Area-Efficient Analog Operations by Dynamicly-Reconfigured Switched-Capacitor Circuits

Fuminori Kobayashi and Shintaro Higuchi
formerly Dept. of Systems Design and Informatics
Kyushu Institute of Technology
Iizuka, Fukuoka, Japan
f_koba@k9.dion.ne.jp

Mohamad Zain Azreen Bin Ramli
Institute of Microengineering and Nanoelectronics
Universiti Kebangsaan Malaysia
Bangi, Selangor, Malaysia

***Abstract* - Together with software on a micro-controller, simple additional hardware can implement electronics with less cost and power. Though analog before digitization, in particular, is effective, it comes with a drawback of larger circuit area. This paper proposes a scheme to shrink area by devising clock waveforms of switched-capacitor devices, through dynamic circuit restructuring. Application to a D/A converter using weighted adder is presented.**

Keywords - Cypress PSoC; nonuniform clocking

I. INTRODUCTION

Today, electronics is mainly implemented by software on micro-controllers, and only limited parts rely on hardware. Especially, analog is not preferred except A/D converter (ADC), though some functions can more efficiently be realized by analog particularly in radio-frequencies, low-power or high-speed areas [1, 2].

Major drawback of analog is its large circuit area [3]. In order to cope with this problem, we use the principle of "dynamic reconfiguration" (hereinafter abbreviated as DR) [4]. DR, in contrast to usual circuits inherently fixed in structure, is that circuit changes its topology while it runs. With this property, it can sometimes offer sophisticated operations with less amount of hardware. Though DR has mainly dealt with digital, analog DR is already investigated [5, 6 and others].

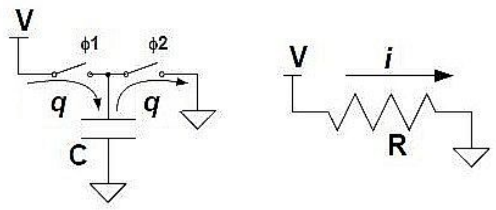
By changing its structure with DR, our scheme of analog operation can accept plural inputs with a single-input operational (OP) amp(lifier), leading to smaller circuit. Its key is asymmetric clock realized by DR in the switched- capacitor (hereinafter abbreviated as SC) technique [7]. Among commercial SC chips, the prototype used PSoC [8] rather than dpASP, formerly called FPAA [9].

In sections below, mechanism of SC is shown first, followed by principle and experiment of adder/subtracter by a one-input OP amp with DR; finally shown is adder with weights and its application to D/A converter (DAC), closed by conclusion.

II. SWITCHED-CAPACITOR TECHNIQUE

A. Principle of Switched-Capacitor Scheme

Fig. 1 shows that an SC circuit [7] is equivalent to a resister, where $\phi 1$ and $\phi 2$ are complementary, two-phase clocks. When the left switch driven by $\phi 1$



(a) SC circuit (b) Resister circuit
Figure 1. Fundamental SC Structure

closes, charge $q=VC$ flows into capacitor C, while the right switch driven by $\phi 2$ is open. When right closes and left opens, q flows out of C into the ground at right. At every commutation of two switches, q flows from left to right. This operation is similar to a resister in Fig. 1 (b), because current i (Ampere) carries charge i (Coulomb) every second in (b) and current ripples in (a) can be ignored if commutation is fast.

B. Switched-Capacitor OP Amp

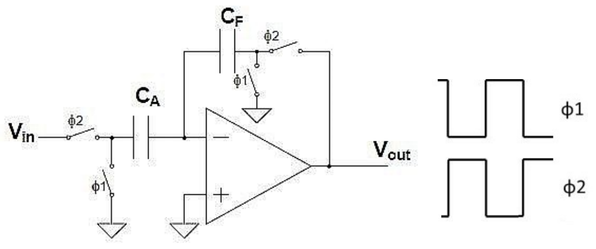
Fig. 2 is an OP amp circuit, configured as inverting amplifier, based on the basic SC technique of Fig. 1. Here, $\phi 1$ and $\phi 2$ are complementary, as shown in (b).

Note that the inverting input "-" of OP amp is "virtual ground". During $\phi 2$, Vin is applied to C_A, and the charge from Vin goes to C_F to drive $Vout$ to $-Vin\ C_A/C_F$. During $\phi 1$, two capacitor charges are reset.

