Kaijun Li
ECE 614 Spring 2008

Folding ADC Tutorial

Introduction:

The folding ADC employs folding techniques to reduce the number of comparators in Flash ADC. This tutorial will illustrate the basic operation of folding ADC in LTSpice using ideal components.

Basic Operation:

The basic idea of Folding ADC comes from Sub-ranging (or Two-step) ADC, which is aiming to reduce the amount of comparators and latches in Flash ADC while maintaining relatively higher conversion rate (speed) of the Analog-to-Digital converter. The operation of an N-bit folding ADC is depicted in Figure 1. From the diagram, we can see that the input signal V_{in} is fed into Coarse ADC and Folder separately. Also it is noticed that there are two options for implementing the sample-and-hold block, one of which is placing sample-and-hold right after the input signal and feeding the sampled input signal into Coarse ADC and Folder separately and the other is building the sample-and-hold block inside the Coarse ADC block and Folder block. In this tutorial, the latter method is used.

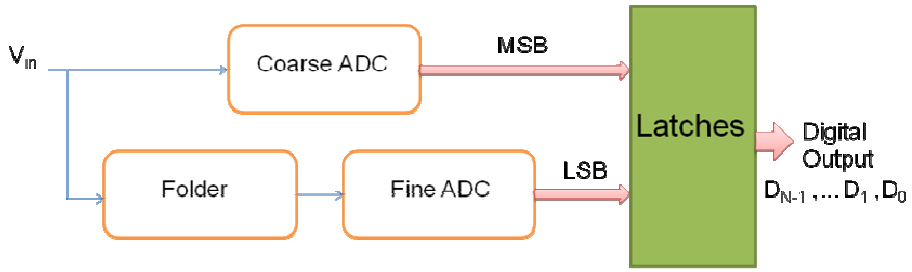

Figure 1: N-bit folding ADC

Then the task for Coarse ADC is to convert the input signal into digital outputs D_{N-1} …, $D_{N/2+1}$, $D_{N/2}$, which are the Most Significant Bits (MSBs) for the total digital outputs by assuming the same number of MSBs with Least Significant Bits (LSBs). In spice simulation, the algorithm used in Coarse ADC is based on pipeline ADC, which will be detailed later.

The same sampled input signal is also fed into the folder block and obviously the fine ADC will give us the LSBs of the total digital outputs. What the folder does is folding the input signal into $2^{N/2}$ which is shown in Figure 2 (also the same amount of MSBs and LSBs is assumed). From the figure, we know LSBs will be determined by the fine ADC based on which folding region the input signal resides in.

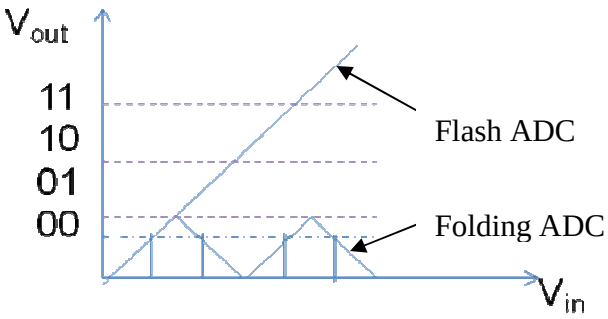

Figure 2: Transfer curve for 2-bit folder

About the timing issue, since coarse and fine ADCs are operating in parallel rather in series, there are less latency issues for Folding ADC compared to pipeline ADC or other types of ADCs.

Implementation in LTSpice

In spice simulation, V_{DD} of 5V will be used, and then V_{CM} would be 2.5V. A clock frequency is chosen as 100 MHz. Ideal components will be used in this session to simplify the design and increase simulation speed. One important issue for folding ADC is synchronization of coarse ADC and fine ADC, and otherwise glitches may appear on the reconstruction output signal.

In this session, both top-level blocks and low-level sub-blocked schematics will be described in great detail in order to give reader a better insight into the functionality of the folding ADC. Also, to make the simulations more understandable, a 4-bit folding ADC, which is composed of 2-bit coarse ADC and 2-bit fine ADC, is first developed to demonstrate the basic operation of folding ADC. Then based on the understanding of 4-bit folding ADC, a higher resolution (8-bit) folding ADC is built to provide with better resolution.

