DIRECT SIGMA-DELTA MODULATED SIGNAL PROCESSING IN FPGA

Chiu-Wa Ng, Ngai Wong, Hayden Kwok-Hay So, and Tung-Sang Ng

Department of Electrical and Electronic Engineering
University of Hong Kong
Pokfulam Road, Hong Kong
email: {cwng,nwong,hso,tsng}@eee.hku.hk

ABSTRACT

The effectiveness of implementing bit-stream signal processing (BSSP) multiplier circuits in FPGAs, in terms of hardware resources and clock frequency, is presented. In particular, the result of realizing BSSP multipliers on FPGA architectures that utilize 6-input lookup tables (LUTs) is compared against architectures that utilize 4-input LUTs. It is found that architectures featuring 6-input LUTs suit well in BSSP applications where wide combinatorial paths are common. Furthermore, the performance of a BSSP multiplier is compared against conventional parallel multipliers in terms of LUT resource requirements. For a given resource requirement, it is found that an over-sampling ratio of less than 32 is required for a BSSP multiplier to outperform its parallel counterpart.

1. INTRODUCTION

Sigma-delta modulators (SDMs) are widely used to build analog-to-digital (A/D) and digital-to-analog (D/A) converters due to their simple architectures and good tolerance to analog component inaccuracy[1]. Conventional digital signal processors (DSPs) operate at the Nyquist rate while SDMs always generate over-sampled data. Decimators and interpolators must therefore be inserted before and after the DSPs for sampling rate conversion in order to interface with these over-sampled SDMs. The inclusion of decimators and interpolators inevitably introduce extra logics and routing resource consumptions. To allow for a resource-efficient way of signal processing, digital circuits that directly process the over-sampled bit-stream signal from the SDM output have been developed[2, 3, 4, 5, 6, 7, 8]. This technique is referred to as bit-stream signal processing (BSSP) [2]. In this paper, we evaluate the performance gain in implementing bit-stream multipliers using FPGAs with a 6-input LUT architecture over those with a 4-input LUT, in terms of resource utilization and speed. The contributions of this work are:

1. We present the detailed architectural design of the efficient bi-level bit-stream multiplier in [6] showing how the new 6-input LUTs FPGA architecture can be utilized to achieve compact and high-speed implementation.
2. We compare the result of implementing BSSP multipliers using a Xilinx Virtex-4 and a Xilinx Virtex-5 device to study the advantages due to FPGA architectural change.
3. We compare BSSP multipliers with traditional multi-bit multipliers to obtain conditions under which BSSP technique becomes more efficient than traditional Nyquist rate approach in terms of hardware resource consumption.

The rest of the paper is organized as follows: In Section 2, we describe our FPGA implementations of the bi-level and tri-level bit-stream multipliers. In Section 3, we present our implementation results and performance analysis. Finally, we conclude the paper in Section 4.

2. BIT-STREAM MULTIPLIERS IMPLEMENTATION IN FPGA

2.1. Resource-efficient Bi-Level Bit-stream Multiplier

Figure 1 shows the conventional bi-level bit-stream multiplier [2]. An efficient FPGA implementation of the bit-stream multiplier utilizing 4-input adder structure is recently proposed in [6]. Here, we provide the detailed architectural design of the bit-stream multiplier. A 4-input adder structure is shown in Figure 2. Using error feedback, it computes the average of four input bit-stream signals through the following equation:

$$\begin{aligned} &4y[n] + s[n] \\ &\quad = r_1[n] + r_2[n] + r_3[n] + r_4[n] + s[n-1] \end{aligned} \tag{1}$$

where $r_i[n], i = 1, 2, 3, 4$, denote the input bit-stream sequences, and $s[n] = 2s_1[n] + s_0[n]$ is the 2-bit truncation error. When implemented on the latest FPGA featuring 6-input LUTs, the 4-input adder (denoted as Type-II adder below) can be efficiently mapped onto three LUTs and two registers. Each 4-input bit-stream adder replaces three 2-input