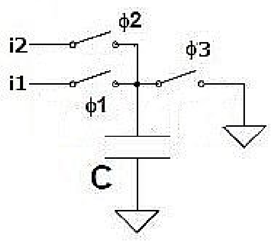
III. SC ADDER WITH ASYMMETRIC CLOCKS

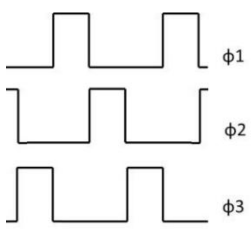
A. Charge Addition with Three-Phase Clocks

Fig. 3 (a) is similar to Fig. 1 (a), modified for two inputs. Note that inputs are in current, not in voltage, which is easily realized with the OP amp scheme, shown



(a) Schematic (b) Clocks
Figure 2. SC Operational Amplifier and its Clocks





(a) Principle (b) Asymmetric clocks

Figure 3. Plural-input SC with Asymmetric Clocks

in Fig. 2 (a). Clocks to this circuit are of three-phase, asymmetric, noncomplementary type, as in Fig. 3 (b).

In this circuit, first $\phi 1$ is activated, making current *i1* to flow into *C;* then $\phi 2$ is activated, flowing *i2* into *C*. Since charge caused by *i1* was already kept in *C*, resulting charge will correspond to *i1* + *i2*. Finally, when $\phi 3$ is activated, the summed charge will be drawn at the output, meaning addition implemented by the three-phase, asymmetric clock waveforms.

Some researches on similar nonuniform clocking for SC are found, for example in [10].

B. *Implementation with a Commercial SC Chip*

A hurdle for implementing this principle is that building blocks of most commercially available SC amplifier do not have plural inputs, nor asymmetric, three-phase clocks. Here, we propose to use the DR property of PSoC$^{(R)}$ from Cypress Inc, to realize plural inputs and asymmetric clocks.

Fig. 4 is a simplified schematic of PSoC [8] SC amp, where controllability of feedback capacitor clock through “FSW0” and “FSW1” bits is noted (For simplicity, “AutoZero” bit for canceling OP amp offset voltage is ignored). These are flags driven by registers of MPU inside PSoC, not only at startup but also at runtime, as a property of DR in PSoC [11].

In addition, note that a multiplexer to select one of the “A Inputs” is placed at the left of Fig. 4. It is controlled by “AMux” bits, via MPU also by DR.

Combining the two DR properties, operation shown in Table 1 implements Fig. 3. In the table, first column means time step every clock, 2nd to 3rd columns how to realize asymmetric clocking, and 4th to 5th columns how AMux emulates plurals input with a one-input OP amp. Column 6 shows relationship of steps to Fig. 3 (b).

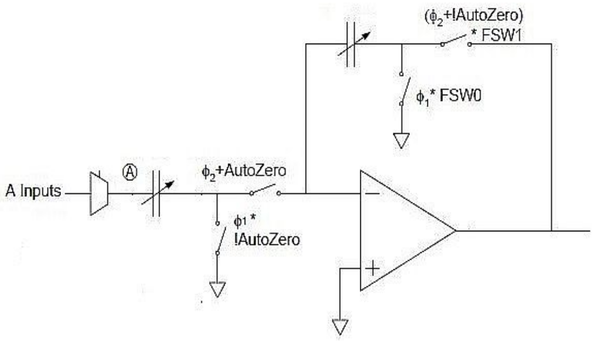


Figure 4. Detailed Circuit of PSoC OP Amp

TABLE 1. ADDITION SEQUENCE ON PSOC

step	Clocking		Input selection		Fig. 3 (b) clock
	Chip clock	FSW0	AMX_ IN bit	Pin selected	
1	Φ1	1(load)	00	P0[1]	$\phi 3$
	Φ2				$\phi 1$
2	Φ1	0(add)	10	P0[5]	-
	Φ2				$\phi 2$

First, in row 1 of step 1, feedback capacitor is discharged because FSW0 is 1 for Φ1, to reset operation. This corresponds to $\phi 3$ in Fig. 3 (b) (Upper Φ is chip clock, while lower ϕ is clock of Fig. 3 (b)). Next, in row 2 of step 1, voltage from P0[1], meaning pin 1 of Port 0 selected by “AMX_IN” of 00, is sampled by Φ2 and charged into the capacitors, serving as $\phi 1$ in Fig. 3 (b). Then, in row 2 of step 2, voltage from P0[5] selected by AMX_IN of 10 goes to the capacitors, corresponding to $\phi 2$ in Fig. 3 (b). This charge is *added* to the charge given in step 1 because FSW0 is 0.

