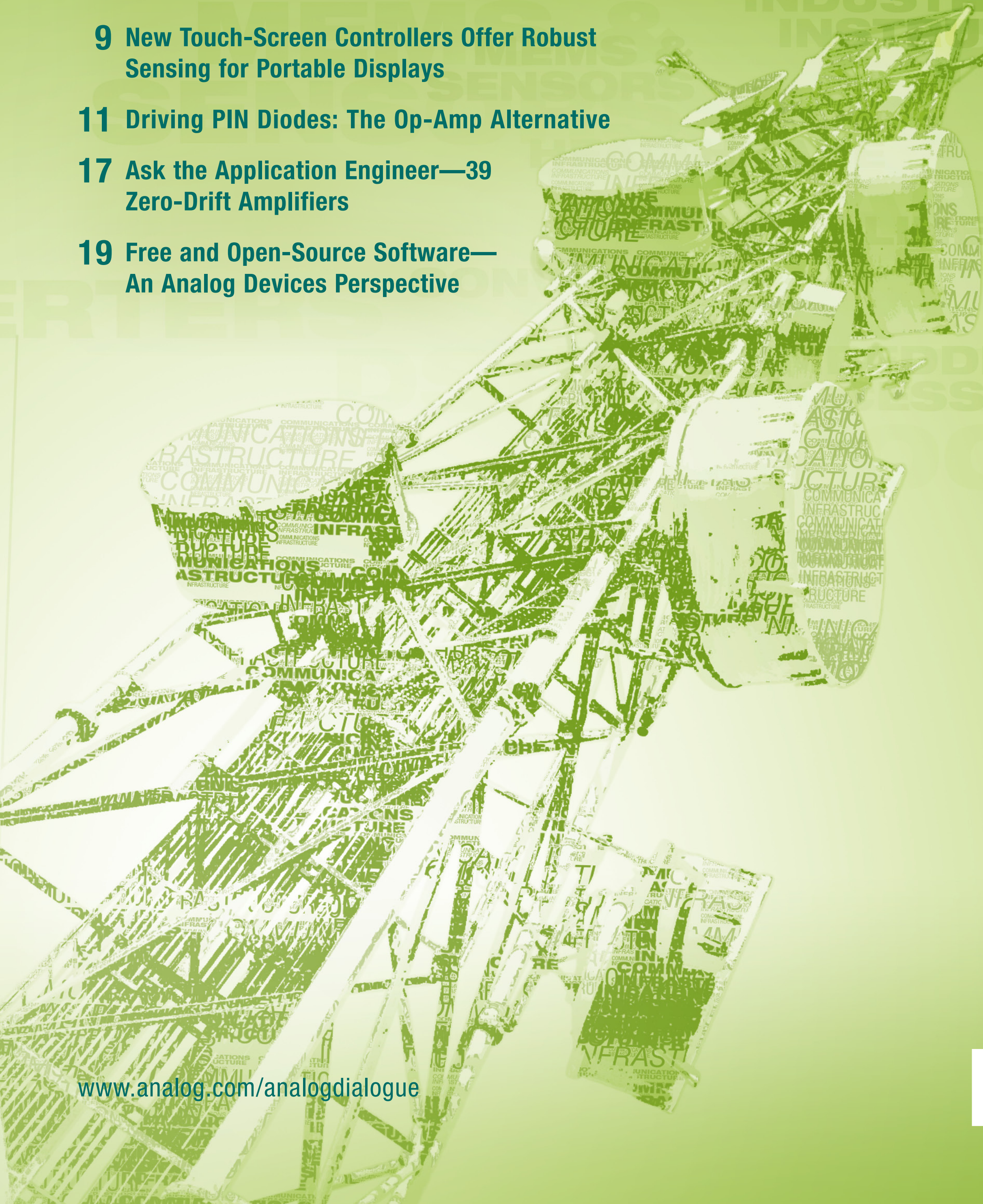
Volume 44, Number 1, 2010

A forum for the exchange of circuits, systems, and software for real-world signal processing

Analog Dialogue

In This Issue



www.analog.com/analogdialogue

ANALOG DEVICES

Editors' Notes

IN THIS ISSUE

Termination of High-Speed Converter Clock Distribution Devices

When using clock distribution devices or clock fanout buffers to clock ADCs and DACs, two main sources of signal degradation need to be dealt with: PCB traces behave like low-pass filters, attenuating clock signals and distorting clock edges as they travel along the trace; and reflections can cause undershoot and overshoot, severely degrading the signal and the overall clock performance. Page 3.