A. Top Level simulation files

sim_4bit.asc: The operation of a 4-bit folding ADC is shown in this file. The input (V_{in}) is chosen to be linearly increasing for 5 micro second changing from 0V to 5V. The output V_{outM} has resolution of 2-bit, while V_{out} has 4-bit resolution. What folder does can be seen by probing at V_{fold} which is folding input signal into 4 regions. Note that the folder output does not bounce around $V_{CM,}$ but we can achieve this by multiplying V_{fold} by the folding factor 4.

sim_8bit.asc: The operation of an 8-bit folding ADC is shown in this file. The input is also chosen to linearly change from 0V to 5V. The difference from previous one is its higher resolution, simply 8-bit, which means more complicated implementation of all blocks for the folding ADC. The input signal then is folded into 16 regions, and V_{fold} multiplied by 16 is bouncing around V_{CM}.

sim_8bit_sine.asc: This file simply shows the operation of an 8-bit folding ADC with input signal being sinusoidal waveform. The frequency of the sinusoidal input signal is selected as 1MHz, and the simulation period is then chosen to be 2 micro second which generates two periods of input signal.

B. Sub-block schematic files

B1. Coarse ADC Sub-blocks

2_bit_coase_ADC.asc: This sub-circuit shows the structure of the coarse ADC and is used in *sim_4bit.asc*. Since we are using the idea of pipeline ADC, the input signal is first fed into first comparator, which is ADCbit block that will be described shortly, and is compared to $V_{DD}/2$ to either subtract V_{CM} ($V_{in} > V_{DD}/2$) or nothing ($V_{in} < V_{DD}/2$). Then D_3 is obtained and the residue is amplified by two and passed into the second comparator. Similarly digital output D_2 is achieved. Note that V_{CM} is held for both comparators as switching point since the output voltage is amplified by a factor of 2 varying in full range.

4_bit_coase_ADC.asc: This sub-circuit shows the structure of the coarse ADC and is used in *sim_8bit.asc* and *sim_8bit_sine.asc*. Similar to the operation discussed in previous file, the digital outputs D_7 to D_4 can be obtained. Note that V_{CM} is held for each stage of the pipeline ADC's comparators as switching point since the output voltage of each stage is amplified by factor of 2 which gives a rail-to-rail varying signal.

Sample_and_hold.asc: This sub-circuit shows the sample_and_hold circuit built with switch capacitor implementation using ideal switches. The value of clock signal is compared to the switching point V_{trip}, which is $V_{DD}/2$, and therefore, for clock signal being low the switch S1 is turned on and the input is sampled, while for clock signal being high the switch S2 is turned on and the output value is held. Note that the input and output buffer are modeled using ideal op-amp.

ADCbit.asc: This sub-circuit shows the single bit circuit for pipeline ADC. V_{CM} is the switching point for the comparator, and depending upon the value of *Bitout*, which is either V_{DD} ($V_{in} > V_{DD}/2$) or 0 ($V_{in} < V_{DD}/2$), $2(V_{in}- V_{CM})$ when *Bitout* is V_{DD} or $2V_{in}$ when *Bitout* is 0 will be passed to next stage.

B2. DAC Sub-blocks for Signal Reconstruction of MSBs

ideal_2_bit_dac.asc: This sub-circuit shows the structure of the ideal 2-bit DAC. Each digital input is compared V_{trip}, which is $V_{DD}/2$, and *BiL* (i=0,1) can be obtained. Then we can the analog output by simply combining these two binary-weighted voltages. Note that V_{one} is set to be 1, so that means the *BiL* is digital value, and to accomplish the conversion from digital to analog the combination of digital value with binary-weighted will be multiplied by 1 LSB.

Ideal_4_bit_DAC.asc: This sub-circuit shows the structure of the ideal 4-bit DAC. Similar to the operation discussed in previous file, the digital outputs D_7 to D_4 can be obtained.

B3. Folder Sub-blocks

2bit_Folding_ADC.asc: This sub-circuit shows the structure of the 2-bit folding ADC. The function of this block is to fold the input signal into 4 regions and then use the fine ADC to get LSBs. The details about how folder and fine ADC work will be left to the following descriptions of each individual block. The sample_and_hold block is put out there to get the sampled input signal. In addition, the reference for the fine ADC is set to be $V_{DD}/4$ because the folding signal varies between 0V and $V_{DD}/4$. Note that the signals *Reg1~ Reg4* are used to help fine ADC determine which region the folding signal resides in, which will be detailed in the fine ADC block.