Fig. 5 illustrates how to implement *asymmetric operation clocks* based on *symmetric chip clocks*.

FSW0 and AMux are operated as in Table 1, by an MPU inside the chip. Since MPU clock is different from switching clocks Φ1 and Φ2, code section to change FSW0 and AMux should be interrupt-driven by switching clock. Appendix A shows code details.

C. *Experimental Result*

A simple experiment is conducted to confirm this type of add operation. Fig. 6 shows SC amp output at top, two inputs P0[5] and P0[1] at center and bottom.

At top, voltages in the circle and the following, every other “Π”-shaped waveforms reflect the bottom input, and the ones in squares reflect the sum of center and bottom. This means that the output gives P0[1] first and P0[1]+P0[5] next, every other clock. The dashed lines connect squares and every other voltages, showing the sum of triangular and square input waves.

Thus, it was shown that, with symmetric clocks and DR, the SC amp can add two inputs every two clocks corresponding to the two steps in Table 1.

One problem with this method is, as shown in Fig. 6 top, that the output waveform represents the sum only discretely, mixed with one input appearing first.

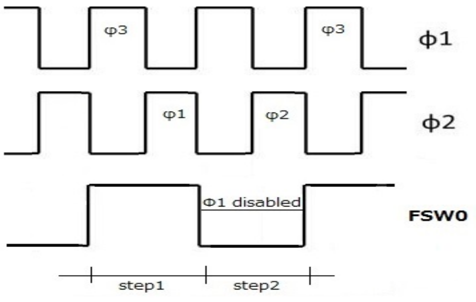


Figure 5. Φ1,2 with FSW0 to Implement ϕ 1-3

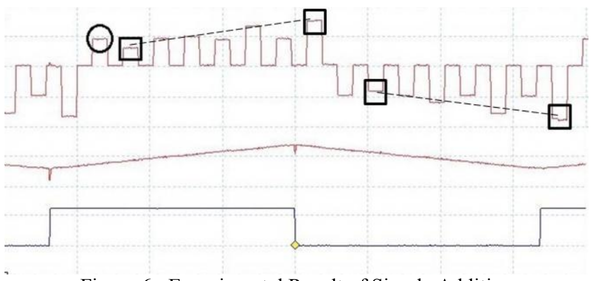


Figure 6. Experimental Result of Simple Addition

However, this can easily be solved by employing sample-and-hold, driven by every other clock.

IV. Other Operations and An Application to DAC

A. *Implementation of Subtraction*

In the case above, inputs P0[1] and P0[5] are *added,* because in steps 1 and 2, chip clocks to charge capacitors are the same Φ2. However, P0[5] can be *subtracted* from P0[1], or P0[1] from P0[5], by swapping clocks.

PSoC features additional circuitry shown in Fig. 7 at position Ⓐ of Fig. 4. "ASign" of 0 in Fig. 7 is normal, and if ASign is 1, operation of the same Table 1 offers -(P0[1]+P0[5])[1]. By changing this ASign dynamicly, subtraction looks possible, but ASign only is not enough.

As tabulated in Table 2, procedure for P0[1]-P0[5] actually needs *three* steps, instead of *two* for addition. It is because timing of pin reading is different depending on ASign as in the last column. Just after reading P0[5] at Φ2 in step 3, P0[1] should be read at Φ1 in step 1, but that is actually very difficult due to setup/hold timing constraint. Then, step 1 is dummy for interfacing between negative and positive, and step 2 actually serves

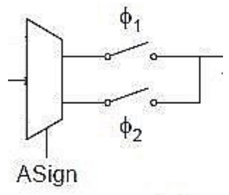


Figure 7. Clock Sign Selection Circuit

TABLE 2. Subtraction Sequence

step	Clocking		Sign	Input selection	
	Chip clock	FSW0	ASign	AMX_ IN bit	Pin read
1	Φ1	1(load)	0 (pos)	00	(P0[1])
	Φ2				-
2	Φ1				P0[1]
	Φ2				-
3	Φ1	0(add)	1 (neg)	10	-
	Φ2				P0[5]

[1] By making use of this functionality, noninverting amplifier is feasible with the "inverting" connection like Fig.4.

