A Compiled 9-bit 20 MS/s 3.5 fJ/conv.step SAR ADC in 28 nm FDSOI for Bluetooth Low Energy Receivers

Carsten Wulff, *Member, IEEE*, Trond Ytterdal, *Senior Member, IEEE*

***Abstract*—This paper presents a low power 9-bit compiled successive-approximation register (SAR) analog-to-digital converter (ADC) for Bluetooth low energy receivers. The ADC is compiled from a SPICE netlist, a technology rule file, and an object definition file into a design rule check (DRC) and layout versus schematic (LVS) clean layout and schematic in 28 nm FDSOI. The compiled SAR ADC reduces the design time necessary to port to a new technology, and to demonstrate technology porting the same SAR ADC architecture is compiled in 28 nm FDSOI with IO transistors. This work also includes a comparator clock generation loop that uses the bottom plate of the capacitive digital-to-analog converter. The proposed compiled core transistor SAR ADC achieves state-of-the-art FoM of 2.7 fJ/conv.step at 2 MS/s, and 3.5 fJ/conv.step at 20 MS/s with an area of 0.00312 mm^2.**

Carsten Wulff (Corresponding author), carsten.wulff@ntnu.no, +4793200714, Trond Ytterdal, trond.ytterdal@ntnu.no, Department of Electronics and Telecommunications, NTNU, 7491 Trondheim, Norway.

***Index Terms*—analog layout, analog layout synthesis, ADC, analog-to-digital conversion, FDSOI, successive approximation, low-power, Bluetooth low energy**

I. Introduction

A low-power, medium resolution analog-to-digital converter (ADC) is a key building block in Bluetooth® low energy [1] receivers. Low power in the ADC is essential, as it can take up a large portion of the receiver power budget, especially since two ADCs are usually required in receivers with quadrature down-conversion; one for the real branch, and one for the imaginary branch.

In a Bluetooth® low energy receiver the design time of the ADC can be high compared to design time of the anti-alias filter, low-noise amplifier, and mixer. Reducing design time of ADCs is an active research area, and part of this research is analog layout generation.

One of the challenges with analog layout generation is the large solution space. A schematic designer has freedom to choose from any architecture, any combination of transistors, and a large number of transistor widths and lengths. The schematic alone is not sufficient to fully determine the layout. For example, the schematic does not contain placement information for layout, since the optimum layout placement might be different from optimal schematic placement.

Analog layout generation has a long history with works from the previous century [2], [3], but state-of-the-art analog layout generation, as reviewed in [4], is not widely adopted. More promising research avenues avoid the challenge of analog layout generation from schematics, by not having a schematic. Recently, ADCs have been compiled in a digital flow [5], [6], and although the ENOB was less than 6-bit, it is an interesting approach. A similar approach has been used successfully for all-digital PLLs [7].

This work, first introduced at a conference [8], presents a method where the layout is not generated based on drawn schematics. The ADC is described using an approach borrowed from object oriented programming. A custom compiler is used to compile the ADC. The ADC is described in a SPICE netlist, an object definition file, and a technology rule file. The compiler outputs a schematic and layout in SKILL[1] and GDSII. The output can be loaded into Cadence Virtuoso and a standard design flow can be used for parasitic extraction, simulation, and verification.

To reduce the solution space, the presented method uses a limited set of circuit blocks, a low complexity ADC architecture, and proposes a technology independent method for generating a design rule check (DRC) and layout versus schematic (LVS) clean schematic and layout in multiple technologies. Technology independence in this work has been modeled with a tapeout of one ADC in 28 nm FDSOI with core transistors, and another in 28 nm FDSOI with IO transistors.

To demonstrate that an ADC can be compiled is not sufficient to displace traditional analog design methodology. A compiled ADC is of limited use unless it can

[1]SKILL is the Lisp variant script language used in Cadence Virtuoso

be demonstrated that the performance can match state-of-the-art ADCs. To the best of the authors knowledge, this work presents the most efficient compiled ADC and achieves a FoM comparable to the state-of-the-art [9]–[11] (as surveyed in [12]) with a smaller area, making it perfectly suited for Bluetooth® low energy receivers.

The remainder of this paper is organized as follows: Section II describes the SAR ADC architecture, while Section III describes the compiler. Section IV describes the design methodology using the compiler. Measurement results are shown in Section V and conclusions given in Section VI.

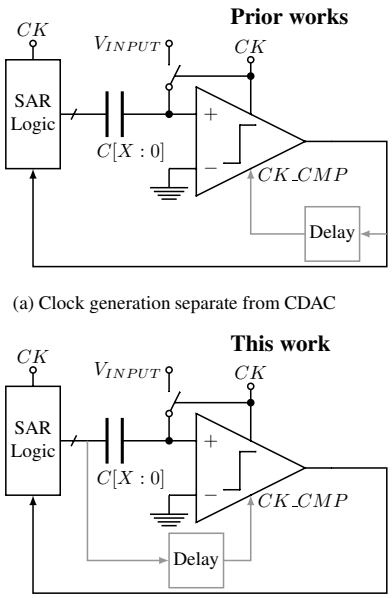
II. SAR ADC Architecture

The successive-approximation register (SAR) ADC architecture proposed in this work includes a technique that uses the bottom plate of the capacitive digital-to-analog converter (CDAC) in the bit-cycling clock generation loop. Two clocks are commonly used in SAR ADCs. One to set the sample rate, which in this work is applied externally to the ADC, and one bit-cycling clock, which can be generated locally in the ADC. A typical SAR ADC architecture can be seen in Fig. 1(a), with a comparator, SAR logic, and the CDAC. The comparator is forced into reset during sampling ($CK = 1$), and it triggers when the sample clock turns off. It is common to use detection methods to determine when the comparator has made a decision, and use this to trigger the subsequent resets of the comparator. Such self-timed loops, shown in Fig. 1(a), are common in prior art [9], [13]–[15].

One of the challenges in a self-timed loop is to allow sufficient time for the CDAC to settle. Each time the comparator makes a decision, one (or more) of the capacitors are switched and the resulting transient must settle to the accuracy of the ADC. It is possible to use circuit techniques like delay cells [9], [13], [14], or replica delays [15] to adjust the time between comparator decisions. Common to these techniques is that they try to model the required CDAC settling time, and allow enough time before the comparator makes the next decision. In this work, we use the bottom plate of the CDAC directly in the comparator clock generation loop to improve robustness, and ensure sufficient time for CDAC settling, as shown in Fig. 1(b).

The SAR ADC architecture used to demonstrate the compiler is shown in Fig. 2(a). It has NMOS boot-strapped input switches, a strong-arm comparator with kick-back compensation [16], and a metal-oxide-metal finger CDAC.

Sampling of the differential input signal ($V_P - V_N$) is controlled by the sample clock CK. The sample clock



(a) Clock generation separate from CDAC

(b) Clock generation including CDAC

Fig. 1. Comparator clock generation in: (a) Prior works, (b) This work.

has a duty-cycle of less than 25 % to increase the time available for the SAR algorithm. The first SAR logic block ($LOGIC[8]$) is enabled when $CK = 0$, the next logic block ($LOGIC[7]$) is enabled when $LOGIC[8]$ has completed and sets $EO = 1$. The bit-cycling continues until $EO = 1$ for the last logic block ($LOGIC[0]$).

