A Scalable and Programmable Sound Synthesizer

Tae-Hwan Kim, Young-Joo Lee and In-Cheol Park
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea

Abstract— **Sound synthesis employed in many multimedia systems is a useful method to generate sounds of musical instruments. In this paper, we propose a new VLSI architecture suitable for a scalable sound synthesizer based on a programmable data-flow. The sound quality and the level of polyphony can be enhanced only by increasing operating speed and enlarging memory. A fully integrated sound synthesis system is implemented as a prototype to verify the proposed architecture. The prototype chip fabricated in a 0.18-μm CMOS process occupies 1.5 x 1.5 mm^2, and can synthesize up to 64-polyphonic sound. The power consumption ranges from 2.05 mW to 13.8 mW depending on the quality of the synthesized sound.**

I. INTRODUCTION

Sound synthesis is to generate a natural sound electrically by appropriately modeling physical characteristics of musical instruments. It is one of the most important technologies commonly required in multimedia systems. Compared to the classical method that plays back recorded sounds, sound synthesis requires much less memory, which makes it feasible to support various sounds even in mobile devices such as cellular phones or portable multimedia players.

Due to the advances of circuit and process technologies, hardware architectures for sound synthesis have evolved by integrating more processing elements to produce high quality, polyphonic sounds [1]-[6]. Recently, a few commercial products were developed focusing on low-power consumption [6], as the mobile multimedia system becomes an attractive application area.

This paper proposes a new programmable hardware architecture that has a configurable data-flow optimized for sound synthesis. In the proposed architecture, both FM synthesis and additive synthesis are supported in a hybrid manner. In addition, without any major revision of the proposed architecture, the sound quality and the level of polyphony can be enhanced simply by increasing operating speed and enlarging the size of memory. A fully integrated sound synthesis system is designed to demonstrate the efficiency of the proposed architecture.

The rest of the paper is organized as follows. Fundamental theories on FM synthesis and additive synthesis are introduced in Section II. Section III proposes a scalable architecture for sound synthesis, and Section IV describes the design of a sound synthesis system based on the proposed architecture. The implementation results and performances of the proposed system are discussed in Section V. Finally, concluding remarks are made in Section VI.

This work was supported by Korean Intellectual Property Office (KIPO) and IC Design Education Center (IDEC).

II. SOUND SYNTHESIS METHODS

FM synthesis can make harmonic components of various timbres in a nonlinear manner by modulating the carrier signal [7]. In the time domain, the FM signal, $FM(t)$ can be expressed as follows:

$$FM(t)=A(t)\cdot sin(f_c t+B(t)\cdot sin(f_m t)), \tag{1}$$

where f_c is the carrier frequency, f_m is the modulating frequency, and $A(t)$ and $B(t)$ are functions of the envelopes of the carrier and modulating signals, respectively. The envelope, $A(t)$, is employed to model the physical mechanics of an instrument, which is conventionally modeled as attack, decay, sustain, and release (ADSR) curves [8]. The inner modulating signal $B(t)\cdot\sin(f_m t)$ deviates the carrier frequency, resulting in a large number of harmonics whose frequency components exist in the sidebands of the carrier frequency. The main frequency component determines a pitch, and the side band components determine a timbre.

Fig. 1 shows the basic FM synthesis algorithm expressed in (1), which requires two signal generators. Each signal generator called an operator consists of an envelope generator and a modulator. The output of the operator denoted by OP$_0$ is fed into OP$_1$ to modulate the carrier frequency.

The additive synthesis is an intuitive method to produce an arbitrary sound composed of many sinusoidal tones, and it is based on Fourier theory that any signal can be decomposed into a number of tones. The additive synthesis can be applied to more general applications such as voice synthesis, whereas the FM synthesis is appropriate for harmonic sound.

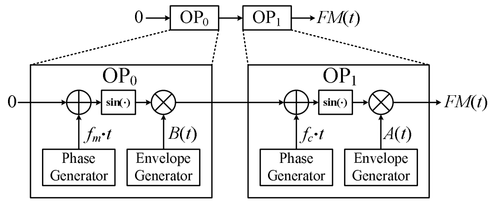

Figure 1. Two-operator algorithm for FM synthesis.