978-1-4244-1961-6/08/$25.00

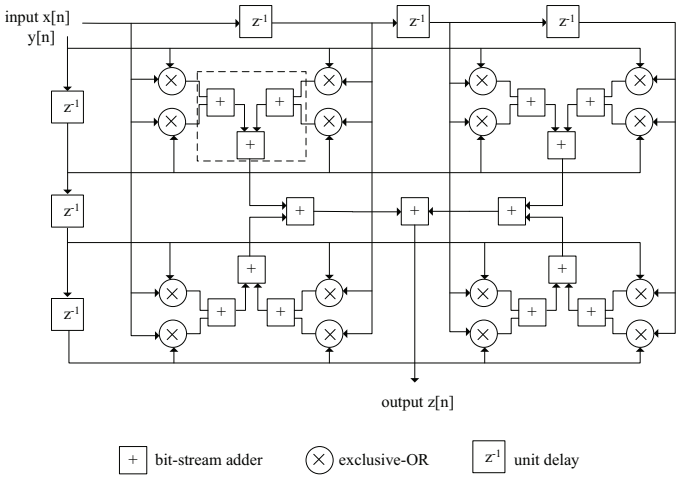

Fig. 1: A bi-level bit-stream multiplier[2]

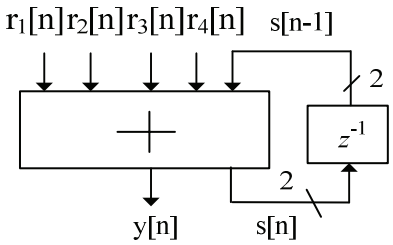

Fig. 2: A bi-level bit-stream adder with 4 inputs.

bit-stream adders. An example is indicated by the dashed box in Figure 1. Referring to the same figure, the use of 4-input adders results in only two layers of adders (instead of four) with the top level being fed by the sub-products $x[i]y[j]$.

When used to implement a bit-stream multiplier, the special nature of the inputs (the sub-products $x[i]y[j]$) to the top layer of the 4-input bit-stream adders actually gives rise to an efficient implementation of these adders, which is described below.

Denote, for simplicity sake, the four inputs to the bit-stream adder in the top layer as $x[0] \oplus y[0]$, $x[0] \oplus y[1]$, $x[1] \oplus y[0]$ and $x[1] \oplus y[1]$, where $\oplus$ denotes the XOR logic operation and the number in the bracket of each term denotes the number of unit delays. That is, instead of writing $x[n], x[n-1], \ldots$, we write $x[0], x[1], \ldots$. The right hand side of Equation 1 therefore becomes

$$\begin{aligned}&\Big[(x[0] \oplus y[0]) + (x[0] \oplus y[1]) \\ &+(x[1] \oplus y[0]) + (x[1] \oplus y[1])\Big] + s[n-1].\end{aligned} \tag{2}$$

Now we prove that the least significant bit (LSB) of the 2-bit truncation error $s[n]$ is always zero if the initial value of $s[n]$ is zero. Assuming first the LSB of $s[n-1]$ is zero,

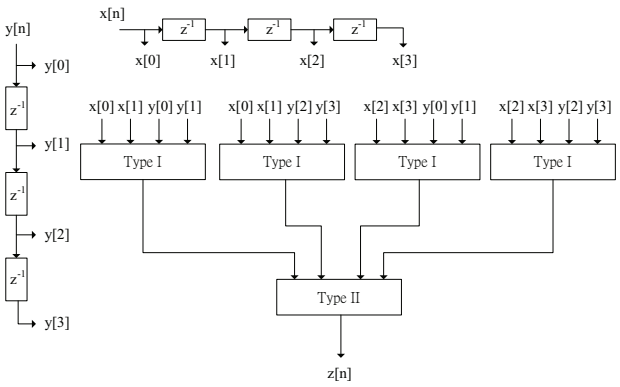

Fig. 3: A resource-efficient bit-stream multiplier.

i.e. $s[n-1]$ is even. Consider the case $y[0] \neq y[1]$. The summation of the square-bracketed term in Equation 2 reduces to $\Big(x[0] + \overline{x[0]} + x[1] + \overline{x[1]}\Big)$, which is 2. When added to $s[n-1]$, which is an even number, the result is also an even number, and hence the LSB of $s[n]$ is zero. For the case $y[0] = y[1]$, either they are zero or one. If $y[0] = y[1] = 0$, then Equation 2 becomes

$$\big(x[0]+x[0]+x[1]+x[1]\big)+s[n-1] = 2\big(x[0]+x[1]\big)+s[n-1]$$

which is an even number. Hence, the LSB of $s[n]$ is again zero. The case for $y[0] = y[1] = 1$ is similar and leads to the same conclusion. As it is assumed that the initial value of $s[n]$ (i.e., when $n = 0$) is zero, the LSB of $s[n]$ is always zero by mathematical induction.

