]
IOBSAINC*
IOBSA[10:0]
14'h3004
RBL[7:0]
IRBSAINC*
IRBSA[10:0]
14'h3008
ITRN[7:0]
BN[7:0]
14'h300C
DATA_VALID →
14'h3010
ACCESS →
14'h3014
TMODE[1:0] →
14'h3018
BUSY (read only) →
14'h301C
EXCEPTION (read only) →
14'h3020
14'h3024
Control Registers
unused
.....
14'h3FFF
*IOBSAINC[4:0]，IRBSAINC [4:0]

圖 六 記憶體位址

3. 靜態浮點運算

Static Floating Point Arithmetic

為了進一步精簡此數位訊號處理核心，我們提出了靜態浮點運算，以期能以定點運算單元的硬體複雜度，達到趨近浮點運算的精確度。

3.1. 簡介

在多媒體應用中，數值的範圍及精確度代表聲音、影像訊號的品質表現，然而在有限位元的數位系統中，範圍及精確度卻無法兼顧，為了提供大的動態範圍，則必須犧牲數值的精確度。因此如何在系統的運算過程中，防止溢位並維持足夠的精確度，是個重要的問題。在常見的數值表示方法中，浮點運算在數值的精確度及範圍有非常優異的表現。浮點運算將數值分成三個部分表示：(1)sign (2)exponent (3)mantissa。sign 表示數值的正負，exponent 表示數值的指數部分，數值則動態地正規化(normalize)至純小數(fractional number)儲存在 mantissa 部分，提供固定長度的精確度。如此的表示方法可以提供相當大的動態範圍，並且固定每筆數值的精確度。以 IEEE754 標準的單精確度表示法為例，可提供 2^{-128} 至 2^{127} 之動態範圍，並維持每筆數值 24 位元有效精確度，如圖 七。

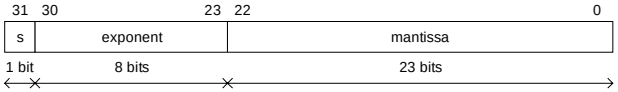


圖 七　IEEE754 單精確度浮點數

然而，浮點數值動態儲存每筆資料的 exponent 及 mantissa，使執行運算的硬體變的相當複雜。圖 八描述了浮點加法及乘法的運算流程。，以浮點的加法為例，在執行 mantissa 部分相加之前，必須將 mantissa 部分進行移位使兩數值之 exponent 對齊，完成相加以後必須將結果之 mantissa 再度正規至純小數範圍。無論是乘法或加法，浮點數值的運算都需要動態判斷數值大小，並依此執行移位動作，使得浮點數運算單元的硬體相當複雜且耗費能量。

Floating-point Addition
1. **Shift** the mantissa to align the exponent part of two numbers
2. Add the mantissas
3. **Normalize** the sum, checking for overflow or underflow
4. Round the sum

(a)

Floating-point Multiplication
1. Add the exponent without bias
2. Multiply the mantissa
3. **Normalize** the product, checking for overflow or underflow
4. Round the product
5. Obtain the result's sign

(b)

圖 八　浮點數運算

對於以市電供應電源的多媒體系統而言，浮點運算單元無疑是個不錯的選擇，可提供相當大的動態範圍及固定精確度。但是對於使用電池的多媒體嵌入式系統來說，產品體積重量及電池壽命決定了產品的價值，硬體複雜且耗電之浮點運算單元在此並不適用。目前常見的解決方案是採用定點運算單元。定點運算單元較簡單的硬體及較低的能量消耗較能符合可攜式系統的對於體積重量及電池壽命的需求。但是為了在避免在運算過程中，可能會出現溢位而造成錯誤，設計者必須事先估計運算過程中間數值的可能範圍，並將輸入數值移位縮小(scale down)，預留運算過程中可能的溢位位置。以 256 灰階的圖形數值通過一 7-tap FIR 對稱係數濾通器 [-0.0645, -0.0407, 0.4181, 0.7885]為例，若採用 24 位元之定點運算單元，保留 8 位元給輸入數值及 3 位元給 FIR 濾通器的累進避免溢位，則係數至多可以 13 位元表示之。將原係數乘上 2^{13} 並取捨至整數成為[-528, -333, 3425, 6459]，完成運算以後，再將運算結果向右移位 13 位元將數值調整至原數值範圍。但在縮小輸入數值的同時，也犧牲了數值運算的精確度，若要較高的精確度，則必須由設計者在運算過程中額外增加多個縮小的動作。