The comparator clock (CK_CMP) is generated by the loop that starts with the comparator outputs (P and N) into the SAR logic blocks and out from the CO output. A pulse signal is generated internally in the SAR logic, and the CI and CO form a chain of OR gates. The digital output from each stage is D_{P1}.

The bottom plate of the CDAC capacitors are controlled directly by the D_{P0}, D_{P1}, D_{N0} and D_{N1} signals, which switch between ground and the reference voltage. The reference voltage is at the same voltage as the supply voltage. The first five stages of the ADC use split monotonic switching [17] to reduce the common mode variation, while the last four stages use monotonic switching [18]. In a monotonic based scheme, an 8-bit CDAC is sufficient. The SAR ADC in this work is intended as building block for noise-shaped SAR ADCs [14], where the last residue is needed, thus it uses a 9-bit CDAC.

The compiled ADC uses unit transistors for all blocks in the design, as explained below. One unit transistor size is used for both PMOS and NMOS. The *W = 258 nm* and *L = 30 nm* for the core transistors, and *W = 344 nm* and *L = 180 nm* for the IO transistors. A small unit transistor was chosen to reduce the power consumption

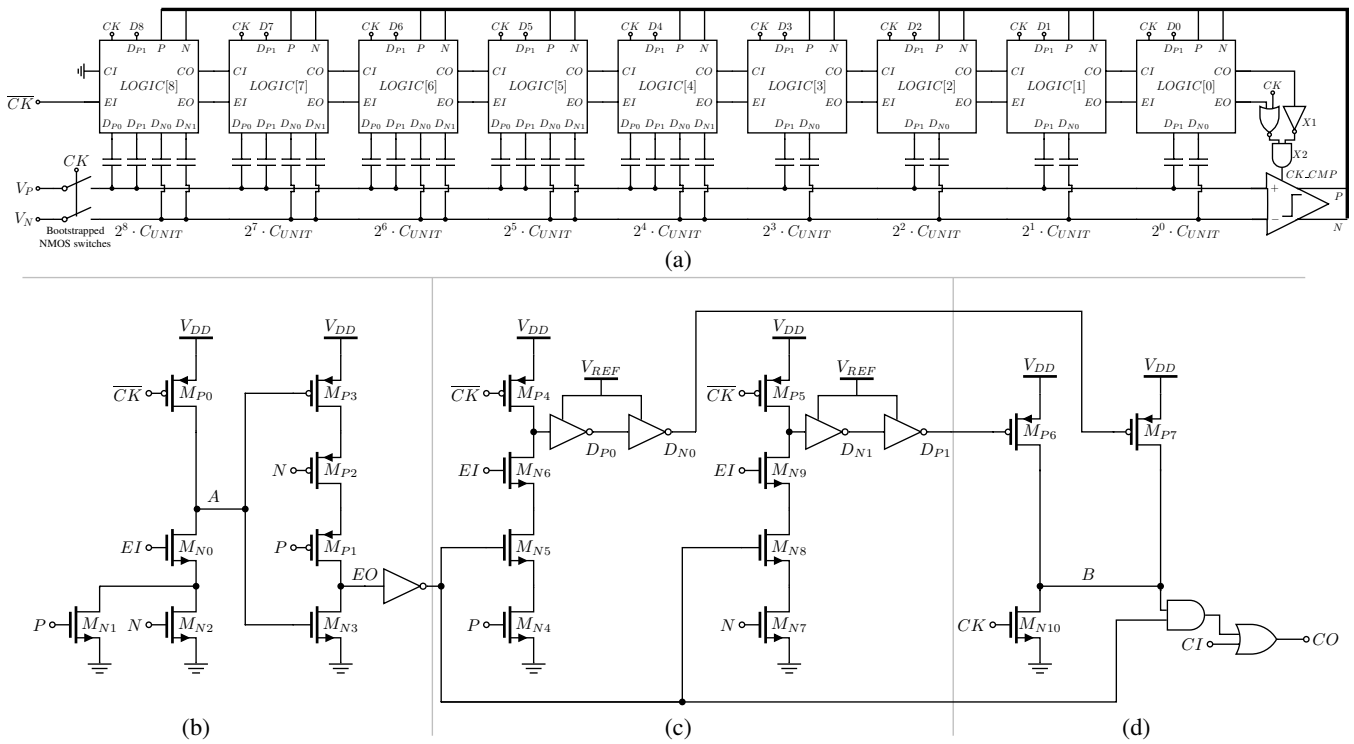


Fig. 2. Proposed SAR ADC architecture with: (a) Block diagram, (b) Enable logic, (c) CDAC state control, (d) Clock generation.

in the comparator and digital logic. The schematic and sizing of the comparator [16] is shown in Fig. 3. The boot-strapped switch transistor has four unit transistors in parallel.

The SAR logic consists of three parts: enable logic (Fig. 2(b)), CDAC state control (Fig. 2(c)), and clock generation loop (Fig. 2(d)). During sampling of the ADC input the SAR logic is reset ($CK = 1$). In the enable logic of the first stage $EI = \overline{CK} = 0$, and node $A = 1$, while $EO = 0$. Thus $EI = EO = 0$ of all subsequent stages. The CDAC state control has $D_{P0} = D_{N1} = 0$ while $D_{N0} = D_{P1} = 1$. In the clock generation loop of the first stage $CI = 0$, and since node $B = 0$, then $CO = 0$. Accordingly $CI = CO = 0$ for all subsequent stages.

The first comparator decision is initiated by $CK = 0$ (steps 1 and 2 in Fig. 4). At that point the first latch ($M_{N0} - M_{N2}, M_{P0}$) in the enable logic is armed, and as soon as $(P||N) = 1$, then $A = 0$. This arms the second latch ($M_{N3}, M_{P1} - M_{P3}$) in the enable logic. Still $EO = 0$. The CDAC state control is also triggered by $(P||N) = 1$ (step 3 in Fig. 4).

The advantage of the bit-cycling clock generation loop in this work, is that it includes the transition of D_{P1} and D_{N0}. Since either D_{P1} or D_{N0} is guaranteed to transition from high to low, these signals can be used to trigger comparator reset. When either D_{P1} or $D_{N0} = 0$ then M_{P6} or M_{P7} turns on, and sets node $B = 1$, and consequently $CO = 1$ (step 4 in Fig. 4).

The comparator in Fig. 3 is reset when $CK\_CMP = 0$, which occurs when $CO = 1$, since $CK = 0$, and for the last stage $EO = 0$ (steps 5 and 6 in Fig. 4). The comparator will set signals $P = N = 0$, which turns on transistors M_{P2} and M_{P1}, and sets $EO = 1$ (step 7). This enables the next stage, and locks the state of the CDAC state control, since M_{N5} and M_{N8} turn off. Also, $CO = 0$ and in the end $CK\_CMP = 1$ (step 8 and 9 in Fig. 4), which clocks the comparator once more, and the next bit is decided. The bit-cycling ends when $EO = 1$ for the last stage.