The fact that the LSB of $s[n]$ is always zero simplifies the implementation of the 4-input bit-stream adder in the top layer: it is now a function of five inputs and two outputs. Therefore, only two 6-LUTs and one flip-flop (FF) are required under this special case of inputs. Furthermore, referring to Equation 2, since the sub-products are formed by four inputs $(x[0], x[1], y[0], y[1])$, the XOR operation among them can thus be merged into the LUTs of the 4-input bit-stream adder. As a result, the overall design of the bit-stream multiplier now consists of two types of 4-input bits-stream adders, denoted as Types I and II, as shown in Figure 3. Type-I adder is the merged-XOR structure in the top layer and Type-II adder is the original structure. There are four Type-I blocks and one Type-II block. Including the six shift registers at the input, this structure requires a total of eleven LUTs and twelve FFs, as is verified in Section 3

2.2. Tri-level Bit-stream Multiplier

The structure of a tri-level bit-stream multiplier, which was detailed in [7], is replicated in Figure 4 for easy comparison. Comparing Figure 4 with Figure 1, it can easily be seen that the sturcture of bi-level and tri-level bit-stream multipliers are very similar. Similar to its bi-level counter part, a

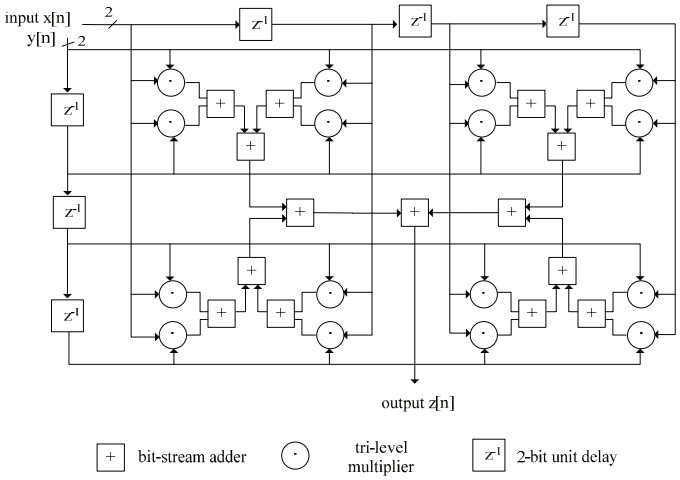

Fig. 4: A tri-level bit-stream multiplier[7]

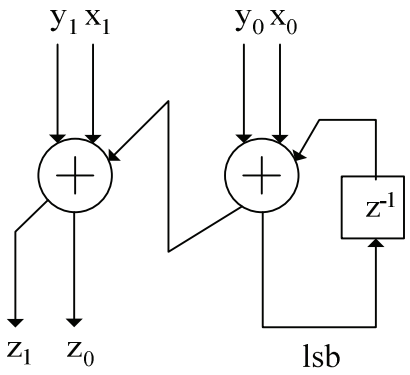

Fig. 5: A tri-level bit-stream adder.

tri-level bit-stream multiplier is also constructed using two types of major components: (a) tri-level bit-stream adders; and (b) tri-level digit multipliers, which will be briefly described below.

A tri-level adder has the structure shown in Figure 5. Unlike the case for the bi-level design, a tri-level 4-input adder cannot be efficiently implemented. Since each signal consists of two bits, a tri-level 4-input adder adds five 2-bit signals (four 2-bit inputs plus a 2-bit feedback). Implementing these logic functions requires multi-level 6-input LUTs. The adder tree of the tri-level bit-stream multiplier just follow the 2-input bit-stream adder tree structure shown in Figure 1. As shown in Figure 5, the outputs of a tri-level bit-stream adder are functions of up to five inputs. As a result, the tri-level bit-stream multiplier can benefit from the Virtex-5 6-input LUT architecture. We just let the synthesizer to optimize the wide-input combinatory logic consisting of the tri-level bit-stream adder tree and the tri-level digit multipliers.