在此提出的新型靜態浮點運算，是一種介於浮點數運算及定點數運算的有效折衷解決方案。將數值以純小數(fractional number)的方式儲存，類似浮點數中的 mantissa 部分；並利用靜態方法數值的範圍並追蹤 exponent。可以接近定點運算單元的硬體複雜度，達到近似浮點運算的精確度。

3.2. 靜態浮點運算單元

針對線性運算，我們的靜態浮點運算單元配置了(N+1)位元小數(fractional)加法器、一小數乘法器及一移位器，如圖 九。其中加法器支援二補數(two's compliment)的小數加減法，且在輸入端設置了一位元的右移器執行數值的對準動作(alignment)，相當於浮點加法器中小數點對齊裝置的縮減形式，同樣的，在加法器的輸出端一樣配置了一位元的右移器，執行運算結果的正規動作，相當於浮點加法器中的正規化裝置的縮減形式。同時採用多一位元的小數加法器，可以維持數值在移位後的精確度。乘法器支援二補數的小數乘法，在輸出端部分配置了一位元左移器，相當於浮點乘法器中正規化裝置的縮減形式。移位器的部分可以執行任意的左右移位，並支援數值移位。

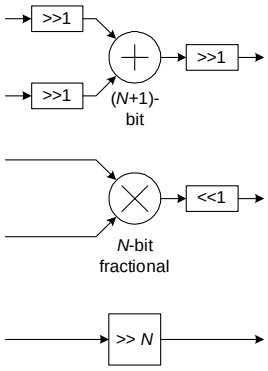


圖 九　靜態浮點運算單元

圖 十比較了(N+1)位元加法器及 N 位元乘法器兩者的差異。針對其中一輸入對準移位器啟動的狀態，N 位元加法會先棄置一位元再進行加法，(N+1)位元加法器則是保留此位元進行加法，再將結果進行取捨的處理。

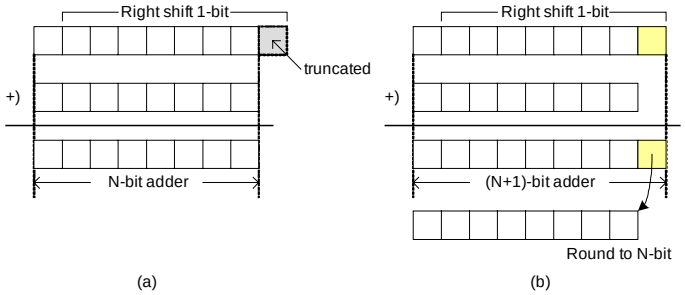


圖 十　(a)N 位元加法器; (b)N+1 位元加法器

圖 十一表示純小數乘法的運作情形，所有的運算元皆為 1　　-1 的數值，相乘後的結果也維持在相同範圍，小數點也保持在同樣的位置，不會產生溢位的情況。

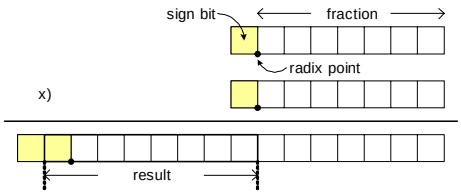


圖 十一　純小數乘法

以線性運算為例，浮點運算單元至少需要一浮點加法器及一浮點乘法器；定點運算單元則需要一整數加法器、一整數乘法器及一移位器。靜態浮點運算單元在硬體上的複雜度相當接近定點運算單元，僅多了縮減至一位元移位器的小數點對齊裝置及正規化裝置。

3.3. 靜態指數追蹤

使用者將提供的浮點演算法核心以同步資料流圖(SDFG, synchronous data flow graph)表示，並分析圖上每一個邊線(edge)上可能的最大值，藉此安插運算中所需的對準及正規動作，轉換成靜態浮點運算型態之同步資料流圖，再進一步將靜態浮點運算資料流圖對應至上述靜態浮點運算單元，產生靜態浮點運算單元之控制訊號。在我們的實現中，以線性運算的同步資料流圖為例，包括了下列運算節點：(1)加法節點：執行加法動作，有二輸入邊線及一輸出邊線，(2)乘法節點：執行係數乘法動作，有一輸入邊線、一組輸出邊線，(3)輸入端點：執行外部資料的輸入，有一輸出邊線，(4)輸出端點：執行運算資料的輸出，有一輸入邊線。