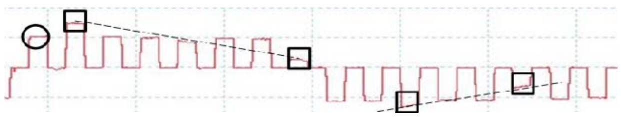


Figure 8. Subtraction Waveform

positive. Also, delays of Φ duration are needed and implemented by two delay loops in Appendix B.

Fig. 8 is the result of subtraction, where circle and squares are similar to the ones in Fig. 6, though the following circles appear twice during 3 clocks and squares are every third. Dashed lines connecting squares, showing P0[1]-P0[5], are correctly reverse in polarity of triangle wave in Fig. 6.

Therefore, subtraction is slower than addition, by a factor of 1.5.

B. *Weighted Adder*

Fig. 7 demonstrates the second DR property of PSoC. The third DR feature to be noted is two arrows over the capacitors in Fig. 4, meaning that capacity, or amp gain, can be programmed. This is done by "ACap" bits in the ASCXXCR0 register, to select one integer in range 0-31.

If they are changed dynamicly while selecting inputs by AMux bits as described in Sec. III *B*, operation like P0[1]+5*P0[5], for example, is possible.

These are among applications of DR functions of PSoC. Though PSoC DR is usually considered as more complex operation including "code reconfiguration" as described in [11], simpler ones to handle registers only is sometimes handy and useful, as found in [12].

C. *DAC Using Weighted Adder*

The method above can be extended to accommodate *n* bits, each with gain of 2^n, for example, to implement *n*-bit DAC. Table 3 shows the operation for 2-bit case, where P0[5] is MSB and P0[1] LSB. Though more input bits are possible based on this scheme, 2-bit case is shown intentionally in order to ease understanding with a simple situation. Appendix C shows the code details, which is an extension of code in Appendix A.

Fig. 9 is an experimental result for this DAC, where upper trace shows DAC output and lower LSB input driven by a counter. Counter counts from 0 to 3, with bit patterns 00, 01, 10 and 11. The 2-bit data for each value is passed to the adder with weights of 2 and 1. Taking the last two "Π" voltages for 11, for example, the first one gives output of 2 and the second gives 2+1=3.

Though the DAC output includes intermediate results, similar to Fig. 6, voltages of four "Π"-shaped parts in

TABLE 3. 2-bit DAC Operation Sequence

step	Enable	Gain	Input	
	FSW0	ACap	AMX_ IN bit	Pin selected
1	1(load)	16	10	P0[5]
2	0(add)	8	00	P0[1]

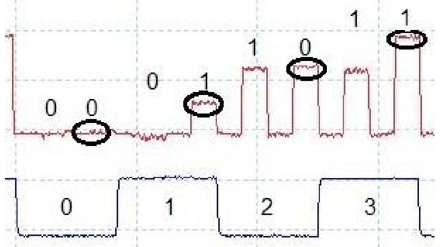


Figure 9. Experimental Result of 2-bit DAC

ellipses, the same as in Fig. 6, represent triangle wave with increasing levels of 0, 1, 2 and 3.

This DAC implementation should be compared with DAC macros included in PSoC development systems. For example, "DAC6" from Cypress needs only one SC amp, but should be driven by MPU. In contrast, our DAC features direct drivability from digital circuit, not relying on MPU, except controlling DR.

V. Conclusion

In order to implement analog operations area-efficiently with an SC technique, asymmetric clocking is proposed and dynamic reconfigurability of Cypress PSoC is made use of for realization.

Experimental results on simple adder, subtractor and DAC reveal reduction in the number of amps, but with some degradation in speed and output waveform.

This will contribute to the employment of "analog before ADC" into electronics, thus reducing part count and power.

Future investigation will include some other applications. A possibility is successive-approximation-register (SAR) ADC, by extending the proposed DAC, combined with a comparator.