III. Compiler

The key contribution in this work is to speed up the design time of SAR ADCs, and reduce the effort necessary to port a design to another technology. Schematic capture, simulation, layout, and parasitic netlist simulation, is a time consuming endeavor for ADCs. For charge based SAR ADCs the small unit capacitance make them sensitive to poor layout. It is common to spend time tracking down fF parasitics, and redoing layout. The design time would be shorter if one could generate DRC/LVS clean layout in minutes, instead of days. Rapid ADC generation would allow for more iterations before tapeout, and could even allow automatic exploration of the design space. It could also enable rapid porting to another technology.

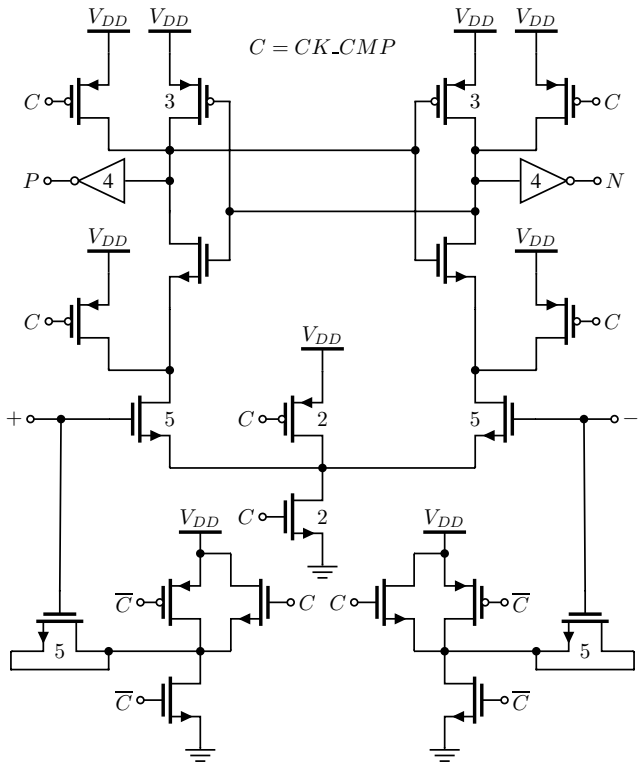


Fig. 3. Strong-arm comparator with kick-back compensation. Transistors without numbers are unit size, while transistors with numbers are parallel combinations of unit transistors.

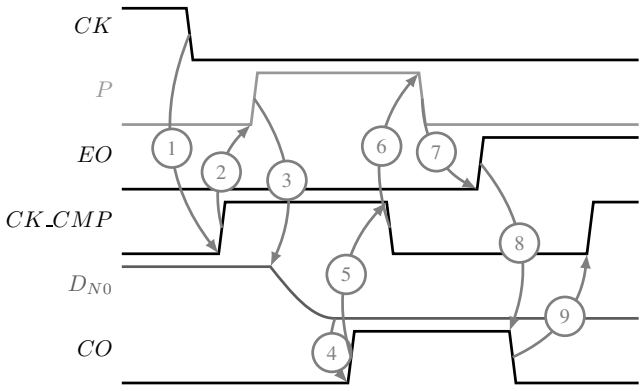


Fig. 4. Timing diagram of the SAR logic.

The compiler in this work (a Perl script) compiles a core transistor SAR ADC into GDSII in 2.7 seconds on a Macbook Air with 1.7GHz Intel Core i7. The Perl version of the compiler is closed source, but there is a reduced feature set open source version written in C++ [19], which compiles a SPICE netlist, an object definition file, and a rule file of a core transistor SAR ADC into GDSII in 0.35 seconds.

The SPICE netlist does not contain transistor lengths or transistor widths, rather it contains references to unit transistors. The unit transistor can be series connected, or parallel connected.

The object definition file is written in JavaScript Object Notation (JSON) [20], a common text data format, which is supported in most programming languages. The object definition file, which contains the definition of the unit transistor, does not contain absolute dimensions, thus it is technology independent. The object definition file also defines the routing of blocks in the ADCs. Absolute dimensions, and technology specific information are defined in a rule file.

The remainder of this Section is organized as follows, the rule file is described in Section III-A, while the unit transistor is described in Section III-B, and the unit capacitor in Section III-C. The place and route of the design is described in Section III-D, while the top level SAR ADC layout is described in Section III-E.

A. Technology rule file

The rule file contains the GDSII layer numbers, GDSII data-type, layer material definitions, what cuts (vias) to use for transitions between layers, and design rules.

All rules, like poly width, cut size, metal width, metal spacing and metal cut enclosure are based on a dimensionless parameter $GAMMA$. The layout rules in this work are inspired by early work on dimensionless layout rules, or LAMBDA rules, where the LAMBDA parameter was equal to half the minimum feature size ($F/2$) [21]. The dimensionless rule approach was further extended to nano-scale technologies with $GAMMA = F/4$ in [22], [23].

In this work the $GAMMA$ has been redefined to $F/6$ to allow better granularity of the layout rules.

In the rule file for 28 nm FDSOI there are 140 unique rules, but 95 of those are for the different metals and cuts (spacing, width, enclosure).

The compiler can automatically insert cuts between layers, and to build the cuts the material type (active, cut, metal, poly), and the connect stack (PO to CO to M1, M1 to VIA1 to M2), is defined in the rule file for each of the layers.

B. Unit transistor

The structure of the object definition file is inspired by object oriented programming, where all objects are instances of a class. The class is defined in the compiler, and the compiler has classes for transistors, capacitors, resistors, digital cells and complex cells (CDAC, SAR ADC).

The layout rules for nano-scale transistors contain complex spacing rules. In the 28 nm FDSOI design rule deck there are close to 5000 design rules. Not all these rules relate to transistors, but it is complex to design a

DRC clean programmable transistor cell that can support multiple technologies, from multiple foundries. Instead of a complex programmable transistor cell this work implements a simplified transistor cell. The layout of the transistor can be seen in Fig. 5.

To avoid complex rules the rule file defines two numbers, the vertical grid size and the horizontal grid size. For the ADCs in 28 nm FDSOI the grids are the same (*86 nm*), but two numbers are defined in the compiler for future flexibility. The compiler also supports modification of the grid on a per cell basis, and multiple unit transistors in a design.

The grid size controls the spacing between poly gates, the active enclosure of cuts, the metal enclosure of cuts, the dummy poly to active spacing and the poly to cut spacing. To improve manufacturability the transistor has two cuts for drain, source and poly, regular poly pitch to improve poly critical dimension (CD) and enlarged metal one rectangles to satisfy minimum metal area DRC rules. There are vertical routing channels between active and gate, and gate and bulk contact to simplify routing. On neighboring transistors the dummy poly can be overlapped to ensure regular poly pitch over large distances.

The unit transistor is described in the object definition file as an ASCII code as shown in Fig. 6. The ASCII has a NxM matrix where each cell is one grid in size. A $-$ character means empty space, x means the grid is filled. m means the grid is filled horizontally, but the height of the rectangle is the minimum routing width for that layer. For w the grid is also filled horizontally, but the height of the rectangle is the normal routing width for that layer. K, C, and c make cuts to the next layer. D, G, S, B are the familiar ports of the transistor.

The PMOS and NMOS have the same physical dimensions, thus they differ only by their respective implant layers and well definitions. The compiler supports the object oriented programming concept of inheritance, where an object can inherit all features of a parent. One example is the PMOS in Fig. 6, which inherits the DMOS. The function *addEnclosures* searches the parent cell and finds the OD rectangles, adds P-implant layer, and low threshold voltage layer with enclosures defined in the rule file. This object oriented design approach allows complex cells to be constructed from small modifications of existing cells.