3.4. 峰值向量分析(PEV Analysis)

同步資料流圖中包括了邊線(edge)及節點(node)，邊線代表著數值資料的傳遞，節點

則代表了資料的處理及運算。分析使用者提供的同步資料流圖，並估計每邊上數值可能的最大值，並以此來安插靜態浮點運算中所需要的對準及正規動作。

將同步資料流圖上的每個邊線關連一個峰值向量**[*M*, *r*]**，其中 M 代表此邊上可能出現的最大量值(magnitude)，r 代表此邊上小數點的位置(radix)，相當於浮點數的 exponent。為了提高位元使用率，同步資料流圖上所有的 M 均被限制在 0.5~1 之間，並以下列規則得到每個邊線上的峰值向量：

- 加法節點的兩個輸入邊線峰值向量之 r 值必須相同
- $[M_1, r_1] \times [M_2, r_2] = [M_1 \times M_2, r_1 + r_2]$
- ***M*** 與 ***r*** 具有下列運算關係: "***M*** divided (multiplied) by 2 when ***r*** minus (plus) 1"

圖 十二為峰值向量在加法及乘法的運算範例。在進行加法運算之前必須將輸入端峰值向量之 ***r*** 值調整至相同，故將上方輸入端峰值向量的 ***M*** 除二、***r*** 減一。兩者 ***M*** 值相加後，再將 ***M*** 值調整在 0.5~1 之間。

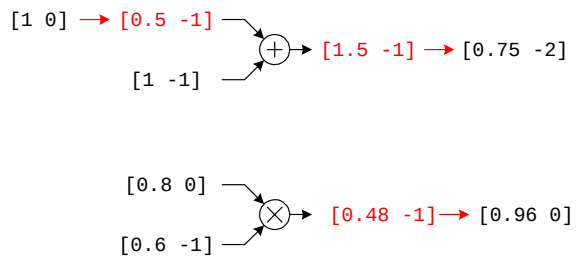


圖 十二　峰值分析範例

3.5. Affine Form 峰值估計

峰值分析僅針對每一節點的輸出入邊線作分析，並無考慮邊線彼此之間的資料關連性(correlation)，因此在估計範圍時，可能會造成過度估計(over estimation)而降低運算的精確度。如圖 十三，計算右邊加法節點時，原估計方法無法將兩輸入邊線之相關項 ***y***

列入估計範圍，因此造成最後結果的過度估計，降低此邊線上的位元使用率。

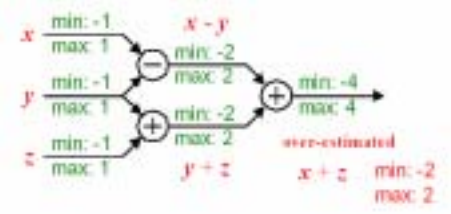


圖 十三　峰值過度估計範例

為解決此過度估計的問題，我們導入 affine form 使估計範圍能更為準確。每個邊線上，除了原有的峰值向量，再關連一組 affine form 向量，記錄每個輸入值在此邊線上的權重(weight)。

$$[w_0 \quad w_1 \quad \Lambda \quad w_n]$$

其中 $w_0 \ldots w_n$ 代表第 $0 \ldots n$ 個輸入端點在此邊線上所佔的比例，其中 w 值可為負值，紀錄此輸入端點經過的運算。而原來的 M 值可由 affine form 向量以下列算式得知。

$$M = \sum_i |w_i|$$

運算加法(減法)節點時，輸出端的 affine form 向量為輸入端的兩組 affine form 向量相加(相減)，乘法節點則是將 affine form 向量直接乘上係數。affine form 向量具有與 M 值相同的運算特性。