III. Proposed Scalable Architecture

A. Overall Architecture of the Proposed System

Fig. 2 shows the conceptual view of the proposed system in which two processors are integrated. One of them is the host processor which interprets musical instrument digital interface (MIDI) data [9], and the other is the sound synthesizer which is responsible for synthesizing a sound according to the interpreted MIDI data. In the proposed system, a channel is logically defined as a physical output of the synthesizer such as an electrical speaker as shown in the figure, and a track as the sound of a musical instrument. A number of operators can be involved to synthesize a track. In the figure, two operators are involved in each of the two tracks to synthesize sounds. As the audio sampling rate in the system is much lower than the target operating frequency of the sound synthesizer and the host processor, a first-in first-out (FIFO) memory is employed. The output of the FIFO is serially delivered to the audio CODEC through the interface unit to produce sounds synchronized with the sampling rate.

In conventional sound synthesis architectures [1]-[4], a number of processing elements are physically integrated and their connections are hard-wired, resulting in a limited number of supported algorithms [1]-[6]. When more processing elements are required to achieve higher quality or polyphonic sound, architecture revision is indispensable in conventional architectures. Additionally, some of the processing elements are idle in synthesizing low-quality or monophonic sound. As such an idle processing element exists physically, it consumes power even if it does not participate in sound synthesis.

In the proposed synthesizer, however, there are only a few processing elements physically, and these are programmed to serve the roles of the logical entities in a synthesis algorithm. Regarding to operators, an operator logically required in a synthesis algorithm is named as a logical operator, while the processing element responsible for the logical operator is named as a physical operator. In the proposed synthesizer, there is only one physical operator, and the physical operator can be programmed to work as the four logical operators shown in Fig. 2 in a time-multiplexed manner. Instructions needed to control the physical operator are sequentially loaded from the data-flow description memory (IDM), which stores how the physical operator is controlled. Parameters and states required for logical operators are provided from memories (Track parameter memory (TPM), operator parameter memory (OPM), operator state memory (OSM)) as shown in Fig. 2. By repeating the processing for a number of logical operators involved in sound synthesis, any synthesis algorithms can be supported without performing hardware revision. Additionally, the quality of sound can be enhanced easily with higher operating speed and larger memories.

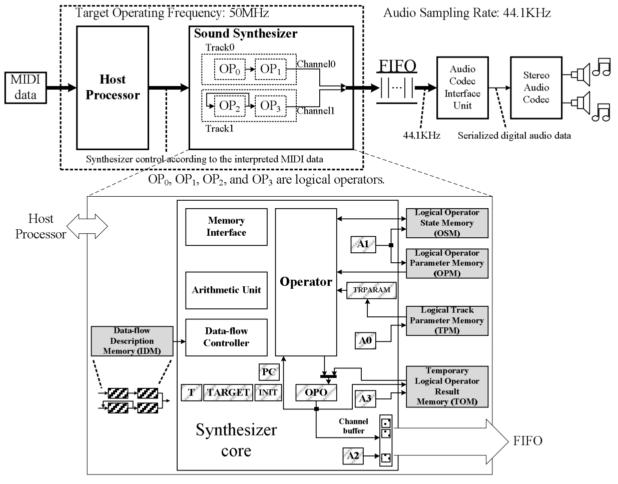
Target Operating Frequency: 50MHz
Audio Sampling Rate: 44.1KHz
MIDI data
Host Processor
Sound Synthesizer
Track0
OP_0
OP_1
Channel0
OP_2
OP_3
Channel1
Track1
FIFO
Audio Codec Interface Unit
Stereo Audio Codec
44.1KHz
Serialized digital audio data
Synthesizer control according to the interpreted MIDI data
OP_0, OP_1, OP_2, and OP_3 are logical operators.
Host Processor
Memory Interface
Arithmetic Unit
Data-flow Controller
Operator
Data-flow Description Memory (IDM)
Logical Operator State Memory (OSM)
Logical Operator Parameter Memory (OPM)
Logical Track Parameter Memory (TPM)
Temporary Logical Operator Result Memory (TOM)
A1
TRPARAM
A0
PC
T
TARGET
INIT
OPO
A3
Channel buffer
A2
Synthesizer core
FIFO

Figure 2. Conceptual view of the sound synthesis system.