To make the unit transistor DRC clean requires an iterative process where the transistor layout is compiled, DRC checks are run, and grid sizes adjusted until the transistor is DRC clean. With this simplified transistor there is a risk that it will not fit all technologies, but so far DRC clean transistors have been compiled in 28 nm FDSOI, 28 nm, 65 nm and 55 nm.

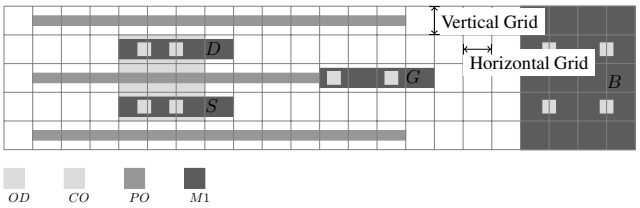


Fig. 5. Unit transistor layout.

```
// Unit transistor definition
{ "name" : "DMOS" ,
  "class" : "Gds::GdsPatternTransistor",
  "fillCoordinatesFromStrings" : [
      [   "OD",
          "-----------------xxxx",
          "----xxK----------xCxC",
          "----xxx----------xxxx",
          "----xxK----------xCxC",
          "-----------------xxxx"
      ],
      [   "PO",
          "-mmmmmmmmmmmmmm--------",
          "----------------------",
          "-mmmmmmmmmmcxc---------",
          "----------------------",
          "-mmmmmmmmmmmmmm--------"
      ],
      [   "M1",
          "-----------------xxxx",
          "----wDww---------xxxx",
          "----------wGww---xBxx",
          "----wSww---------xxxx",
          "-----------------xxxx"
      ]
  ]
},
// PMOS definition
{ "name" : "PCHDL",
  "inherit" : "DMOS",
  "beforePlace" :{
    "addEnclosures" : [
    ["OD",1,["PP","LVT"]],
    ["OD",0,["PP"]]
    ]
  }
}
```

Fig. 6. Object definition of the unit transistor (DMOS) and the PMOS (PCHDL).

C. Unit capacitor

The building block for the 9-bit CDAC is a 5-bit CDAC cell, with an additional unit capacitor to make 32 unit capacitors. A 3D view [24] can be seen in Fig. 7(a), and the cell in Fig. 7(b). The bottom plate is a metal finger in metal four and metal three. The capacitor top plate surrounds the bottom plate fingers in metal four. The bottom plate routing, in metal one, is covered by a ground shield in metal two to reduce the parasitic coupling to the top plate. The compiler has a built-in class to make this cell, and the class gets the metal width, capacitor finger spacing and metal route spacing from the rule file.

The four MSB in the 9-bit CDAC uses 15 cells (bit 8 = 8 cells, bit 7 = 4 cells, bit 6 = 2 cells, bit 5 = 1 cell). Bit 4 uses one cell, but half the capacitors are connected

to ground. Bit 3, bit 2, bit 1 and bit 0 share a cell, thus in total 17 cells are used in the 9-bit CDAC. The netlist for the IO ADC with the detailed CDAC cell arrangement is available at [19].

The total capacitance seen from the top plate node of the 9-bit CDAC is simulated at 169 fF. The parasitic capacitance to ground is simulated at 67 fF. The remaining 102 fF is the combination of the 512 unit capacitors. The unit capacitor is simulated at 0.2 fF in 28 nm FDSOI.

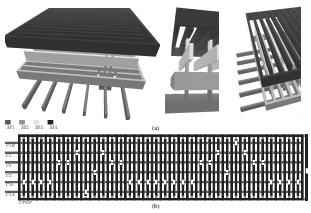


Fig. 7. Layout of the 5-bit CDAC cell used in the 9-bit CDAC with: (a) A 3D view, (b) The 32 unit capacitor cells.

D. Place and Route

The layout of the enable logic from Fig. 2(b) can be seen in Fig. 8, with a 3D view [24] in Fig. 8(a). The enable logic layout in Fig. 8(d) has been compiled from the input files into LaTeX [25] with *cic2tikz* [19] to improve figure readability, thus it does not include the implant layers, or well definitions.

The placement of instances in the layout is determined by the order of instances in the SPICE netlist (Fig. 8(b)). The instance name (i.e. $MN0$) is used to group elements. It is the group name, defined as the characters up to the first number (i.e MN), that determines whether row or column number is incremented. The first transistor, $MN0$, is placed in column zero and row zero, as seen in Fig. 8(d). If the next instance has the same group name the row number is incremented. Thus all MN transistors will be placed on top of each other. If the next instance has a different group name, for example MP, the row number is reset and the column number is incremented. For example, instance $MP0$ will be placed in row zero and column one, while $MP1$ to $MP3$ are placed on top of $MP0$.

The compiler does not contain an auto-router. It contains routing instructions to route circuits, thus a designer has to specify exactly the necessary routes for the circuit blocks. The compiler does, however, contain instructions to find ports, perform simple routes, and do layer stack transitions.

The routes are defined in the object definition file, and the object definition of the enable logic is shown in Fig. 8(c). The enable logic is an object of a class that has support functions for digital cells. For example, the class automatically adds routes in metal four for VDD and VSS, as seen in Fig. 8(a). Only the outline is shown for the VDD and VSS in Fig. 8(d) to make the metal one routing visible.

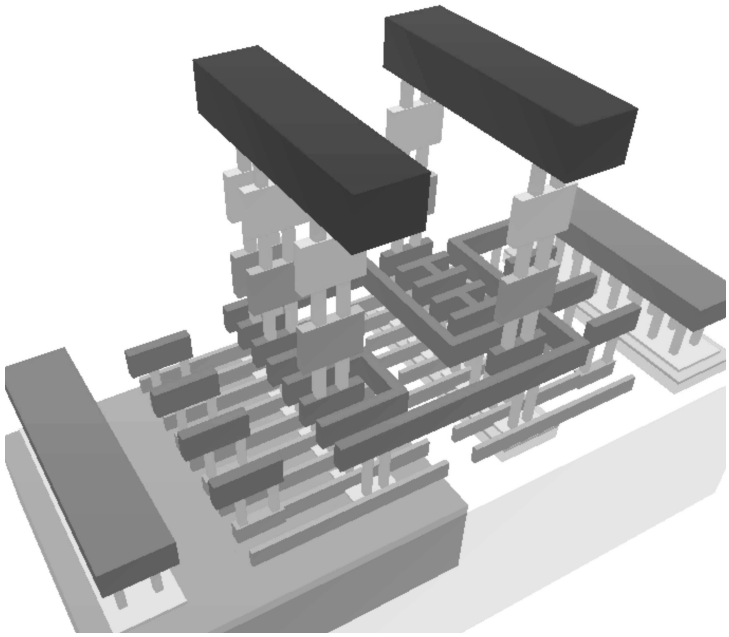
There are two types of route instructions in Fig. 8(c), a connectivity route, and a directed route. A connectivity route can search the circuit for instance ports, and find all instance ports that belong to a net name. The net name match is performed with regular expressions [26]. Route number 3 in Fig. 8(c) is a connectivity route for net EO between M_{N3} and M_{P1}. The instructions for this route are `['M1','EO','--|-','onTopR']`, which translates to: Find all instance ports in metal one (`'M1'`) that match the regular expression `'EO'`, use rightmost port as the start of the route (`'onTopR'`), and route in a left, up or down, left pattern (`'--|-'`) to the other ports. This forms the route marked with 3 in Fig. 8(d).