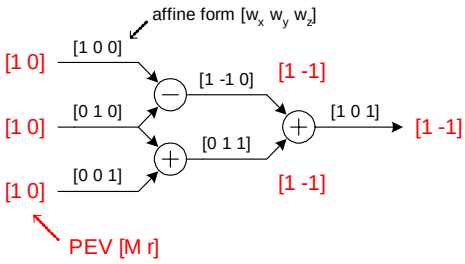


圖 十四　affine form 峰值分析

圖為 affine form 峰值分析之範例，其中右方加法節點中的 y 項，在 affine form 運算中已經消去，得到正確的峰值分析

3.6. 移位安插

完成同步資料流圖的峰值分析以後，將產生的對齊及對準動作對應至硬體上的移位器。以下圖為例，A+B 之前的對準動作為一位元右移，而結果 C 必須正規至 0.5~1 的範圍內，正規動作為一位元右移。

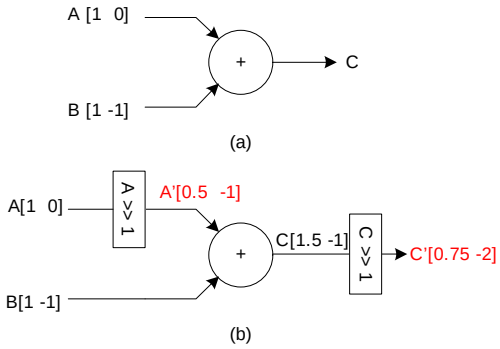


圖 十五　移位安插

4. 軟體產生流程

Software Generation

針對此數位訊號處理核心，我們提供完整的工具程式，包括下列四個程式：

- 同步資料流圖模擬器

  SDFG Simulator

- 靜態浮點運算轉換器

  FP (floating point) to SFP (static floating point) Converter with saturation arithmetic

- 以整數線性規劃為基礎之排程器

  ILP-based Scheduler

- 指令產生器

  Instruction Generator

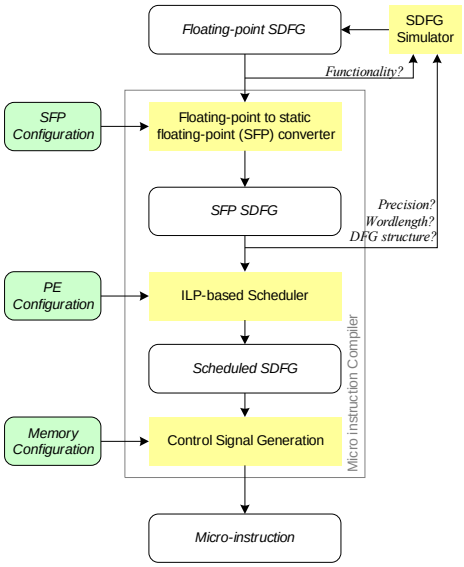


圖 十六　軟體產生流程

4.1. 同步資料流圖模擬器

使用者完成以同步資料流圖描述的演算法之後，可以以提供的模擬器進行功能正確性的驗證工作。使用者將完成的同步資料流圖及要運算的數據輸入程式，輸出經由演算法運算的結果。提供的模擬器支援單精確度、倍精確度的浮點數運算，及位元正確(bit-true)之靜態浮點運算並支援多種資料位元長度。

4.2. 靜態浮點運算轉換

使用者輸入以同步資料流圖描述的浮點數演算法，可輸出轉換成支援靜態浮點運算的同步資料流圖。輸入同步資料流圖以後，程式進行靜態浮點分析，依據峰值向量分析及靜態追蹤指數(exponent)的結果，自動安插所需的移位動作，輸出完成的同步資料流圖。

4.3. 以整數線性規劃為基礎之排程器

完成靜態浮點運算轉換後，必須將同步資料流圖對應至硬體執行。但是硬體的運算資源是有限的，必須將同步資料流圖做時間上的分配，才能順利的執行運算。我們利用整數線性規劃的方式，將排程上的各種限制以不等式的方式呈現，最後再以專門解整數線性規劃(ILP; integer linear programming)的程式求解，得到時間上的排程。

