INFLUENCE OF FIXED-POINT DSP ARCHITECTURE ON COMPUTATION ACCURACY

Daniel Menard, Philippe Quemerais †
† LASTI - University of Rennes I
6 Rue de Kerampont
22300 Lannion, FRANCE
{name}@enssat.fr

Olivier Sentieys †‡
‡ IRISA/INRIA
Campus de Beaulieu
35042 Rennes cedex, FRANCE
sentieys@irisa.fr

ABSTRACT

The minimization of cost, power consumption and development time of DSP applications requires the development of methodologies for the automatic implementation of floating point algorithms in fixed point architectures. In this paper, the influence of the DSP architecture on the computation precision is analyzed. The necessity of taking into account the DSP architecture model in the data coding process is shown. Then, a new methodology for the implementation of algorithms into fixed-point DSPs is defined.

1 INTRODUCTION

Most digital signal processing algorithms are implemented in fixed-point architecture in order to satisfy the cost and power consumption constraints of embedded systems. The reduction of the time-to-market of applications requires high-level development tools that allow the automation of some tasks. Nevertheless, the manual transformation of floating-point data into fixed-point data is a time-consuming and error prone task. Indeed, some experiments [1] have shown that this manual conversion can represent up to 30% of the global implementation time. Thus, methodologies for the automatic transformation of floating-point representation into a fixed-point representation have been proposed [2, 3].

For Digital Signal Processor (DSP), the aim of the methodology is to define the optimal fixed-point data formats which maximize the precision and minimize the size and the execution time of the code. Existing methodologies [2, 3] achieve a floating-point to fixed-point transformation leading to an ANSI-C code with integer data types. Nevertheless, the different elements of the target architecture are not taken into account for the fixed-point data coding.

In this paper, the influence of the fixed-point DSP architecture on the computation precision is analyzed. This study underlines the necessity of taking into account the DSP architecture for optimizing the data coding. Firstly, a review of the different elements of the DSP data path, able to influence the computation precision and the recent evolutions are detailed. Then, the approach followed for analyzing the influence of the DSP architecture on the computation precision and the results obtained for different applications are presented. Finally, a new methodology for the implementation of a floating point algorithm into a fixed-point architecture is introduced.

2 DSP ARCHITECTURE

DSP architectures are designed to compute efficiently the arithmetic operations involved in digital signal processing applications. Different elements of the data path influence the computation precision. Each processor is defined by its native data word-length which is the word-length of the data that the processor buses and data path can manipulate in a single instruction cycle [4]. For most of the fixed-point DSPs, the native data word-length is equal to 16 bits. For ASIP or some DSP cores, this native data word-length is customizable in order to fit better the architecture to the target application.

Most of the DSPs allow the achievement of a multiply accumulate (MAC) operation without lost of information, by computing the operation in double precision. The word-length of the adder and the multiplier output is equal to the double of the native data word-length. Nevertheless, the increase of data dynamic range due to successive accumulations can lead to an overflow. Thus, some DSPs [5, 6] extend the accumulator word-length by providing guard bits. These supplementary bits allow the storage of the additional bits generated during successive accumulations.

The number of storage elements present in the data path influences the computation precision. Indeed, some data are spilled into memory when any data path register is free for storing these intermediate data. For limiting the execution time of the data transfer, these data are stored in memory in single precision and thus a cast operation is achieved. In conventional DSP architecture [5, 6, 7], the limited number of accumulator registers can lead to an important number of spill operation.

In order to decrease the execution time of the code, some recent DSPs like the TMS320C64x [8] and the TigerSharc [9], allow the exploitation of the data-level

parallelism by providing SWP (Sub-Word Parallelism) capacities. An operator (multiplier, adder, shifter) of word-length N is split in order to execute k operations in parallel on sub-word of word-length N/k. Thus, these processors can manipulate a wide diversity of data types (8, 16, 32, 40, 64 bits). Nevertheless, the use of SWP instructions can require to achieve the computation in single precision[1] in order to compute the same number of operations on the multiplier and the adder.

Given that the data word-lengths in a DSP are limited, scaling operations are required in order to maintain the maximal precision. Different kinds of shift register are available for scaling operations. For some processors [7, 6], a specialized shift register is situated at the output of the multiplier and several specific shifts can be achieved. When this kind of shift register is present, the output multiplier can be scaled without supplementary cycles. For more flexibility, most of the recent DSPs offer a barrel shifter performing any shift operation in one cycle. For VLIW DSP [9, 8], the architecture is homogeneous and the barrel shifter can scale the output of a multiplication or an addition. Nevertheless, in MAC based architecture [5, 6, 7], this kind of shifter is connected to the accumulator register and can only scale efficiently the output of an addition.