For the net A in Fig. 8(b) the connectivity route is not sufficient, as a connectivity route would route from the gate of $MP3$ to the gate of $MN3$ in metal one, thus creating a short. The directed route is used in these cases to customize the route. For net A there are three directed routes (route number 6, 7, 8). The instructions for route number 6 is `['PO','A','MN3:G-MP3:G']`, which translates to: Find instance `'MN3'`, check if the instance has a port called `'G'` in poly, and route in poly (`'PO'`) horizontally to an instance with name `'MP3'` and port `'G'`.

The complete object definition file contains object definitions for all the blocks in the ADC. Most of the lines in the object definition file are route instructions, and it requires effort from an analog designer to define the routes of a complete ADC, but this is a one time effort. Once the routes are defined they can scale to multiple technologies without change. The reason it scales is because route information is void of technology information.

E. Top level SAR ADC layout

The object definition of the SAR ADC top level uses a SAR ADC class in the compiler. The layout placement in this class is controlled by the SPICE netlist, but the placement algorithm is different from the digital cells. The SAR ADC class assumes that any instance in the SPICE netlist with a group name $XCDAC$ is a CDAC. Any instance with group name XB is assumed to be a input switch, and instances with group name XA are digital cells. Based on these assumptions the compiler places the input switches at the bottom, CDAC in the



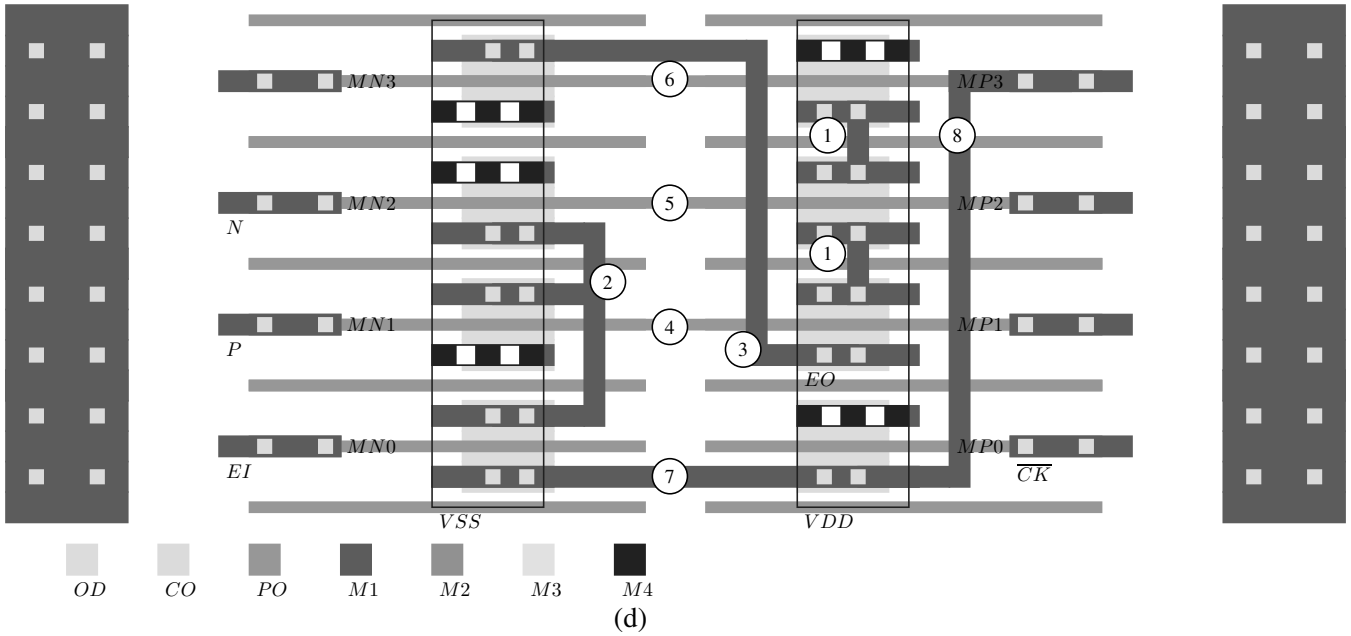
(a)

```
.SUBCKT SAREMX1_CV P  N EI EO CK_N AVDD AVSS
  MN0 N3   EI   A    AVSS  NCHDL
  MN1 N3   P    AVSS AVSS  NCHDL
  MN2 AVSS N    N3   AVSS  NCHDL
  MN3 EO   A    AVSS AVSS  NCHDL
  MP0 AVDD CK_N A    AVSS  PCHDL
  MP1 N2   P    EO   AVSS  PCHDL
  MP2 N1   N    N2   AVSS  PCHDL
  MP3 AVDD A    N1   AVSS  PCHDL
.ENDS
```

(b)

```
{ "name": "SAREMX1_CV",
  "class" : "Layout::LayoutDigitalCell",
  "addConnectivityRoutes": [
      ["M1","N1|N2","||",""],      1
      ["M1","N3","-|",""],         2
      ["M1","EO","--|-","onTopR"]  3
        ],
    "addDirectedRoutes" : [
      ["PO","P","MN1:G-MP1:G"],    4
      ["PO","N","MN2:G-MP2:G"],    5
      ["PO","A","MN3:G-MP3:G"],    6
      ["M1","A","MN0:S-MP0:S"],    7
      ["M1","A","MP0:S-|--MP3:G"]  8
      ]
    }
}
```

(c)



(d)

Fig. 8. The enable logic with: (a) 3D layout, (b) SPICE netlist (c) Object definition (d) Layout.

middle, digital cells at the top. A compiled layout of an IO transistor SAR ADC can be seen in Fig. 9(a), and a core transistor SAR ADC in Fig. 9(b). The layout in Fig. 9 was compiled from the input files into Encapsulated Postscript (EPS) [27] with *cic2eps* [19] to improve figure readability, thus it does not include cuts, implant layers, or well definitions.

To port the SAR ADC from core transistors to IO transistors required some changes to the compiler inputs. The metal spacing was changed in the rule file, which is why the CDAC in the IO ADC is taller than in the core ADC. The unit transistor was redefined, and PMOS and NMOS were modified to change the implant layers. There were no changes to the SPICE netlist. Less than 5 % of the lines in the input files required changes to compile the IO transistor ADC from the core transistor ADC input files.

IV. Design Methodology

The ADC design in this work started with an architecture exploration using hand analysis and schematics, as shown in Fig. 10. Compiled cells (5-bit CDAC,

Fig. 9. Top level layout for: (a) An IO SAR ADC, (b) A Core SAR ADC.

transistors) were used as schematic building blocks for more complex cells (9-bit CDAC, comparator). When a suitable architecture was found, the initial netlist was modified to control placement of instances in the layout, and the routing instructions implemented. As the design progressed, more and more of the ADC was compiled. In the end, the ADC was fully described by the SPICE netlist, object definition file and technology file. The architecture schematic was no longer needed, although it was kept for most circuits to visualize the implemented architecture. Testbenches and extracted parasitic netlist simulations were used to verify the physical implementation. It was assumed that the physical verification was sufficient to verify the compiler code. The complete object definition file that was used to generate the core transistor ADC in this work consists of 1266 lines of code, but that file generates 21 different SAR ADC variants, with 8-bit, 9-bit, 10-bit and 11-bit CDACs. The design effort for the initial compiled SAR ADC is estimated at 400 hours, including development of the compiler code.