References

[1] M. Verhelst and A. Bahai, "Where analog meets digital: Analog-to-information conversion and beyond", IEEE Solid-State Circuits Magazine, vol.7, no.3, pp.67-80, 2015

[2] Y. Huang, N. Guo, M. Seok, Y. Tsividis and S. Sethumad-havan, "Analog computing in a modern context: A linear algebra accelerator case study", IEEE Micro, vol.37, issue 3, pp. 30-38, 2017

[3] N.L. Pappas, *CMOS circuit design: Analog, digital, IC layout,* CreateSpace, 2014

[4] R. Vaidyanathan and J. Trahan, *Dynamic reconfiguration: architec- tures and algorithms*, Springer, 2003

[5] V. Vasudevan, "Dynamically reconfigurable analog front- end for ultrasonic imaging applications", IEEE Int. Ultra- sonics Symp., pp.1924-1927, 2014

[6] F. Kobayashi and S. Higuchi, "Analog dynamic reconfiguration for area-efficient implementation", Midwest Symp. CAS, 2011

[7] P.J. Quinn and A.H.M. van Roermund, *Switched-Capacitor Tech-niques for High-Accuracy Filter and ADC Design,* Springer, 1995

[8] M. Mar, B. Sullam and E. Blom, "An architecture for a config-urable mixed-signal device", IEEE J. Solid-State Circuits, vol.38, no.3, pp.565-568, 2003

[9] Anadigm Corp., dpASP products, http: //www.anadigm.com/ products.asp

[10] O.E. Gysel, P.J. Hurst and S. H. Lewis, "Highly programmable switched-capacitor filters using biquads with nonuniform internal clocks", 23rd IEEE International SOC Conference, pp.33-38, 2010

[11] A. Doboli, P. Kane and D. Van Ess, "*Dynamic reconfiguration in a PSoC device*", Int. Conf. Field- Programmable Tech., pp. 361-363, 2009

[12] cy.wbz (Cypress Corp.), "Amplifier with dynamic gain switching", PSoC 4 Pioneer Kit Community Project #94, https://www. element14.com/community/ thread/26850/, 2017

Appendix A Simple Addition Detail

Main.c (to select switching clock)

```
CLK_CR0 = 0x02; // use ACLK0
CLK_CR1 = 0x40; // ACLK0 from DBB 00, S/H off
CLK_CR2 = 0x00; // ACLK0 from Row 0 & 1
```

Interrupt.asm (to proceed steps by M8 MPU assembly)

```
   push a ; save A reg. for interrupt use
   inc  [step] ; step++
   mov  a,[step] ; if (step==1)
   and  a,0x01
   jnz  step2
   or   reg[ASC10CR3],0x10 ; FSW0=1 (load)
   mov  reg[AMX_IN],0x00 ; select P0[1]
   jmp  cont
step2: ; else
   and  reg[ASC10CR3],0xef ; FSW0=0 (add)
   mov  reg[AMX_IN],0x02 ; select P0[5]
cont:
   pop  a ; restore A to return from interrupt
```

Appendix B Assembly Interrupt Code for Subtraction

```
   push a ; save A reg. for interrupt use
   inc  [step] ; step++
   mov  a,[step] ; if (step!=3)
   cmp  a,3
   jz   step3
   mov  a,0x80 ; delay 50us
delay1:
   dec  a
   jnz  delay1
   or   reg[ASC10CR3],0x10 ; FSW0=1 (load)
   and  reg[ASC10CR0],0xdf ; ASign=0 (positive)
   mov  reg[AMX_IN],0x00 ; select P0[1]
   jmp  cont
step3: ; else
   mov  [step],0
   and  reg[ASC10CR3],0xef ; FSW0=0 (add)
   mov  a,0x80 ; delay 50us
delay2:
   dec  a
   jnz  delay2
   or   reg[ASC10CR0],0x20 ; ASign=1 (negative)
   mov  reg[AMX_IN],0x02 ; select P0[5]
cont:
   pop  a ; restore A to return from interrupt
```

Appendix C Assembly Interrupt Code for 2-bit DAC

```
   push a ; save A reg. for interrupt use
   inc  [step] ; step++
   mov  a,[step] ; if (step==1)
   and  a,0x01
   jns  step2
   or   reg[ASC10CR3],0x10 ; FSW0=1 (load)
   mov  reg[AMX_IN],0x02 ; select P0[5]
   mov  reg[ASC10CR0],0x10 ; ACap=16 (gain=2)
   jmp  cont
step2: ; else
   and  reg[ASC10CR3],0xef ; FSW0=0 (add)
   mov  reg[AMX_IN],0x00 ; select P0[1]
   mov  reg[ASC10CR0],0x08 ; ACap=8 (gain=1)
cont:
   pop  a ; restore A to return from interrupt
```