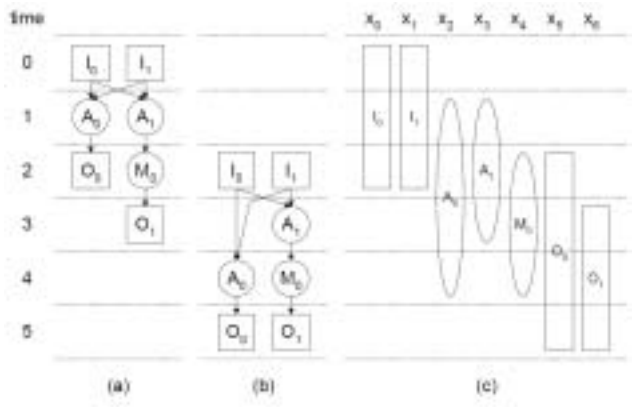


圖 十七　排程範例 (a)ASAP; (b)ALAP; (c)scheduling range

首先先將同步資料流圖進行 ASAP 及 ALAP 的排程。ASAP 排程是不顧硬體資源的限制，僅考慮資料相依性(dependency)的問題，盡量早執行每個節點。如圖 十七(a)，所有的節點只要運算元準備完成立刻執行。ALAP 排程是指在給定的時間限制之內，不顧硬體資源的限制，僅考慮資料相依性的問題，盡量晚執行每個節點。如圖 十七(b)，在給定第五個時間單位內要將所有的節點執行完畢，且盡量的晚執行。完成 ASAP 及 ALAP 的排程後，可以得知每個節點的排程範圍(scheduling range)。以圖 十七(c)為例，A_0 節點在 ASAP 排程結果在第一個時間點執行、ALAP 排程結果在第四個時間點執行，故我們可得知 A_0 節點的排程範圍在第 1~4 的時間點，換言之，在五個時間要完成此同步資料流圖的限制之下，A_0 節點可排程的位置可在第 1、2、3 及 4 個時間點。

在此我們導入一個變數 $x_{i,j}$，其中 i 代表節點，j 代表排程在第 j 個時間點，則 $x_{i,j}$ 可為 0 或 1 的數值。$x_{i,j}$ 為 1 代表決定第 i 個節點排程在第 j 個時間點，為 0 則代表第 i 個節點**不排程**在第 j 個時間點。我們利用下列四種限制及前述排程範圍，可列出不等式，最後求解出的 $x_{i,j}$ 則代表最後的排程結果。

運算資源限制

同一時間點，可執行的相同節點數，不得超過硬體所提供的硬體資源。以圖 十七(c)的輸入節點為範例，若硬體上僅有一個輸入單元，則可得下列不等式：

$$x_{0.0}+ x_{1.0} \le 1;\ x_{0.1}+ x_{1.1} \le 1;\ x_{0.2}+ x_{1.2} \le 1$$

單一位置限制

同一個節點，只能在某一個時間點執行一次。以 I_0 為例，必定只能在第 0~2 時間點選擇一個時間點執行。以不等式表示：

$$x_{0.0} + x_{0.1} + x_{0.2} = 1$$

相依性限制

節點的順序必須依照資料的相依執行。

$$x_{0.0} + 2\,x_{0.1} + 3\,x_{0.2} - 2\,x_{2.1} - 3\,x_{2.2} - 4\,x_{2.3} - 5\,x_{2.4} \leq -1$$

記憶體輸出入限制

硬體上的 register file 僅有單一寫入埠，不得有兩筆資料同時寫入同個 register file。以加法器為例：

$$x_{0.0} + x_{1.0} \leq 1;\ x_{0.1} + x_{1.1} \leq 1;\ x_{0.2} + x_{1.2} \leq 1;$$