When a scaling operation occurs, the quantization mode used by default is the truncation. But this process leads to a non-centered quantization noise. Thus, some DSPs [5, 9] provide a rounding mode in order to eliminate this bias.

3 EXPERIMENTS

The most commonly used criteria for evaluating the precision of a fixed-point implementation is the Signal to Quantization Noise Ratio (SQNR). The architecture influence on the computation precision has been studied through the comparison of the output SQNR resulting of the implementation of several digital signal processing algorithms in different DSPs. Firstly, the approach for fixed-point data coding in the case of FIR and IIR filters is described. Then, the approach for precision evaluation and the different quantization noise sources are presented.

3.1 Fixed-point data coding

A fixed-point data is made up of an integer part and a fractional part. The number of bits required for the integer part is defined from the dynamic range of the data in order to avoid the occurrence of an overflow. The data dynamic range is obtained from the l_1 norm [10]. The diversity of data dynamic range requires to scale some data in the algorithm in order to maintain a sufficient precision. If the dynamic range of the data increases, supplementary bits for the integer part are required in order to avoid an overflow. Different alternatives can be considered for introducing these supplementary bits for MAC operations. For some DSPs the word-lengths of the adder and the accumulator register are greater than the word-length of the multiplier output. The presence of these guard bits within the accumulator allows the storage of the supplementary bits issued of successive accumulations. Otherwise, if the multiplier output can be shifted, a scaling operation can be inserted between the multiplication and the addition *(internal scaling)*. For processors without guard bits or the capacity of scaling the multiplier output, the scaling is achieved on one of the multiplier inputs. Firstly, the filter input can be scaled in order to insert the supplementary bits *(External scaling)*. Otherwise, an alternative is to scale the filter coefficients by introducing the supplementary bits during the coefficients coding. For this last technique no cycle is wasted for scaling the data but the frequency response of the filter is modified.

[1] the multiplier input and output word-lengths are identical

3.2 SQNR computation

The SQNR is defined as the ratio of the signal power to the quantization noise power. This quantization noise present at the output of the system corresponds to the difference between the outputs of the fixed-point and floating-point versions of the system. For the different applications, the SQNR has been computed with our methodology presented in [11]. This method determines automatically the analytical expression of the signal and noise power at the output of a linear system. The output quantization noise is modelized as the sum of the output noises due to the quantization of the coefficients and the noises b_α due to the propagation in the system of the noise sources b'_α presented in the next section. The determination of the frequency response of the transfer functions between the system output and each noise sources b'_α is required for computing the statistical parameters of the noise b_α. These transfer functions are automatically computed from the Signal Flow Graph (SFG) of the system.

3.3 Noise sources

The elimination of some bits during a cast operation leads to a quantization noise. The statistical parameters of this quantization noise are defined from a noise model [12] according to the number of bit eliminated,