ADIsimPower Provides Robust, Customizable DC-to-DC Converter Designs

Designers of dc-to-dc converters are faced with an overwhelming number of options for power management ICs. Finding the best combination of features, performance, integration level, and price can be difficult enough, and the actual design work can be daunting. ADIsimPower™ simplifies the IC selection process and provides the information required to build an optimized dc-to-dc converter. Page 5.

New Touch-Screen Controllers Offer Robust Sensing for Portable Displays

Touch-screen displays are replacing mechanical buttons in smart phones, MP3 players, navigation systems, laptop computers, and other devices. First-generation devices suffered from low accuracy, false detection, and high power consumption. New devices—which offer improved accuracy, lower power consumption, and result filtering—can also sense temperature, supply voltage, and touch pressure. Page 9.

Driving PIN Diodes: The Op-Amp Alternative

PIN diodes, which sandwich a lightly doped intrinsic region between heavily doped P and N regions, are used extensively in RF and microwave applications. PIN diode drivers—which provide a controlled forward bias current and reverse bias voltage—use discrete designs or specialized ICs. As an alternative, widely available op amps can be used. Op amps in this class feature wide bandwidth, high slew rate, and enough steady-state current to drive PIN diodes. Page 11.

Ask the Applications Engineer 39—Zero-Drift Operational Amplifiers

Zero-drift amplifiers dynamically correct their offset and reshape their noise density. Two commonly used types—auto-zero and choppers—achieve nanovolt-level offsets and extremely low drift. 1/f noise is seen as a dc error, so it is removed as well. In addition, zero-drift amplifiers have higher open-loop gain, power-supply rejection, and common-mode rejection than standard amplifiers. Page 17.

Free and Open-Source Software—An Analog Devices Perspective

The rapid increase in use of free and open-source software (FOSS) represents a significant long-term trend. FOSS licenses make source code available and grant developers the right to study, change, and improve the design. FOSS, already playing a role in every major software category from 64-bit servers to 8-bit microcontrollers, will fundamentally change the nature of software for all users and developers. Page 19.

Dan Sheingold [dan.sheingold@analog.com]

Scott Wayne [scott.wayne@analog.com]

PRODUCT INTRODUCTIONS: VOLUME 44, NUMBER 1

Data sheets for all ADI products can be found by entering the part number in the search box at www.analog.com.

January

ADCs, sigma-delta, 12-/16-bit, low-power **AD7170/AD7171**
Codec, audio, stereo 24-bit, 96-kHz, low-noise **ADAU1781**
Converter, dc-to-dc, step-down, 3-A/6-A outputs **ADP2116**
Mixer, balanced, dual, 500-MHz to 1700-MHz **ADL5358**
Mixer, balanced, dual, 1200-MHz to 2500-MHz **ADL5802**
Receiver, HDMI, low-power **AD9393**
Regulator, buck, dual, 3-MHz, 600-mA **ADP5022**
Transceivers, RS-485, 250-kbps, full-duplex **ADM488A/ADM489A**

February

Amplifier, operational, quad, low-power, wideband ... **ADA4691-4**
Controller, QWERTY keypad, mobile I/O expander .. **ADP5587**
Detector, TruPwr, 450-MHz to 6000-MHz **ADL5504**
Microcontrollers, precision analog, 12-bit, ARM7TDMI **ADuC7023/ADuC7029**
Sensor, vibration, digital, programmable **ADIS16220**

March

ADC, pipelined, dual, 16-bit, 20-MHz/40-MHz/65-MHz/80-MHz **AD9269**
ADC, sigma-delta, dual, 16-bit, continuous-time **AD9262**
Amplifier, operational, quad, low-power, precision **AD8624**
Driver, isolated half-bridge, 4-A output **ADuM7234**
Energy Meter, polyphase active- and reactive power ... **ADE7878**
Generator, network clock, two-output **AD9575**
Isolator, digital, 5-channel, 1-kV rms isolation **ADuM7510**
Mixer, active, 1550-MHz to 2150-MHz **ADRF6602**
Modulator, quadrature, 950-MHz to 1575-MHz ... **ADRF6750**
Processors, Blackfin, consumer-device connectivity .. **ADSP-BF51x**
Regulators, very-low-dropout, 500 mA **ADP124/ADP125**
Sensor, inertial, six-degrees-of-freedom **ADIS16367**
Transceiver, RF, dual, WiMAX/BWA/WiBro/LTE **AD9356**