B. Programmable Architecture Based on Data-Flow Analyses for Sound Synthesis

The data-flow of sound synthesis can be modeled with a few kinds of processing elements including the operator. The most important processing element is the operator whose functionality is formulated as follows.

```
$O \leftarrow$ Operator($State$, Parameters, $I$, $m$){
    $O \leftarrow State.\mathrm{E} \cdot \sin(State.P + I \cdot m)$
    Calculate $\Delta E$ with Parameters and $State.E$
    Determine $\Delta \mathrm{P}$ with Parameters
    $State.E \leftarrow State.E + \Delta E$, $State.P \leftarrow State.P + \Delta P$
},                                                        (2)
```

where I and O stand for the input and the output of the operator, respectively. The envelope and the phase for the operator are denoted as $State.E$ and $State.P$, respectively. They are incrementally updated by ΔE and ΔP. The envelope increment ΔE is calculated with the parameters representing the slope of each region in ADSR curve and the current envelope value. The phase increment ΔP is determined by considering the sampling rate and using the parameters related to octaves and notes.

From the viewpoint of operator connections, the logical operators can be classified into two types. One has an input connected to the output of another logical operator, and the other has an input which is fed back. In Fig. 2, OP2 corresponds to the latter, and OP1 to the former. As sound synthesis algorithms are programmed to execute logical operators sequentially, the result of a logical operator is stored into a special register or forwarded by using a bypassing network, if it is to be processed in another logical operator. To support the feedback input, the proposed architecture supports memory operations to store the result of such a logical operator into a memory, and to retrieve it into a register for the processing of the next sample. The operator located first in a track, such as OP0 in Fig. 2, may have no feedback input. In this case, the physical operator is processed with the zero multiplying factor (m=0). Based on this feature, the input and the output of the physical operator are constrained to the register, and the feedback connection is programmed with memory operations.

A parallel interconnection between two operators is modeled with addition. This case arises when two sound tracks are played into one channel. To combine the sounds synthesized from multiple tracks, an addition operation is supported in the architecture.

TABLE I. SOME INSTRUCTIONS IN THE PROPOSED SYNTHESIZER

Instruction	Function
OP *m*, (A1++)	Operator processing with the multiplication factor *m*, the states OSM[A1], and the parameters OPM[A1]. Selectively increases A1.
LT (A0++)	Load a track parameter from TPM[A0]. Selectively increases A0.
WOB (A2++)	Write OPO to the A2-th channel buffer. Selectively increases A2.
LOP (A3++)	Load data from TOM[A3] into OPO. Selectively increases A3.
LAOP (A3++)	Load data from TOM[A3], and accumulate it to OPO. Selectively increase A3.
SOP (A3--)	Store OPO to TOM[A3]. Selectively decrease A3.
COM	Commit the data in channel buffers into FIFO, and reset PC and all address registers to repeat processing from the start of a program.

Based on the analyses of data-flows, we defined a set of instructions optimized for sound synthesis. Some of them are listed in Table I. Every instruction is 8 bits long. The instructions have operands from registers or memories. For example, the OP instruction reads the states and parameters for a logical operator from memories addressed by A1, and reads I defined in (2) from the register, OPO, and writes O in (2) to the same register. The multiplying factor is specified in the instruction itself as an immediate operand. Instructions including LOP, LAOP, and SOP are defined to store and retrieve the result of a logical operator with the temporary output memory (TOM). Some instructions implicitly use one of the address registers, A0~A3, and can increment or decrement the corresponding address register automatically. As the parameters and the states are usually accessed in a sequential order, the feature of auto-increment or auto-decrement helps reduce the overall program size.

The architecture works according to the configured data-flow program with states and parameters for logical operators. To synthesize a higher quality sound by adopting a more complex algorithm in which more operators are involved, what we have to do is to reprogram the data-flow with enlarging memories. Although the sound synthesizer needs to operate in a higher frequency for this case, the modern CMOS technology usually affords such frequency.

In the proposed architecture, various synthesis algorithms can be supported in a hybrid manner. As the operator processing defined in (2) can be applied to additive synthesis as well as FM synthesis, a hybrid synthesis is possible in the proposed architecture. For example, we can generate some instrumental sounds by FM synthesis and vocal sounds by additive synthesis, simultaneously.

IV. DESIGN OF A PROPOSED SYSTEM

We design a sound synthesis system based on the proposed architecture with fundamental components fully integrated. Fig. 3 shows the overall architecture of the proposed synthesis system. Two processors are integrated in the proposed system. The first processor called Core-A is a 32-bit general-purpose processor developed for embedded applications. This is working as the host processor that interprets MIDI data and controls the synthesizer. The second processor is a sound synthesizer. The data-flow description of the synthesizer and parameters needed for logical operators and tracks are all mapped into some memory segments in the system, and they can be accessed by the host processor through the on-chip bus. The host processor configures the data-flow of a specific synthesis algorithm according to interpreted MIDI data, and dynamically updates the track parameters and operator parameters.