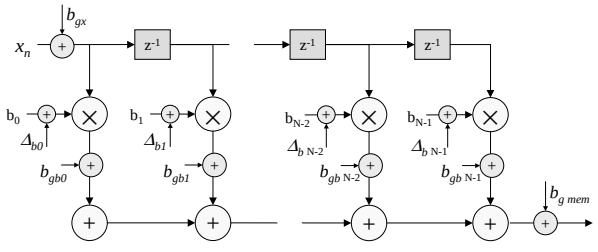

Figure 1: FIR filter noise model

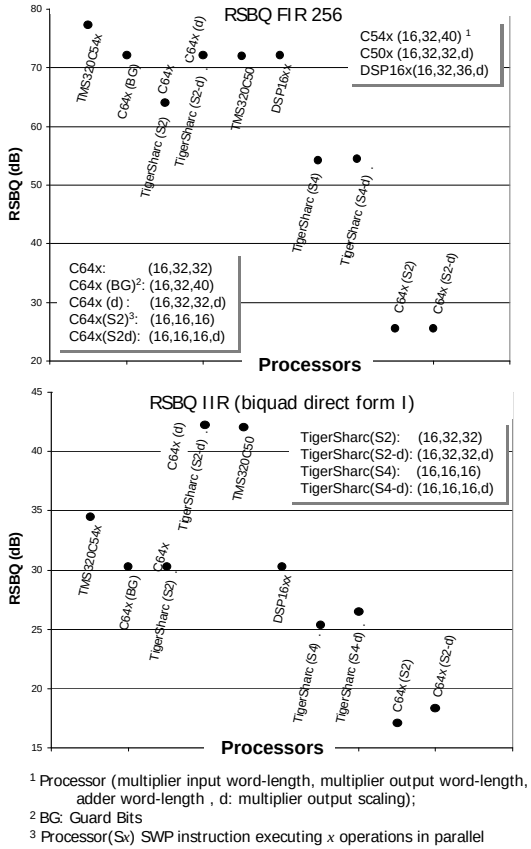

Figure 2: Output filter SQNR evolution

the format of the data and the quantization mode used. The SFG of a FIR filter is given at figure 1 and the different potential noise sources b'_α are reported in this SFG and detailed below. The noise b_x is made up of two noises b_{qx} and b_{gx}. b_{qx} corresponds to the quantization noise associated with the input and the noise b_{gx} is generated if the input x is scaled during an *external scaling*. The noise b_{gmi} present at the output of the multiplier is made up of two noises b_{gmsp} and b_{gmd}. The noise b_{gmsp} is generated if a single precision multiplier as in SWP instructions is used. The scaling of the multiplier output during an *internal scaling* leads to the quantization noise b_{gmd}. The quantization error associated with each coefficient c_i is Δ_{c_i}. Finally, b_{gmem} is the noise due to the cast operation required for storing the filter output in memory.

4 EXPERIMENTS RESULTS

In this section the results of the experiments done on several digital signal processing algorithms are analyzed. The SQNR obtained for the implementation of FIR and IIR filters into several DSPs are presented in the figure 2. The input and output of the filter are stored in memory on 16 bits.

High performances can be obtained with guard bits (TMS 320C54x) for successive accumulations like in FIR filters. However, these guard bits can not be exploited in IIR filters since the dynamic range of the adder inputs and output are closed. In this case, a scaling of the filter input is necessary (C54x, C64x(BG)). The scaling of the multiplier output leads to good results for both kinds of filter (C64x (d), TigerSharc (S2-d)). For a processor without guard bits or the capabilities to scale the multiplier output, the filter input or the coefficients must be scaled. In both cases, the performances obtained in double precision are definitely inferior since the scaling operation is done before the multiplications. The decrease of performance due to the single precision computation with SWP instructions, is important for a truncation quantization mode (C64x (S2)). However, for rounding quantization mode (TigerSharc (S4)) this decrease of performance is definitively lower. This kind of instruction can significantly reduce the code execution time.

For VLIW processors (C64x, TigerSharc), the results of an implementation without multiplier output scaling (MOS) has been tested since it can lead to a smaller execution time. For illustrating and quantifying this aspect, the execution time extra cost $\mathcal{C}_s$ due to MOS, has been measured on different algorithms implemented into a TMS320C64x. $\mathcal{C}_s$ represents the ratio between the additional execution time due to MOS and the execution time of the algorithm without MOS. The results are reported in table 1. This extra cost depends of the average IPC (instructions per cycle) obtained for the filter core section without MOS. When the IPC is closed to its maximal value 8, the extra cost is relatively important. Indeed, most of the functional units are used and supplementary cycles are required for executing the scaling operations. When the IPC decreases, the extra cost diminishes and can climb to 0%.

The experiments achieved with the FFT algorithm underline the influence of the number of storage elements available in the data path. Indeed, for computing respectively a radix 2 and a radix 4 FFT butterfly, 2 and 6 storage elements are required for storing the intermediate variables. Thus a lack of register leads to a SQNR degradation of $4dB$ for a radix 2 FFT butterfly.

These different experiments show the necessity of taking the different elements of the DSP architecture into account in order to optimize the data coding.

Filter	IPC	$\mathcal{C}_s$ (%)
Real single sample FIR	7.5	47
Real block FIR	6	22
Real symmetrical FIR	7.2	47
Real symmetrical block FIR	7.4	35
Complex single sample FIR	6.5	45
Complex block FIR	4.875	0
Biquad block IIR	2.8	18

Table 1: Multiplier output scaling extra cost

5 A NEW METHODOLOGY

The aim of this section is to present a new methodology for the implementation of floating-point algorithms into fixed-point DSPs under SQNR constraint. Available methodologies [2, 3] transform the floating-point C source code into an ANSI-C code with integer data types. Nevertheless, the different elements of the architecture are not taken into account for the fixed-point data coding. In our methodology, the determination and optimization of the data format is directed by the SQNR constraint. Moreover, the DSP architecture is completely taken into account during these two phases. The different phases of our methodology are represented at figure 3.