Analog Dialogue

Analog Dialogue, www.analog.com/analogdialogue, the technical magazine of Analog Devices, discusses products, applications, technology, and techniques for analog, digital, and mixed-signal processing. Published continuously for 44 years—starting in 1967—it is currently available in two versions. Monthly editions offer technical articles; timely information including recent application notes, new-product briefs, pre-release products, webinars and tutorials, and published articles; and potpourri, a universe of links to important and relevant information on the Analog Devices website, www.analog.com. Printable quarterly issues feature collections of monthly articles. For history buffs, the *Analog Dialogue* archive includes all regular editions, starting with Volume 1, Number 1 (1967), and three special anniversary issues. If you wish to subscribe, please go to www.analog.com/analogdialogue/subscribe.html. Your comments are always welcome; please send messages to dialogue.editor@analog.com or to: *Dan Sheingold*, Editor [dan.sheingold@analog.com] or *Scott Wayne*, Publisher and Managing Editor [scott.wayne@analog.com].

 ISSN 0161-3626 ©Analog Devices, Inc. 2010

Termination of High-Speed Converter Clock Distribution Devices

By Jerome Patoux

When using clock distribution devices[1] or fanout buffers to clock ADCs and DACs, two main sources of signal degradation—printed-circuit board (PCB) trace implementation and output termination—need to be dealt with.

Clock Traces and Signal Swing

PCB traces behave like low-pass filters, attenuating clock signals as they travel along the trace and increasing pulse-edge distortion with trace length. Higher frequency clock signals are subject to increased attenuation, distortion, and noise, but to improve jitter, which is worst at low slew rates (Figure 1), clock edges with a high slew rate are typically used. To correctly implement a quality clock, use high-swing clock signals and short clock PCB traces; place the device to be clocked as close to the clock-distribution device as possible.

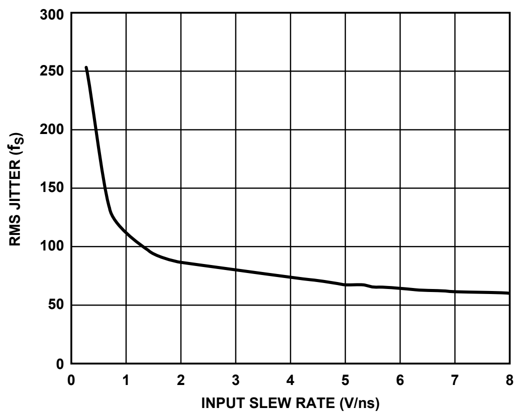


Figure 1. ADCLK925 rms jitter vs. input slew rate.

Two such clock-distribution devices are the ADCLK954[2] clock fanout buffer and the ADCLK914[3] ultrafast clock buffer. The ADCLK954 comprises 12 output drivers that can drive 800-mV full-swing ECL (emitter-coupled logic) or LVPECL (low-voltage positive ECL) signals into 50-Ω loads for a total differential output swing of 1.6 V, as shown in Figure 2. It operates at toggle rates to 4.8 GHz. The ADCLK914 can drive 1.9-V high-voltage differential signals (HVDS) into 50-Ω loads for a total differential output swing of 3.8 V. The ADCLK914 features a 7.5-GHz toggle rate.

When driving a DAC, the clock-distribution device should be placed as close as possible to the DAC's clock input so that the required high slew rate, high amplitude clock signals do not cause routing difficulties, generate EMI, or become degraded by dielectric and other losses. Note that the characteristic impedance (Z_0) of the trace will vary with trace dimension (length, width, and depth); the driver's output impedance must be matched to this characteristic impedance.

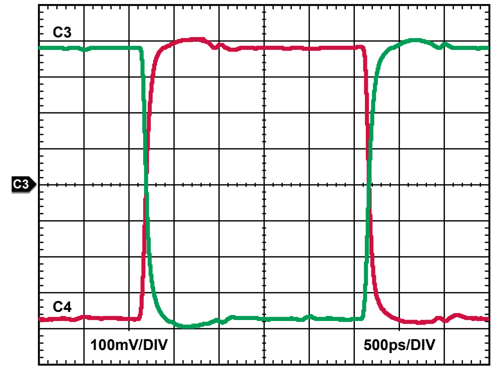


Figure 2. ADCLK954 clock buffer output waveforms with 3.3-V supply.