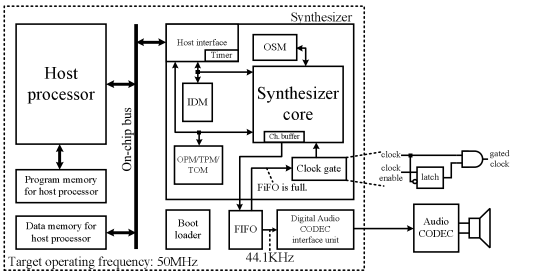

Figure 3. Overall architecture of the proposed synthesis system.

The synthesized sound samples are first stored into the FIFO and fetched into the digital audio CODEC interface unit. Reading the FIFO is synchronized with the audio sampling rate, while writing is done in pace with the system clock frequency. As the audio sampling rate is much lower than the target operating frequency of the system, the FIFO has to be controlled carefully. Therefore, it is desired to safely stop the synthesizer as soon as the FIFO is full. For this, the clock of the synthesizer is gated with the state of the FIFO, as shown in Fig. 3 where a conventional latch-based circuit is employed to achieve glitch-less clock gating. When stopped, the synthesizer does not consume dynamic power due to the gated clock. The simpler synthesis algorithm leads to the longer inactive time. Exploiting this feature and the programmable data-flow, we can adaptively change the synthesis algorithm considering the system power budget.

The sound synthesizer is a reduced instruction set computer (RISC) specially designed to support the instructions defined for sound synthesis. The core has six pipeline stages: instruction fetch (IF), instruction decoding (DEC), memory access (MEM), two stages of operator processing (OP1st and OP2nd), and write back (WB). An instruction indexed by PC is fetched from IDM in the first stage. The fetched instruction is decoded and some registers are read in the second stage. In the third stage, memory operands are loaded and a synthesized sample is stored into the channel buffer. The fourth and the fifth stage are for operator processing or accumulation. In the last stage, the result is written back into a register. Fig. 4 shows the pipeline architecture of the synthesizer core, including bypassing networks to avoid data hazards between instructions.

V. IMPLEMENTATION RESULTS

Table II summarizes the characteristics of the prototype chip designed as a soft IP, and the layout of the prototype chip is shown in Fig. 5. As the size of states needed for a logical operator is 32 bits, the maximum number of logical operators that can be supported in the prototype chip is 128, considering the sizes of integrated memories. This means about 64-

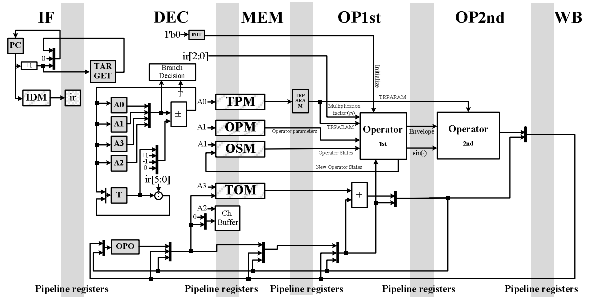

Figure 4. Pipeline architecture for the proposed synthesizer.

polyphonic FM sounds are synthesizable, if a track is synthesized with the two-operator algorithm. As the limitation of the number of logical operators comes from the memory size and clock frequency, higher-quality sound can be synthesized by simply increasing memory size or clock frequency without performing any architecture revision.

To investigate the power efficiency of the proposed architecture, power simulation is carried out for the post-layout design annotated with switching activities. In Table II, we can see that the power consumed for the low-quality sound is much lower than that for the high-quality sound. In synthesizing the low-quality sound, the synthesizer core is stopped most of the time by the gated clock explained in Section IV, as the data-flow is relatively simple.

In Table III, the prototype system is compared with previous works. The prototype system can support both FM synthesis and additive synthesis in a hybrid manner and any algorithms by programming the data-flow description, while the previous ones can support only one or a limited set of algorithms. In the prototype system, the number of logical operators is up to 128, which is much larger than those of the previous works, and is easily scalable by increasing memory size and clock frequency. In the proposed system, there is only one physical operator that is repeatedly controlled to have the same functionality as a specific data-flow composed of multiple logical operators. As the operator is the most critical component in hardware complexity, the proposed system is efficient in terms of area. Table III shows that the gate count and chip area resulting from the proposed system are much smaller than those of previous works, even taking into account the technology differences.