The first stage of the methodology is the determination of the data dynamic range. The results obtained are used for the definition of the minimal number of bits required for the integer part of the data in order to avoid an overflow. Then, the data are coded in two steps. The goal of the first step is to define the word-length of each data in order to take account of the diversity of the data types available in DSPs. The methodology selects the instructions which respect the SQNR constraint and minimize the code execution time. The format of each data is determined according to the DSP architecture. More especially, the scaling operations required for adapting the data format to the dynamic range or for respecting the fixed-point arithmetic rules are introduced according to the DSP shifter capacities. The second step corresponds to the data format optimization in order to minimize the code execution time as long as the SQNR constraint is fulfilled. This optimization is done by eliminating some scaling operations and is achieved in parallel with the scheduling stage. Indeed, as shown in the previous section for VLIW processor, the extra cost due to a scaling operation depends on the instruction scheduling.

The determination and optimization of the data formats are made under SQNR constraint. The methodology briefly presented in section 3.2 and detailed in [11] is used for evaluating the SQNR.

6 CONCLUSION

In this paper, the influence of the DSP architecture on the computation accuracy is analyzed. The different results allow the comparaison of different DSP architecture models. Thus, this kind of study can expand the classical performance measurement methodologies [13] based on code size or execution time measurement.

The diversity of data word-lengths offered by new DSP architectures requires investigation and evaluation of the tradeoff between accuracy, code size and execution time. Therefore, a new methodology for implementing floating-point algorithm in fixed point architecture under SQNR constraint is proposed. This study allows the definition of the different elements of the architecture required for the modelisation of the processor. Moreover, this method is coupled with the code generation process in order to minimize the code execution time under SQNR constraint.

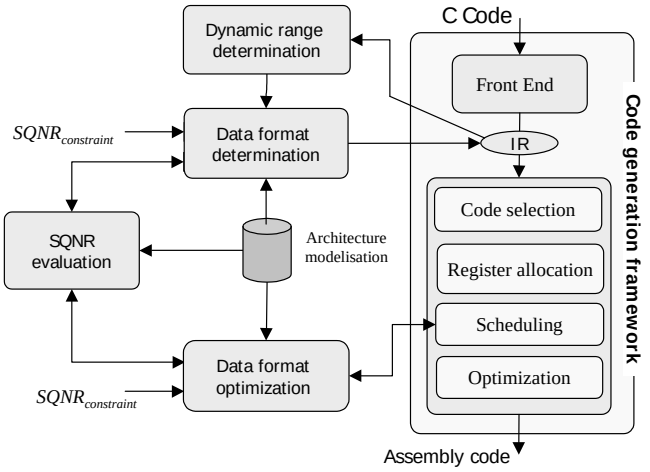

Figure 3: Methodology representation

References

[1] T. Grötker, E. Multhaup, and O.Mauss. Evaluation of HW/SW Tradeoffs Using Behavioral Synthesis. In *ICSPAT-96*, Boston, October 1996.

[2] K.I. Kum, J.Y. Kang, and W.Y. Sung. AUTOSCALER for C: An optimizing floating-point to integer C program converter for fixed-point digital signal processors. *IEEE Transactions on Circuits and Systems II*, 47:840–848, September 2000.

[3] M. Willems, V. Bursgens, and H. Meyr. FRIDGE: Floating-Point Programming of Fixed-Point Digital Signal Processors. In *ICSPAT-97*, San Diego, 1997.

[4] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee. *DSP Processor Fundamentals: Architectures and Features.* Berkeley Design Technology, Inc, Fremont, CA, 1996.

[5] Texas Instruments. *TMS320C54X DSP CPU And Peripherals Reference Set.* Texas Instruments, Dallas, Jan. 1999.

[6] Lucent Technologies. *DSP16xx Information Manual.* Lucent Technologies, January 1998.

[7] Texas Instruments. *TMS320C5X User's Guide.* Texas Instruments, June 1998.

[8] Texas Instruments. *TMS320C64x Technical Overview.* Texas Instruments, February 2000.

[9] Analog Device. *TigerSHARC Hardware Specification.* Analog Device, December 1999.

[10] L.B. Jackson. On the Interaction of Roundoff Noise and Dynamic Range in Digital Filters. *The Bell System Technical Journal*, 49(2), Feb. 1970.

[11] D. Menard and O. Sentieys. Automatic Evaluation of the Accuracy of Fixed-point Algorithms. In *DATE-02*, Paris, March 2002.

[12] G. Constantinides, P. Cheung, and W. Luk. Truncation Noise in Fixed-Point SFGs. *IEE Electronics Letters*, 35(23):2012–2014, November 1999.

[13] BDTi. The BDTImark2000: A Measure of DSP Execution Speed. Technical report, BDT Inc, 2001.