Output Termination

Clock-signal attenuation can cause increased jitter, so it is important to terminate the driver outputs to avoid signal reflection and to maximize power transfer over a relatively large bandwidth. Indeed, reflections may cause undershoot and overshoot, severely degrading the signal and the overall clock performance or, in extreme cases, possibly damaging the receiver or driver. Reflections, caused by impedance mismatches, occur when the traces are not properly terminated. They are more significant for high-speed signals with fast rise- and fall times due to the high-pass nature of the reflection coefficient. The reflected pulse is superimposed on the main clock signal, thus degrading the clock pulse. It also affects the edges of the clock signal by adding a time-delay uncertainty to the rising and falling edges, as shown in Figure 3.

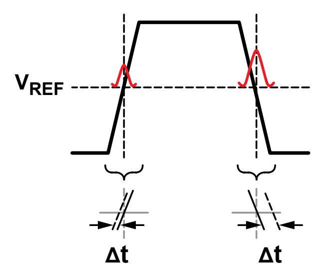


Figure 3. Jitter impact of reflected signal due to improper termination.

The magnitude of the echo due to the improper termination varies with time, so Δt will also vary with time. The termination time constant also affects the shape and width of the echo pulse. For these reasons, this additional reflection-induced jitter shape, which looks Gaussian, adds to the classical jitter. To avoid the adverse effects of this jitter and clock quality reduction, use proper signal termination, as summarized in Table 1. Z_0 is the impedance of the line; Z_{OUT} is the output impedance of the driver; and Z_{IN} is the input impedance of the receiver. Only CMOS and PECL/LVPECL circuits are shown.

References

[1] www.analog.com/en/clock-and-timing/clock-generation-and-distribution/products/index.html.

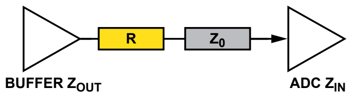
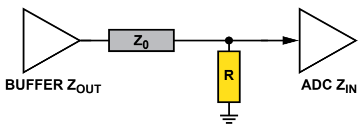
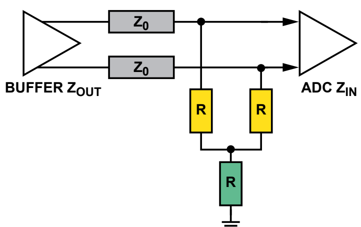
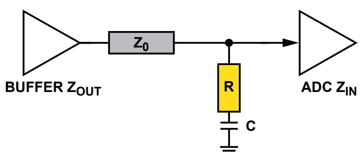
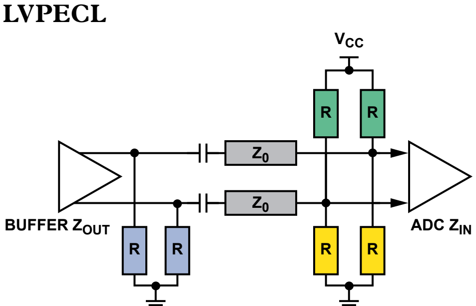
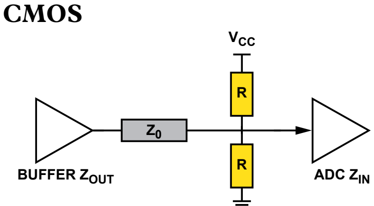
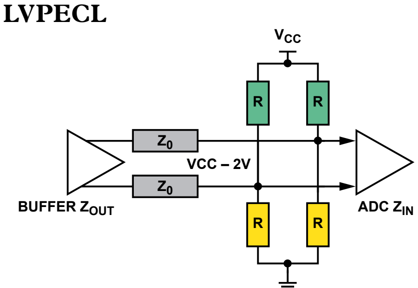
[2] www.analog.com/en/clock-and-timing/clock-generation-and-distribution/adclk954/products/product.html.

[3] www.analog.com/en/clock-and-timing/clock-generation-and-distribution/adclk914/products/product.html.

Author

Jerome Patoux [jerome.patoux@analog.com] is a product marketing engineer for ADI's Clock and Signal Synthesis Group in Greensboro, NC. In 2002, he graduated from ESIGETEL, Avon, France, with a master's degree in electronics and telecommunications engineering. He also has a master's degree in international project management from the University of Quebec in Hull-Gatineau, Canada, and ISMANS, Le Mans, France. Prior to joining ADI in 2005, Jerome worked as a radio engineer for SFR Group and as a department manager for SNCF.