TABLE II. CHARACTERISTICS OF THE PROTOTYPE SYSTEM

Sound sampling rate	44.1 KHz
Sound dynamic range	120.4 dB (20 bit)
Technology	0.18-μm CMOS, 1-poly 6-metal, 1.8V
Chip size	1.5 x 1.5 mm^2
Operating frequency	50 MHz (Max. 123 MHz)
Equivalent gate counts	13.5 K (Synthesizer), 40.2 K (Total)
Power consumption	2.05mW (1-track) , 13.8mW (64-track)

VI. CONCLUSION

In this paper, we have presented a programmable VLSI architecture developed for sound synthesis. Analyzing data-flows of sound synthesis algorithms, we proposed a new architecture in which the data-flow of sound synthesis is programmable with a small set of instructions. The proposed architecture can support higher quality or polyphonic sound by increasing the sizes of parameter and state memories and the clock frequency. We implemented a complete sound synthesis system to validate the proposed architecture. The prototype chip designed in a 0.18-μm CMOS process occupies about 2.25 mm^2, and its power consumption ranges from 2.05 mW to 13.8 mW depending on the quality of sound.

REFERENCES

[1] S. Bernadas, M. Alexander, J. Bian, G. Chowdhury, Q. Dong, M. Gentry, and et al., "A single-chip multimedia audio system with digital sample rate conversion and FM sound synthesis," *in Proc. of ISSCC*, pp. 252-253, 455, Feb. 1996.

[2] F. De Bernardinis, R. Roncella, R. Saletti, P. Terreni, and G. Bertini, "An efficient VLSI architecture for real-time additive synthesis of musical signals," *IEEE Trans. on VLSI Systems*, vol. 7, no. 1, pp. 105-110, Mar. 1999.

[3] T. Hodes, J. Hauser, J. Wawrzynek, A. Freed, and D. Wessel, "A fixed-point recursive digital oscillator for additive synthesis of audio," *in Proc. of ICASSP*, vol. 2, pp. 993-996, Mar. 1999.

[4] Ho Keun Jang, "A design of sound synthesis IC," *in Proc. of ASP-DAC*, pp. 327-328, Feb. 1998.

[5] YAMAHA Coporation, "YMF-262 - FM operator type 3 (OPL-3),".

[6] YAMAHA Coporation, "YMU-786 - mobile audio 5 (MA-5),".

[7] J. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the Audio Engineering Society*, vol. 21, issue. 7, pp. 526-534, Sep. 1973.

[8] F.R. Moore, "Elements of computer music," 1990, Prentice-Hall, Englewood Cliffs, NJ.

[9] MIDI Manufacturers Association, "Complete MIDI 1.0 detailed specification," Nov. 2001, http://www.midi.org/techspecs/midispec.php

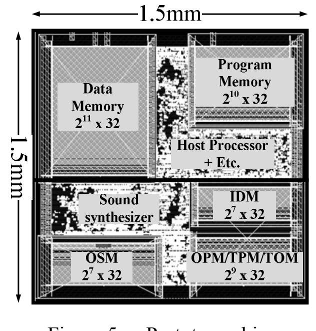

Figure 5. Prototype chip.

TABLE III. PERFORMANCE OF THE PROPOSED SYSTEM COMPARED WITH PREVIOUS WORKS

Architecture	Proposed[a]	[1][b]	[4][c]	[2]	[5]	[6]
Supported synthesis	Additive + FM	FM	FM	Additive	FM+Wavetable	
Supported algorithms	Anything programmable	2-operator algorithms	N.A.	Additive	2- or 4-operator algorithms	
# of operators[d]	128	36	N.A.[e]	N.A.	36	32
Technology	0.18-μm CMOS	0.6-μm CMOS	0.8-μm CMOS	0.5-μm CMOS	N.A.	
Equivalent gate counts	40.2K	816K	50K	N.A.	N.A.	
Chip size	2.25 mm^2	56 mm^2	259 mm^2	19 mm^2	N.A.	

a. MIDI controller + sound synthesizer, and all the components described in Fig. 3.
b. Sound synthesizer + sample rate converter.
c. MIDI controller + sound synthesizer.
d. The maximum number of operators logically used in synthesis algorithms.
e. 32 poly-phonic sound can be synthesized.