以上述四種限制，可以列出將同步資料流圖在運算單元上執行的所有限制不等式，再利用專門解 ILP 的程式(Lindo)解出一組解，即為可行之排程。

4.4. 指令產生器

將排程好的同步資料流圖，再以此程式直接對應成硬體上的控制訊號，並產生存取暫存器檔案所需的位址，最後產生此同步資料流圖對應的指令。

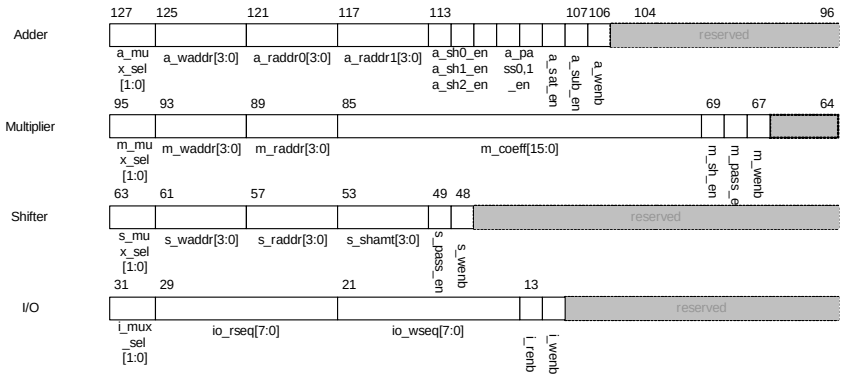


圖 十八　指令格式

5. 模擬結果及硬體實現

Simulation Result and Hardware Implementation

5.1. 靜態浮點運算模擬結果

我們將靜態浮點運算結果與浮點運算結果相比較，以 IJG (Independent JPEG Group)’s JPEG source code 中的 8X8 DCT 為比較對象，16 位元長度的靜態浮點運算就可以超過 32 位元整數運算的精確度。其中靜態浮點運算包括了 affine form 的分析。就運算所需時脈而言，浮點運算單元皆假設單一時脈可完成一組加法及乘法，也無須額外的移位動作。16 位元定點運算需要較多的移位動作來維持精確度，故完成運算所需時脈較多。而靜態浮點運算所需要的時脈數則介於兩者之間。

表 三　靜態浮點運算結果

Algorithm Kernel	Cycle count	PSNR (dB)
Single precision FP	672	--
16-bit fixed-point integer	848	33.2220
32-bit fixed-point integer	672	36.0981
16-bit SFP	**688**	**38.1165**
24-bit SFP	**688**	**62.1439**

5.2. 運算效能結果

我們針對常見的演算法核心進行效能的實測。並與具有相近運算單元(一個加法器、一個乘法器及一個移位器)的 Analog Device Inc.的 ADSP-218x 系列數位信號處理器比較其運算之時脈數。可注意到在資料相依度較低的工作上，我們的處理器核心可以達到更高的平行度及效能。

表 四　提出之數位信號處理核心及 ADSP-218x 之效能比較

Algorithm Kernel	ADSP-218x (< 160MHz)	Proposed DSP Core (314MHz)
3rd order lattice filter	32	12
2nd order biquad filter	13	16
16-point complex FFT	874	268
8-point 1-D DCT	154	43
8X8 2-D DCT	2,452	688

5.3. 硬體實現結果

我們將提出的 DSP 處理器核心，以 CIC 提供的 UMC 0.18um CMOS 製程實現。

表 五　晶片規格

Technology	UMC 0.18um 1P6M CMOS
Core size	1.5 x 1.5 mm^2
Transistor/Gate Count	197655
Power dissipation	52 mW
Max. frequency	314 MHz
On-chip memory size	16KB (8KB data / 8KB instruction)