Table 1. Clock Terminations

Method	Description	Strength	Weakness	Comments
Series Termination	**CMOS**  In practice, resistance (R) is omitted at the buffer output as it is hard to match the impedance due to its dynamic behavior over frequency.	Low power solution (no sink current to ground). Easy to calculate R ($Z_0 - Z_{OUT}$).	Rise/fall time impacted by circuit R and C, increasing jitter. Only useful with low-frequency signals.	CMOS drivers. Not suitable for high-frequency clock signals. Suitable for low-frequency clock signals and very short traces.
Pull-Down Resistor	**CMOS** 	Very simple ($R = Z_0$)	High power consumption.	Not recommended.
	LVPECL 	Simple, 3-resistor solution. Slightly better in terms of power saving, while saving a component compared to 4-resistor termination.		Recommended. Place termination resistors as close as possible to the PECL receiver.
AC Termination	**CMOS** 	No dc power consumption.		C should be small to avoid high power consumption, but not too small to allow sink current.
	LVPECL 	AC-coupling allows bias voltage adjustment. Avoids power flow between the two sides of the circuit.	AC-coupling is only recommended for balanced signals (50% duty cycle clock).	AC-coupling capacitors should be low ESR, low capacitance.
Resistor Bridge	**CMOS** 	Reasonable trade-off on power.	Uses two parts for single-ended clocks.	
	LVPECL 		Uses four external parts for differential output logic.	Widely used termination for 3.3-V LVPECL drivers.

ADIsimPower Provides Robust, Customizable DC-to-DC Converter Designs

By Matt Kessler

Introduction

Designers of dc-to-dc converters, both novice and expert alike, are faced with an overwhelming number of options for power management[1] ICs. Finding the best combination of features, performance, integration level, and price can be difficult enough—and the actual design work can be daunting. ADIsimPower™[2] is designed to both simplify the IC selection process and to provide the information required to build an optimized dc-to-dc converter.

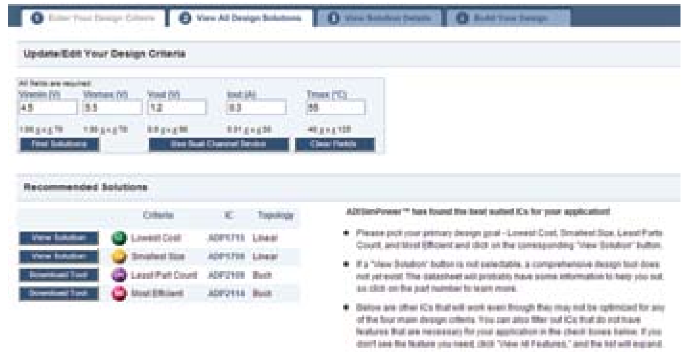
While most dc-to-dc selection guides simply direct users to switching regulators,[3] switching controllers,[4] and linear regulators[5] that will work with a given set of inputs—without providing the means to quantify the trade-offs made in selecting one part over another—ADIsimPower allows designers to investigate power-conversion trade-offs and navigate design complexities. An intelligent selection guide combined with a comprehensive design assistant, this new tool provides robust designs that are optimized to the user's exact inputs for size, efficiency, cost, parts count, or some combination thereof.

The ADIsimPower selection and design process comprises four steps: 1. **Enter Your Design Criteria**, 2. **View All Design Solutions**, 3. **View Solution Details**, and 4. **Build Your Design**. At the end of the experience, the goal is for ADIsimPower to provide a customized schematic, bill of materials (BOM) with vendor part numbers and prices, efficiency plots, performance specifications, closed-loop transfer function, and the means to rapidly build the design.

Enter Your Design Criteria

The first page of ADIsimPower includes user input fields for minimum input voltage, maximum input voltage, output voltage, output current, and maximum ambient temperature; bounds for each parameter are shown below the corresponding text box. After filling in the boxes, choose **Find Solutions** to find recommended solutions for this application.

Or, users who know which Analog Devices power management part they would like to use can select **choose the IC**. This will activate a pull-down menu that displays a list of currently supported power management ICs. After selecting one of these parts, the user will be taken directly to **View Solution Details**.



View All Design Solutions

The second stage of ADIsimPower helps the user to choose the best part for the design. At the top, the design inputs from the first stage are repeated for clarity. Below this, the **Recommended Solutions** suggests ICs and topologies for solutions yielding the lowest cost, smallest size, fewest parts, and highest efficiency. The recommendations are based on the entire dc-to-dc converter design, including the power management IC, inductors, capacitors, resistors, MOSFETs, and diodes.