A tri-level digit multiplier implements the following logic equations:

$$z_0 = x_0 y_0$$
$$z_1 = \overline{x_1} x_0 y_0 + x_1 \overline{y_1} y_0$$

where $z_1 z_0$ denots the 2-bit output while $x_1 x_0$ and $y_1 y_0$ denote the 2-bit inputs.

Table 1: Implementation results on Xilinx Virtex-4 and Virtex-5 for logic utilization and clock frequency (MHz).

	Bi-level		Tri-level	
	Virtex-4	Virtex-5	Virtex-4	Virtex-5
No. of LUTs	26	11	84	59
No. of FFs	12	12	27	27
No. of slices	18	5	55	16
Max freq.	302	634	252	318

3. IMPLEMENTATION RESULTS AND DISCUSSION

3.1. Virtex-4 vs Virtex-5

The bit-stream multipliers are implemented with Xilinx Virtex-5 XC5VLX30 and Virtex-4 XC4VLX25 using the design tool ISE 9.1i. Table I presents the implementation results for the bi-level and tri-level bit-stream multipliers. We can see that moving from the 4-input LUT architecture (Virtex-4) to the 6-input LUT architecture (Virtex-5), both bit-stream multipliers show resource savings on LUTs and higher clock speed. This means that our multiplier designs can take advantage of the new 6-input LUT feature.

The effect of LUT reduction and speed-up on the bi-level bit-stream multiplier is greater than that of the tri-level one. This is due to the use of the 4-input bit-stream adder shown in Figure 2. As explained in Section 2.1, the 4-input bit-stream adder (Type-I adder) and its variant, Type-II adder can be efficiently mapped onto one level of 6-input LUTs. This allows for the implementation of the bi-level bit-stream multiplier using only two levels of LUTs, thus achieving a very high speed. In contrast, on Virtex-4 platform, Type-I and -II adders must be split among 4-input LUTs and hence more LUTs are required and poorer speed performance is observed.

For the tri-level bit-stream multiplier, only 2-input bit-stream adder tree structure is implemented. This results in a higher logic complexity and slower speed than the bi-level design. As the multiplier consists of combinatory logic functions of over four inputs (tri-level bit-stream adders), the logic mapping in 6-input LUTs is more efficient than in 4-input LUTs. This is confirmed in Table 1 when the implementation results for Virtex-4 and -5 are contrasted.

3.2. Bit-stream vs Multi-bit Multiplication

We compare the FPGA resource requirements for bit-stream and multi-bit multipliers to investigate the condition on over-

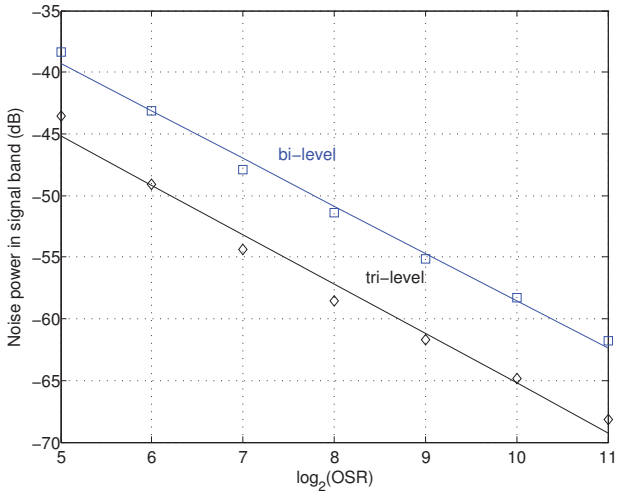

Fig. 6: Noise power in signal band against OSR.

Table 2: Multi-bit multiplier implementation results on Xilinx Virtex-4 and Vitex-5.