2bit_folder.asc, Folder_ADCbit.asc: These sub-circuits are constructed to realize the functionality of the 2-bit folder. To describe them together is due to the concern of description's consistency and ensuring better understanding of the folder. Basically, sampled input is fed into each Folder_ADCbit block. In Folder_ADCbit block, to determine if the input signal reside in this folding region ($V_{THL}<V_{in}<V_{THH}$), two comparators' outputs are used by setting their switching points as V_{THL} and V_{THH}. If $V_{THL}<V_{in}<V_{THH}$, *Bitout1* will be V_{DD} and *Bitout2* will be 0V, which are the maximum and minimum of each comparator's output, so we can compare the value of 1.1 times their difference, which is *Bitout*, to V_{THH} to testify if input signal is in this region. If it does, S3 will be closed, and otherwise S4 will be on, and also note that factor 1.1 is selected to avoid contention state for V_{THH} being V_{DD}. The signal V_{ctr} is used to control which signal is output of the folder when input signal is in this folding region, either V_{in} or V_{THH}-V_{in}. V_{ctr} is chosen as 0V for the odd numbered folders (Folder 1 and 3) since we want V_{in} to be output, while as $1.1V_{DD}$ for even numbered folders (Folder 2 and 4) since V_{THH}-V_{in} is the desired output. Note that there can be only one folding region the folding signal resides in, so by adding up all the folder's output we get the folding signal V_{fold}.. In addition, the signals Reg1 ~ Reg4, which is *Bitout* in Folder_ADCbit block, will be passed to the fine ADC to help digitize the folding signal V_{fold}.

2_bit_fine_ADC_*LSB*.asc, 2bit_PG.asc: This sub-circuit shows the structure of the 2-bit fine ADC which determines the LSBs. With folding signal V_{fold} being the input INSH of this block, it is fed into . Note that is no longer $V_{DD}/2$, but rather is $V_{ref\_fine}/2$ which is half the new reference for fine ADC with value of $V_{DD}/8$ for 2-bit fine ADC. Using the pipeline ADC to determine folding signal's digital codes and its complementary codes, we need to decide whether to pass out its original digital codes or complementary ones with the assistance of *Reg1 ~ Reg4* coming from the folder block. In 2bit_PG block, we can see if the input signal resides in the first or third region, the original codes *B1_org* and *B0_org* will be passed to the output *B1* and *B0*, otherwise the complementary codes *B1i* and *B0i* will be passed.

4bit_Folding_ADC.asc, 4bit_folder.asc, 4_bit_fine_ADC_*LSB*.asc, 4bit_PG.asc: The same analysis can be applied, and the only difference is that the number of blocks increases.

B4. Analog Signal Reconstruction Sub-blocks

4_bit_register.asc, 8_bit_register.asc, DFF.asc: These sub-circuits show the structure of the 4-bit and 8-bit registers implemented with D Flip-Flop.

Ideal_4_bit_DAC.asc, Ideal_8_bit_DAC.asc: These sub-circuits show the structure of the ideal 4-bit and 8-bit DAC. Similar to the operation discussed in the previous file *Ideal_2_bit_DAC.asc*, the digital outputs $D_3 \sim D_0$ or $D_7 \sim D_0$ can be obtained.

Conclusion:

This tutorial is aiming to illustrate the basic operation of the folding ADC. The folding ADC adopts the idea of sub-ranging (or two-step) ADC by getting MSBs from a coarse ADC and LSBs from a fine ADC. A simple 4-bit folding ADC was built in LTSpice and function of each block is documented in great detail to provide insights into folding ADC.

In addition, to demonstrate how higher resolution folding ADC can be implemented using the same structure, an 8-bit folding ADC was built. All the simulations files are provided for readers to play with and hopefully get the idea of how the folding ADC operates.

References

[1] R. J. Baker, *CMOS Circuit Design, Layout, and Simulation,* Revised Second Edition, Wiley-IEEE, 2008

[2] http://webcast.berkeley.edu/EE 247

[3] R. van de Plassche, *Integrated Analog-to-Digital and Digital-to-Analog Converters*, Second Edition, Kluwer, 2003.

[4] http://www.analog.com

[5] Willy M.C. Sansen, *Analog Design Essentials*, Springer, 2006