Below the **Recommended Solutions**, a table quantifies to the first order the solution cost, size, efficiency, and component count for every IC that works, allowing users to see the design trade-offs without having to do individual designs with each part. Each column can be sorted to highlight the most important trade-offs. A feature list can be found at the right of the table. To expand or collapse this list, click on **Show all features** or **Show default features**. Clicking on the associated check box selects or deselects a feature. Select all the features that are required for the application. ICs that do not include the selected features will be removed from the table and from the recommended solutions.

After selecting the IC that best balances features, performance, integration level, and cost, click on the corresponding **View Solution** button. If this button is not active, comprehensive design support is not yet available on ADIsimPower. Click on the IC name to see the data sheet and other information. Click **Download Design Tool** to get an Excel-based design tool that can be run locally.

View Solution Details

In this stage, ADIsimPower generates and displays the complete design, including a customized schematic, a well-documented bill of materials, and estimates of operational parameters, power dissipation, and maximum temperature. Switching converter designs also show plots of efficiency, loss, and, in some cases, the closed-loop transfer function.

At the top of the page, the original inputs are repeated for clarity; they can be updated if the design parameters have changed. Click on **Modify Advanced Settings**, which will open up a window allowing the user to modify many settings, including, but not limited to, accuracy, maximum component height, peak-to-peak output voltage ripple, input filter requirements, load transient response, inductor ripple current, and MOSFET vendor preference (a subset of these features may be shown depending upon the part selection). The ability to modify these settings is one way in which ADIsimPower differentiates itself from other dc-to-dc converter design tools. Though understandable enough to make novice power supply designers feel comfortable, the advanced settings are exactly the kind of parameters expert power supply designers expect to control in their designs.

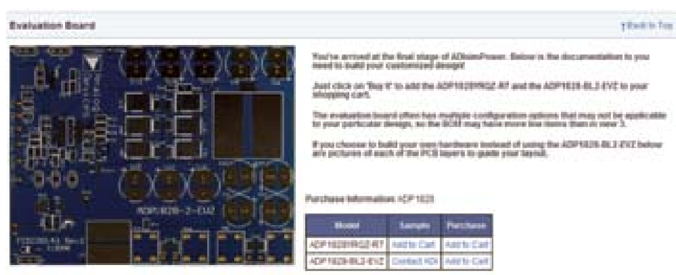
The next section on the page has several tabs that specify various important design parameters. Once again, the information in these tabs will vary depending upon the chosen IC, but common tabs include **Operational Estimates**, **Dissipation Estimates**, and **Temperature Estimates**. All parameters are shown at both minimum and maximum input voltage. The **Operational Estimates** tab includes parameters such as PWM duty cycle, peak-to-peak output voltage ripple, and peak inductor current. The **Dissipation Estimates** tab shows the power dissipated in each high loss component. The **Temperature Estimates** tab shows the temperatures of each of the components associated with loss in the **Dissipation Estimates** tab. Power dissipation and temperature calculations assume worst-case values for many of the parameters that dictate power loss to ensure a robust design.

The following section shows the complete customized schematic, including reference designations and pin numbers.

Next on the page is the **Bill of Materials**. This may have several components that can be edited, as indicated by an orange item number. Click on the item number to see a list of other components that are prequalified to work in the design. The column headings will vary depending upon the type of component. Typical headings include Manufacturer, Part Number, Loss (W), Area (mm^2), Hgt (mm), Cost ($), and other specifications that characterize the part and how it will work in the circuit. These allow the user to continue to make performance-vs.-size-vs.-cost trade-offs to fully customize the design. Each of these columns can be sorted, which makes quantifying gains and losses associated with changing parts easier. If a new part is selected, ADIsimPower will redesign with the selected component, ensuring that all specifications are still met.

Below the Bill of Materials is the **Graphs** section, which may include plots of efficiency, loss, and the closed-loop transfer function (Bode plot)—all at both minimum and maximum input voltages. The efficiency and loss curves correspond to losses associated with worst-case values for many high loss parameters. This worst-case analysis is common throughout the ADIsimPower design process. The goal is to give the user confidence that the designs provided by the tool are robust across component tolerances, ambient temperature range, and other circuit variances. This tool provides far more than a basic solution.