	No. of LUTs	
Bit-width	Virtex-4	Virtex-5
4	17	17
5	33	31
6	40	38
7	62	60
8	71	69
9	101	98

sampling rate (OSR) when BSSP begins to show LUT advantage over traditional multi-bit approach. Figure 6 shows the noise power verus OSR plot for the bi-level and tri-level bit-stream multiplers. The graph is obtained using Matlab simulation. The quantization noise of an n-bit multiplier is approximated in [2] as:

$$\text{Quantization Noise} = -(6n + 5)\,\text{dB} \qquad (3)$$

We use Xilinx CORE generator to implement multipliers with various bit-lengths on Virtex-4 and -5 devices. The results are tabulated in Table 2.

Comparing the LUT results in Table 1 with Table 2, for the case of bi-level bit-stream multiplier implemented on a Virtex-4 device, we see that the 26 LUTs sits between the LUT resources of a 4-bit and a 5-bit multiplier. According to Equation 3, the quantization noise power of a 5-bit multiplier is about -35 dB. From Figure 6, this can be achieved when the OSR is about 32. The "break-even" OSRs, i.e., the OSR beyond which a bit-stream multiplier becomes more efficient than a multi-bit multiplier, of the other three cases in Table 1 can be similarly obtained. The results are summarized in Table 3.

Table 3: Break-even OSRs for bi-level and tri-level bits-stream multiplers in Virtex-4 and Virtex-5.

	Bi-level		Tri-level	
	Virtex-4	Virtex-5	Virtex-4	Virtex-5
Break-even OSR	< 32	< 32	256	64

From Table 2, it can be found that the LUT results on multi-bit multiplier implementation do not varies significantly when Virtex-5 is used instead of Virtex-4. In contrast, for bit-stream multipliers, the Virtex-5 implementation is more efficient than the Virtex-4 implementation. Therefore, we can see that the break-even OSR for Virtex-5 occurs early than that for Virtex-4 and we can conclude that BSSP in Virtex-5 performs better than in Virtex-4. Note that for bi-level bit-stream multiplier implementations, the break-even OSRs are below 32. In practical applications, the OSR is almost always higher than 32.

4. CONCLUSION

We have implemented bi-level and tri-level bit-stream multipliers in Virtex-4 and Virtex-5 to study the performance gain of BSSP arithmetic circuits in the new 6-input LUT architecture over the 4-input LUT architecture. It has been shown that BSSP implementation is more effective in FPGA featuring 6-input LUTs in terms of resource utilization and clock speed. We have also found the OSRs for the bi-level and tri-level bit-stream multipliers above which BSSP becomes more resource-efficient than traditional Nyquist rate multi-bit operations.

5. REFERENCES

[1] S. R. Norsworthy *et al.*, *Delta-Sigma Data Converters, Theory, Design, and Simulation*. Piscataway, NJ: IEEE Press, 1997.

[2] H. Fujisaka *et al.*, "Bit-stream signal processing circuits and their application," *Transaction on Fundamentals of Electronics Communications and Computer Sciences*, vol. 85, no. 4, pp. 853–860, 2002.

[3] ——, "Arithmetic circuits for single-bit digital signal processing," in *Proceedings of the 6th IEEE International Conference on Electronics, Circuits and Systems*, vol. 3, 1999, pp. 1389–1392.

[4] Y. Hidaka *et al.*, "Piecewise linear operations on sigma-delta modulated signals," in *Proceedings of the 9th International Conference on Electronics, Circuits and Systems*, vol. 3, 2002, pp. 983–986.

[5] D. A. Johns and D. M. Lewis, "Design and analysis of delta-sigma based iir filters," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 4, pp. 233–240, 1993.

[6] C. W. Ng *et al.*, "Efficient fpga implementation of bit-stream multipliers," *Electronics Letters*, vol. 43, no. 9, pp. 496–497, 2007.

[7] ——, "Bit-stream adders and multipliers for tri-level sigma-delta modulators," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 12, pp. 1082–1086, Dec 2007.

[8] P. O'Leary and F. Maloberti, "Bit stream adder for oversampling coded data," *Electronics Letters*, vol. 26, no. 20, pp. 1708–1709, 1990.