It is reasonable to expect that some redesign must be done after the initial compile, or indeed porting to another technology. The advantage of this work for design and redesign, is that changes to the circuit can be done in minutes, and recompiled in seconds. For example, changing the width of the unit transistor, as shown in Fig. 11(b). This allows the analog designer to start simulation on extracted netlist early in the design flow, and primarily do the optimization for performance, process, voltage and temperature on extracted netlist.

The analog designer can also quickly adopt to new design requirements. A noise-shaped SAR ADC (11-bit ENOB simulated) [28] was designed with a variant of the 9-bit core ADC in this work as a building block. In a noise-shaped SAR ADC the thermal noise sampled on the CDAC is not shaped by the loop filter, thus [28] required more CDAC capacitance. An example of a 9-bit SAR ADC variant with more CDAC capacitance, is shown in Fig. 11(c).

In the authors experience, there is sufficient support in the compiler classes to place and route most analog circuits, but the compiler can only use cells that it generates. It cannot use standard-logic cells, or layout cells drawn with traditional analog design methodology. The compiler was also used for support circuits outside the ADCs. A digital input pad circuit, shown in Fig. 12, uses the standard compiler classes. Some analog circuits may require custom compiler classes, and a combination of analog design expertise and programming expertise is needed to code the classes.

In [28] compiled cells were used as building blocks, similar to the architecture exploration in this work, but the top level layout was drawn in Cadence Virtuoso. The advantage of the compiler for complex analog circuits, for example Bluetooth® low energy receivers, is that key sub-blocks can be quickly compiled. As a result, the analog designer can focus the design effort on the difficult parts.

V. Measurement Results

The micrograph can be seen in Fig. 13(a), and the die measures 1.04 mm $\times$ 1.04 mm without seal-ring. The layout is shown in Fig. 13(b), with the IO transistor ADC and core transistor ADC locations indicated by arrows. On this die there are seven other compiled SAR ADCs with 8-bit, 10-bit and 11-bit resolutions. The prototype is fabricated in a 28 nm FDSOI process with an area of 39 μm x 80 μm = 0.00312 mm^2 for the core transistor SAR ADC, and 40 μm x 106 μm = 0.00424 mm^2 for the IO transistor SAR ADC, including logic, comparator, CDAC, and input switch.

The input signal to the ADCs was supplied from a R&S SML signal generator through a 5th-order passive band-pass filter, and the input frequency was selected for coherent sampling. A balun was used to convert from single ended to differential, and the balun common mode voltage was supplied externally through a resistive divider. The sample clock was supplied from a second R&S SML signal generator at twice the ADC sample rate, and an on-chip sine-to-square circuit with a pulse-picking divide-by-two was used to generate the 25 % duty-cycle. The parallel output data was captured with an RTE1022 oscilloscope with mixed-signal option. The data was post processed in MATLAB, and a Hanning window was used before the FFT.

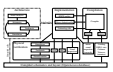


Fig. 10. Design methodology

Fig. 11. Layout variants of the core ADC with: (a) The original core ADC, (b) A core ADC with wide transistors ($1.72\mu m$), (c) A core ADC with more CDAC capacitance.

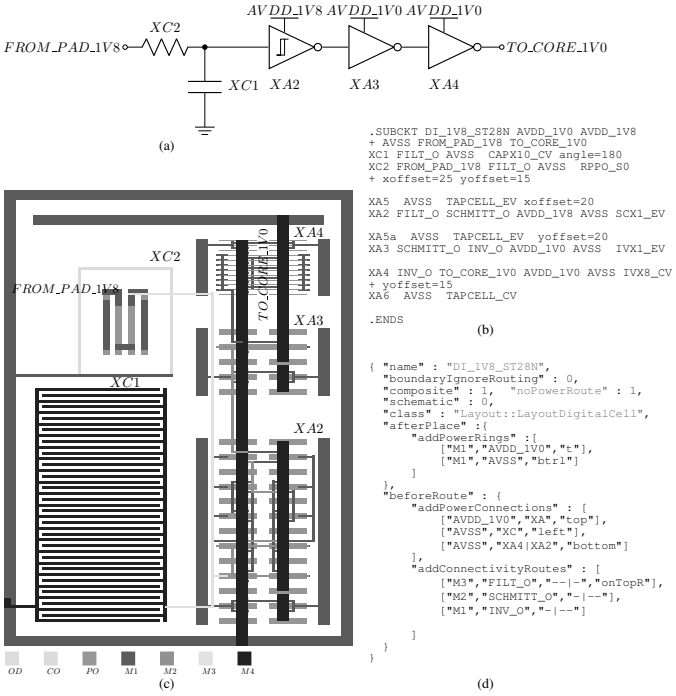


Fig. 12. Digital input pad circuit with: (a) Schematic, (b) SPICE netlist, (c) Layout, (d) Object definition.

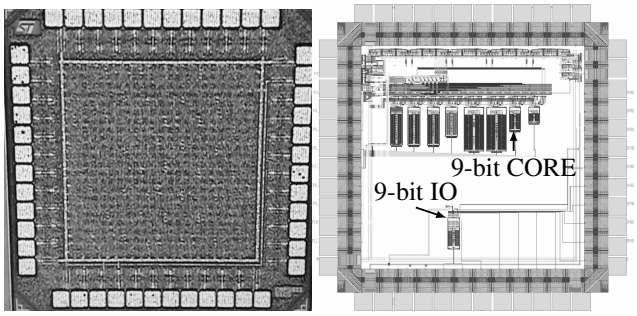


Fig. 13. Overall layout with: (a) Micrograph, (b) Chip Layout.

Three chips have been measured for the core transistor SAR ADC with similar performance. The core transistor SAR ADC in this work achieves a FoM[2] of 3.5 fJ/conv.step at 20 MS/s, shown in Fig. 14(a), and 2.7 fJ/conv.step at 2 MS/s (Fig. 14(b)) placing it among the most power efficient and area efficient ADCs for its sample rate and resolution.

The ENOB of the core ADC is limited by thermal noise. The comparator is the likely source of the thermal noise, as the ENOB can be changed by varying the input common mode voltage. The core transistor SAR ADC is functional from 80 kS/s - 80 MS/s and operates from a single 0.4 V - 1.1 V supply voltage, as shown in Fig. 14(c). The SNDR and SFDR as a function of frequency can be seen in Fig. 14(d). The measured power consumption of the core ADC is 15.87 μW at 20 MS/s and 0.94 μW at 2 MS/s. The measured power consumption of the core ADC CDAC is approximately 44 % at 20 MS/s and approximately 47 % at 2 MS/s. The simulated power consumption of the core ADC CDAC is 42 % at 20 MS/s and 44 % at 2 MS/s. The simulated power consumption of the comparator is 25 % at 20 MS/s and 23 % at 2 MS/s.