At the top of each section in the **View Solution Details** stage are radial buttons for each design criterion (**Lowest cost**, **Part count**, **Efficiency**, and **Size**). Selecting a new radial button will completely redesign the circuit according to the new design criteria, nullifying all BOM changes previously made. When the final design has been determined, click on **Build This Solution!**



Build Your Design

The first item on this page is a picture of the appropriate evaluation board for the IC chosen in the previous stages. Links to order the evaluation board and IC are to the right of the evaluation board picture. Below it is the schematic that corresponds to the entire evaluation board. Note that the schematic in this section of the tool corresponds to the evaluation board, which usually accommodates many different configurations. Next to each component on the schematic are value and package designators that will be helpful while building the board. Many of these will be designated **No Pop**, as they are not required in this specific design. Below the schematic is the bill of materials, which once again applies to the evaluation board. Each component is listed, allowing it to be checked off as the board is populated. The schematic and bill of materials in this section may be considerably longer and more complicated than the schematic and bill of materials in the **View Your Solution** section, which would be more representative of the final design. Below the bill of materials are pictures of the top assembly, bottom assembly, and all PCB layers of the evaluation board. In short, this stage provides everything one needs to build the design created in ADIsimPower.

Links allow the user to download or email all information found in the **Build Your Design** and **View Solution Details** sections in a format similar to that seen while interacting with the tool on the Web.

Parts Database

The parts database that ADIsimPower uses includes more than 3000 unique part numbers, including inductors, MOSFETs, diodes, capacitors, and power-management ICs. Each of these part types naturally has parasitic elements that cause them to behave in a nonideal fashion; they must be considered in order to do robust power supply design. Although many of these parasitic elements are not fully characterized on their data sheets, the architects and implementers of ADIsimPower have worked with manufacturers of these components to procure this unpublished information. The nonideal behaviors taken into account in the tool include, but are not limited to, the following:

Capacitors: change of capacitance with applied voltage (dC/dV), change of ESR with switching frequency (dC/dT).

Inductors: core loss and skin-effect losses as a function of switching frequency.

Diodes: change of forward voltage with forward current (dVf/dI), change of forward voltage with temperature (dVf/dT), change of parasitic capacitance with applied voltage (dC/dV).

MOSFETs: change of Rds(on) with temperature (dRds(on)/dT), change of Rds(on) with applied gate-to-source voltage (dRds(on)/dVgs), change of parasitic capacitance (Coss, Crss, Ciss) with applied voltage (dC/dV).

These are just the most general of the many nonideal component behaviors that designs produced by ADIsimPower take into account. The result is that both frequent and first-time users of ADIsimPower find the designs are robust and as close to production-ready as one could expect from any design tool.

Conclusion

ADIsimPower helps designers, both novice and expert, find the right IC for a dc-to-dc converter design by providing the means to find the best combination of features, performance, level of integration, and price point for their applications. The intelligent selection guide in the **View All Design Solutions** section allows the user to see trade-offs that could otherwise only be seen by doing the whole design for each part individually. The third section of the tool, **View Solution Details**, allows users to further hone the robust and well-documented design shown by editing components and adjusting advanced features. The final stage, **Build Your Solution**, provides all the necessary information to build the evaluation board to evaluate the design. ADIsimPower differentiates itself as a dc-to-dc voltage regulator design and selection tool by providing robust designs for switching controllers, switching regulators, and LDOs that are truly optimized for each unique application.

References

[1] www.analog.com/en/power-management/products/index.html.

[2] http://designtools.analog.com/dtPowerWeb/dtPowerMain.aspx.

[3] www.analog.com/en/power-management/switching-regulators-integrated-fet-switches/products/index.html.

[4] www.analog.com/en/power-management/switching-controllers-external-switches/products/index.html.

[5] www.analog.com/en/power-management/linear-regulators/products/index.html.

Author

Matt Kessler [matt.kessler@analog.com] is an applications engineer for Power Management Products in the Customer Applications Group in Fort Collins, CO. Responsible for technical support for a wide range of a products and customers, Matt is also one of the original architects and developers of ADIsimPower. Matt earned his BSEE from the University of Texas at Dallas and is currently pursuing his MSEE at Colorado State University. He has been with Analog Devices since 2007.

Automotive

Airbag/Crash Sensing, Rollover/Stability Control, Advanced Driver Assistance (ADAS Radar), Advanced Driver Assistance (ADAS Vision), Sensors and Sensor Interface, Battery Management, Hybrid Electric Vehicles (HEV)/Electric Vehicles (EV), Infotainment

http://automotive.analog.com/en/segment/am.html

Healthcare

Medical Imaging, Patient Monitoring, Medical Instrumentation, Consumer and Home Healthcare

http://healthcare.analog.com/en/segment/health.html

Process Control and Industrial Automation

Programmable Logic Controllers (PLC), Distributed Control Systems (DCS), Field Instruments/Smart Transmitters, Temperature Controllers

http://processcontrol.analog.com

Circuits from the Lab™

ADI engineers share their lab work with you in Circuits from the Lab™ circuit notes. You can combine these product pairings quickly and with confidence.