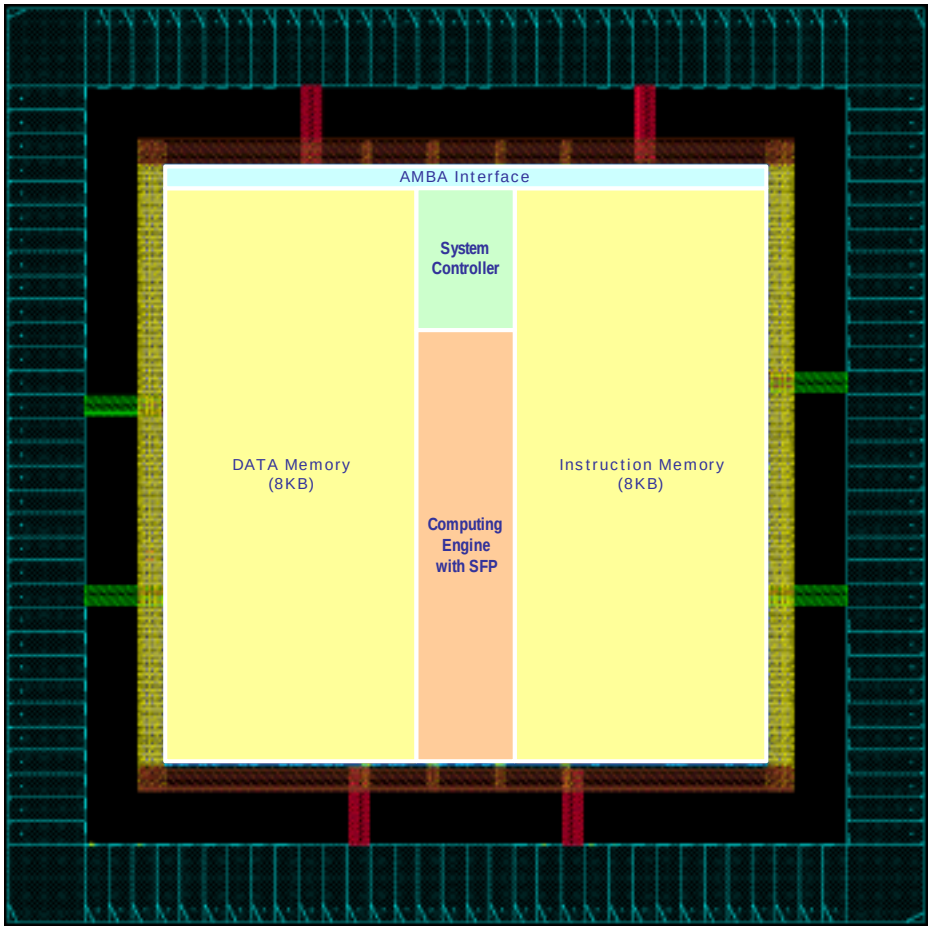


圖 十九　佈局圖

6. 結論

Conclusion

在本計畫中，我們針對多媒體嵌入式系統中常見雙處理器架構，去除其中數位信號處理器與 RISC 處理器重複累贅部分，並純化為一顆單純處理資料的數位信號處理核心，透過非常簡單的控制機制，能有效率運算多媒體應用需要大量運算的演算法，如：DCT、FFT、linear filters 等。與相同運算資源(一加法器、一乘法器及一移位器)的 Analog Device Inc. ADS-218x 處理器相比較運算時脈數，我們的 DSP 核心最高可達到將近 3 倍的效能。

為了更進一步縮減此數位信號處理核心，我們提出了靜態浮點運算，以接近於定點運算單元的硬體複雜度，達到接近浮點運算的精確度。以 8×8 DCT 為例，24 位元靜態浮點運算之 PSNR 達到了 62.1439 dB。我們提供了完整的工具程式組，提供使用者驗證其提供的演算法核心，及轉換成可在我們數位訊號處理核心執行的指令，包括有：同步資料流模擬器、靜態浮點運算轉換、排程器、指令產生器等四套工具。

我們以 CIC 提供的 UMC 0.18um CMOS 製程完成此數位訊號處理核心的硬體實現，以 cell-based design flow，可達到 314 MHz 的時脈，core size 為 1.5×1.5mm^2，消耗功率約 52mW。並完成各設計階層所需的設計模型，包括完整的 physical 設計檔案，synthesizable Verilog RTL、cycle-accurate 之 SystemC model、Seamless CVE model 等。並將此加速器整合進 CIC 提供的 EASY (Example AMBA System)平台，在 EASY 平台完成一完整的 JPEG encoder 開發及驗證。

7. 參考資料

Reference

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture – A Quantitative Approach*, 3rd Edition, Morgan Kaufmann, 2002

[2] *Digital Signal Processing – Using the ADSP-2100 Family*, Analog Device Inc., 1990

[3] *IEEE Standard for Binary Floating-Point Arithmetic*, IEEE Standard 754, 1985

[4] K. K. Parhi, *VLSI Digital Signal Processing Systems – Design and Implementation*, John Wiley & Sons, 1999

[5] F. Fang, R. Rutenbar, M. Puschel, and T. Chen, "Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling," in *Proc. DAC*, 2003

[6] *LINDO API User's Manual*, LINDO System Inc., 2002

[7] *Independent JPEG Group*, http://www.ijg.org

[8] G. A. Constantinides, P. Y. K. Cheung, W. Luk, "Synthesis of saturation arithmetic architectures ," *ACM Trans. Design Automation of Electronic Systems*, July 2003