The spectrum of the IO transistor SAR ADC is shown in Fig. 15.[3] Table I shows a summary and comparison of key measurements with prior-art.

VI. Conclusion

A low power 9-bit compiled SAR ADC has been presented with a comparator clock generation loop that uses the bottom plate of the CDAC. The ADC was compiled from text input files into a DRC/LVS clean layout and schematic in 28 nm FDSOI. The proposed ADC achieves a FoM of 3.5 fJ/conv.step at 20 MS/s with an area of 0.00312 mm^2, and demonstrates that a compiled SAR ADC can achieve state-of-the-art performance.

Acknowledgment

Partial funding for this work was obtained from the Norwegian PhD Network on Nanotechnology for Microsystems, which is sponsored by the Research Council of Norway, Division of Science, under contract no. 221860/F60. The authors thank L. Lewyn and H. Garvik for technical discussions, and Nordic Semiconductor and D. Engelien-Lopes for donation of time to perform this work.

[2] $FoM = P/(2^{ENOB} \times fs)$

[3] Due to limitations for the IO SAR ADC it is not possible to separate the current in pad drivers, and ADC current. The load on each pin is at least 4pF, as specified on the RTE1022 oscilloscope. Accordingly the current for the IO SAR ADC is estimated at ADC current = 542 μA - average(bit transitions x 10 MHz x 4 pF x 1.6 V).

TABLE I
Comparison to prior art.

	Weaver [5]	Harpe [9]	Patil [10]	Liu [11]	This work	
Technology (nm)	90	90	28 FDSOI	28	28 FDSOI	
Fsample (MS/s)	21	2	No sampling	100	2	20
Core area (mm^2)	0.18	0.047	0.0032	0.0047	0.00312	
SNDR (dB)	34.61	57.79	40	64.43	46.43	48.84
SFDR (dBc)	40.81	72.33	30	75.42	61.72	63.11
ENOB (bits)	5.45	6.7 - 9.4	6.35	10.41	7.42	7.82
Supply (V)	0.7	0.7	0.65	0.9	0.47	0.69
Pwr (μW)	1110	1.64 -3.56	24	350	0.94	15.87
Compiled	Yes	No	No	No	Yes	
FoM (fJ/c.step)	838	2.8 - 6.6	3.7	2.6	2.7	3.5

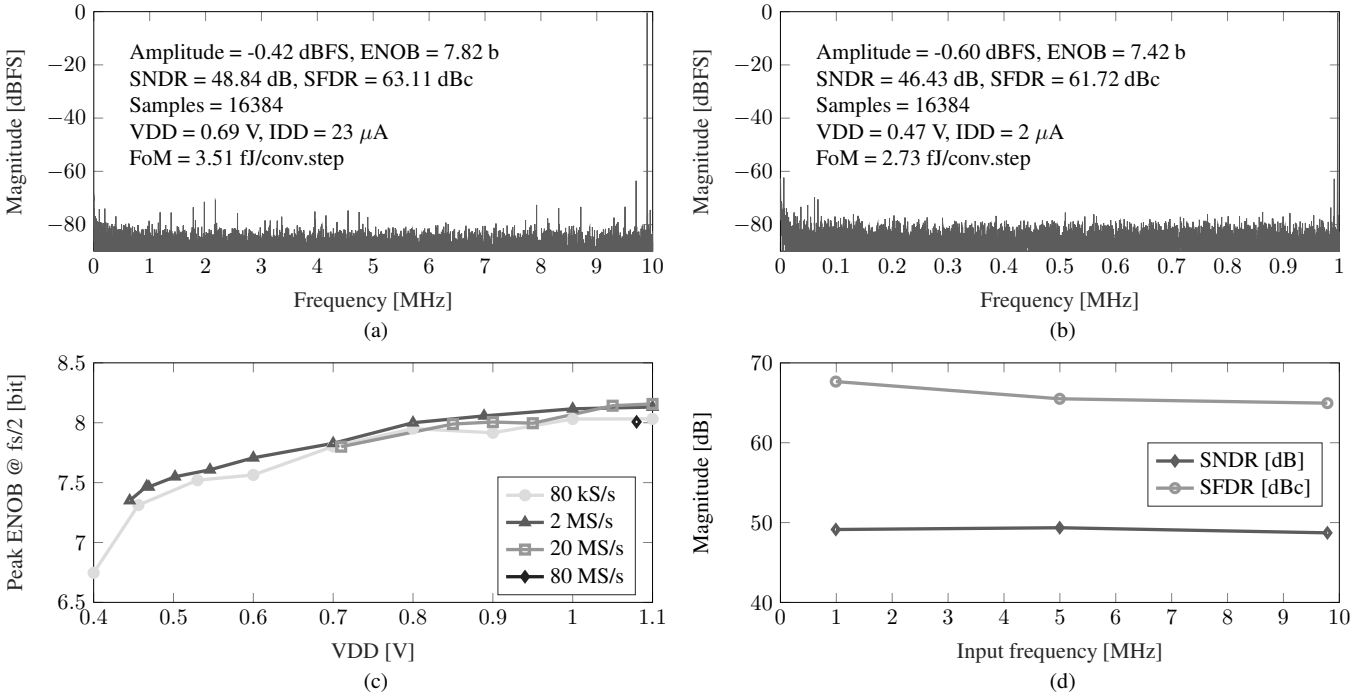


Fig. 14. Measurements for core transistor SAR ADC with: (a) Spectrum at 20 MS/s, (b) Spectrum at 2 MS/s, (c) Peak ENOB as a function of supply voltage, (d) SNDR and SFDR as a function of input frequency at 20 MS/s.

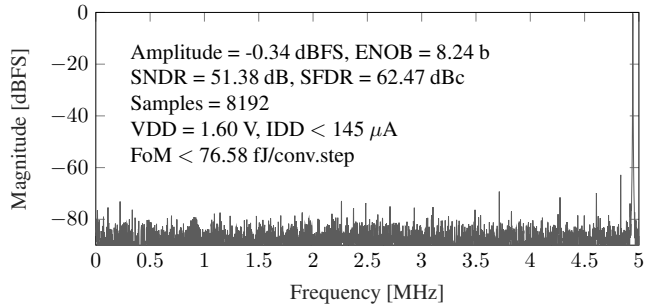


Fig. 15. Spectrum of IO transistor SAR ADC at 10 MS/s.

REFERENCES

[1] (2016) Bluetooth Low Energy, Bluetooth. [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy

[2] J. Rijmenants, J. B. Litsios, T. R. Schwarz, and M. G. R. Degrauwe, "ILAC: an automated layout tool for analog CMOS circuits," *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 417–425, Apr 1989.

[3] J. D. Bruce, H. W. Li, M. J. Dallabetta, and R. J. Baker, "Analog layout using ALAS!" *IEEE J. Solid-State Circuits*, vol. 31, no. 2, pp. 271–274, Feb 1996.

[4] R. Martins, N. Loureno, and N. Horta, "LAYGEN II - Automatic Layout Generation of Analog Integrated Circuits," *IEEE Trans. Comput.- Aided Design Integr. Circuits Syst.*, vol. 32, no. 11, pp. 1641–1654, Nov 2013.

[5] S. Weaver, B. Hershberg, and U. K. Moon, "Digitally Synthesized Stochastic Flash ADC Using Only Standard Digital Cells," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 84–91, Jan 2014.

[6] A. Fahmy, J. Liu, T. Kim, and N. Maghari, "An All-Digital Scalable and Reconfigurable Wide-Input Range Stochastic ADC Using Only Standard Cells," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 62, no. 8, pp. 731–735, Aug 2015.

[7] W. Deng, D. Yang, T. Ueno, T. Siriburanon, S. Kondo, K. Okada, and A. Matsuzawa, "A Fully Synthesizable All-Digital PLL With Interpolative Phase Coupled Oscillator, Current-Output DAC, and Fine-Resolution Digital Varactor Using Gated Edge Injection Technique," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 68–80, Jan 2015.

[8] C. Wulff and T. Ytterdal, "A compiled 3.5fJ/conv.step 9b 20MS/s SAR ADC for wireless applications in 28nm FDSOI," in *Proc. 42nd Eur. Solid-State Circuits Conf. (ESSCIRC)*, 2016, pp. 177–180.

[9] P. Harpe, G. Dolmans, K. Philips, and H. de Groot, "A 0.7V 7-to-10bit 0-to-2MS/s flexible SAR ADC for ultra low-power wireless sensor nodes," in *Proc. 38th Eur. Solid-State Circuits Conf. (ESSCIRC)*, 2012, pp. 373–376.

[10] S. Patil, A. Ratiu, D. Morche, and Y. Tsividis, "A 3-10 fJ/conv-step Error-Shaping Alias-Free Continuous-Time ADC," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 908–918, April 2016.

[11] C. C. Liu, "A 0.35mW 12b 100MS/s SAR-assisted digital slope ADC in 28nm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, 2016, pp. 462–463.

[12] B. Murmann. (2016) ADC Performance Survey 1997-2016. [Online]. Available: http://web.stanford.edu/~murmann/adcsurvey.html

[13] P. Harpe, B. Busze, K. Philips, and H. de Groot, "A 0.47-1.6 mW 5-bit 0.5-1 GS/s Time-Interleaved SAR ADC for Low-Power UWB Radios," *IEEE J. Solid-State Circuits*, vol. 47, no. 7, pp. 1594–1602, 2012.

[14] J. A. Fredenburg and M. P. Flynn, "A 90-MS/s 11-MHz-Bandwidth 62-dB SNDR Noise-Shaping SAR ADC," *IEEE J. Solid-State Circuits*, vol. 47, no. 12, pp. 2898–2904, Dec 2012.

[15] R. Kapusta, J. Shen, S. Decker, H. Li, E. Ibaragi, and H. Zhu, "A

14b 80 MS/s SAR ADC With 73.6 dB SNDR in 65 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 48, no. 12, pp. 3059–3066, Dec 2013.

[16] A. Varzaghani, A. Kasapi, D. N. Loizos, S.-H. Paik, S. Verma, S. Zogopoulos, and S. Sidiropoulos, "A 10.3-GS/s, 6-Bit Flash ADC for 10G Ethernet Applications," *IEEE J. Solid-State Circuits*, vol. 48, no. 12, pp. 3038–3048, Dec 2013.

[17] C. C. Liu, S. J. Chang, G. Y. Huang, Y. Z. Lin, and C. M. Huang, "A 1V 11fJ/conversion-step 10bit 10MS/s asynchronous SAR ADC in 0.18 μm CMOS," in *Proc. IEEE Symp. VLSI Circuits*, June 2010, pp. 241–242.

[18] C. C. Liu, S. J. Chang, G. Y. Huang, and Y. Z. Lin, "A 10-bit 50-MS/s SAR ADC With a Monotonic Capacitor Switching Procedure," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 731–740, April 2010.

[19] C. Wulff. (2016) Custom IC creator. [Online]. Available: https://github.com/wulffern/ciccreator

[20] D. Crockford. (2016) JavaScript Object Notation. [Online]. Available: http://www.json.org

[21] C. Mead and L. Conway, *Introduction to VLSI Systems.*, 1st ed. Reading, MA: Addison-Wesley, 1979.

[22] L. L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin, "Analog Circuit Design in Nanoscale CMOS Technologies," *Proc. IEEE*, vol. 97, no. 10, pp. 1687–1714, Oct 2009.

[23] L. Lewyn and N. Williams, "Is a New Paradigm for Nanoscale Analog CMOS Design Needed?" *Proc. IEEE*, vol. 99, no. 1, pp. 3–6, Jan 2011.

[24] University of Twente, Integrated Circuit Design Group. (2016) GDS3D, Interactive 3D Layout Viewer for GDSII. [Online]. Available: https://sourceforge.net/projects/gds3d/

[25] The LaTeX Project. (2016) LaTeX A document preparation system. [Online]. Available: https://www.latex-project.org

[26] J. E. F. Friedl, *Mastering Regular Expressions*, 2nd ed., A. Oram, Ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2002.

[27] Adobe Systems Incorporated. (1992) Encapsulated PostScript File Format Specification, Version 3.0. [Online]. Available: http://partners.adobe.com/public/developer/en/ps/5002.EPSF_Spec.pdf

[28] H. Garvik. (2015) An energy efficient noise-shaping SAR ADC in 28 nm FDSOI, Master thesis. [Online]. Available: http://hdl.handle.net/11250/2371464

Carsten Wulff received the M.Sc. and Ph.D. degrees in electrical engineering from the Department of Electronics and Telecommunication, Norwegian University of Science and Technology (NTNU), in 2002 and 2008, respectively. During his Ph.D. work at NTNU, he worked on open-loop sigma-delta modulators and analog-to-digital converters in nanoscale CMOS technologies. In 2006-2007, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. He is currently the Group Manager for the Wireless Group at Nordic Semiconductor ASA, Trondheim, Norway and a Post Doctoral fellow at NTNU. His present research interests includes analog and mixed-signal CMOS design, design of high-efficiency analog-to-digital converters and low-power wireless transceivers. He is the developer of Custom IC Compiler, a general purpose integrated circuit compiler.

Trond Ytterdal received his M.Sc. and Ph.D. degrees in electrical engineering from the Norwegian Institute of Technology in 1990 and 1995, respectively. He was employed as a research associate at the Department of Electrical Engineering, University of Virginia (1995-1996) and as a research scientist at the Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute in Troy, New York (1996-1997). From 1997 to 2001 he worked as a senior ASIC designer at Nordic Semiconductor in Trondheim, Norway. Since 2001 he has been on the faculty of the Norwegian University of Science and Technology (NTNU), where he is a Professor at the Department of Electronics and Telecommunications. Prof. Ytterdal's present research interests include design of analog integrated circuits, behavioral modeling and simulation of mixed-signal systems, modeling of nanoscale transistors and novel device structures for application in circuit simulators. He has authored and co-authored more than 200 scientific papers in international journals and conference proceedings. He is a co-author of the books Semiconductor Device Modeling for VLSI (Prentice Hall, 1993), Introduction to Device Modeling and Circuit Simulation (Wiley, 1998) and Device Modeling for Analog and RF CMOS Circuit Design (Wiley, 2003), and has been a contributor to several other books published internationally. He is also a co-developer of the circuit simulator AIM-Spice. Prof. Ytterdal is a member of The Norwegian Academy of Technological Sciences and a Senior Member of IEEE.