- Circuits provide solutions for many common application needs.
- Each circuit includes detailed design documentation, common circuit variations, and more.
- Circuit function and performance are verified in hardware.

http://www.analog.com/circuits

New Touch-Screen Controllers Offer Robust Sensing for Portable Displays

By Gareth Finn

Touch-screen displays that sense the occurrence and location of a physical touch on the display area are increasingly being used to replace mechanical buttons in a variety of devices, including smartphones, MP3 players, GPS navigation systems, digital cameras, laptop computers, video games, and laboratory instruments. First generation devices were not very accurate, suffered from false detection, and consumed too much power. Newer touch-screen controllers,[1] such as the AD7879,[2] offer improved accuracy, lower power consumption, and result filtering. They can also sense temperature, supply voltage, and touch pressure, facilitating robust sensing for modern touch-screen displays.

How Does a Touch Screen Work?

First, let's look at how a resistive touch screen operates. Figure 1 shows a basic diagram of the construction and operation of a touch screen.

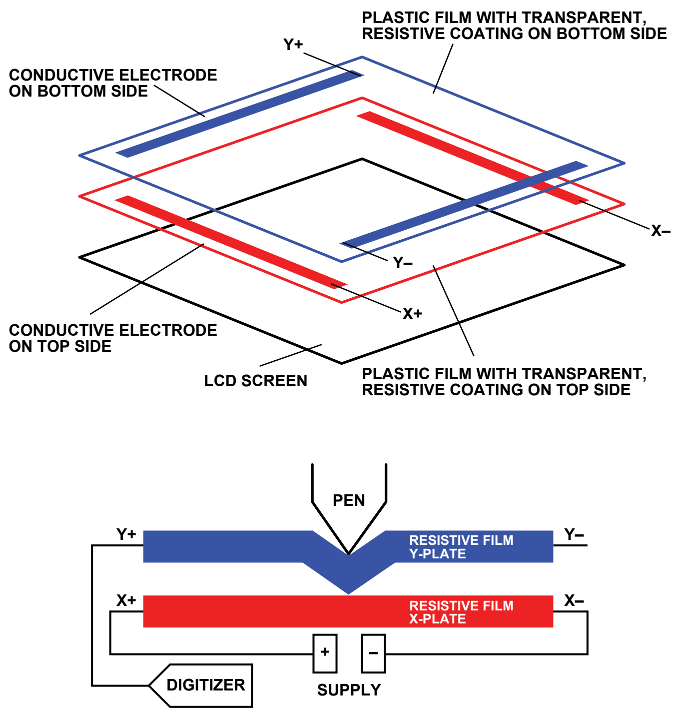


Figure 1. Construction of a resistive touch screen.

The screen is formed by two plastic films, each coated with a conductive layer of metal—usually indium tin oxide (ITO)—that are separated by an air gap. One plate, the X-plate in the diagram above, is excited by the supply voltage. When the screen is touched, the two conductive plates come together, creating a resistor divider along the X-plate. The voltage at the point of contact, which represents the position on the X-plate, is sensed through the Y+ electrode, as shown in Figure 2. The process is then repeated by exciting the Y-plate and sensing the Y position through the X+ electrode.

$$V_{Y+} = V_{REF} \times \frac{R_{X-}}{R_{X(TOTAL)}}$$

$$V_{X+} = V_{REF} \times \frac{R_{Y-}}{R_{Y(TOTAL)}}$$

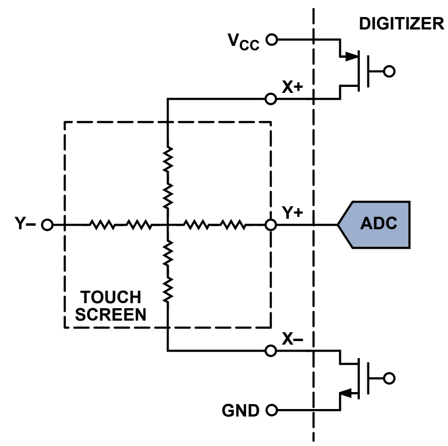


Figure 2. X-position measurement.

Next, the supply is placed across Y+ and X–, and two further screen measurements are made: Z1 is measured as the voltage at X+, and Z2 is measured as the voltage at Y–. These measurements can be used to estimate the touch pressure in one of two ways. If the resistance of the X-plate is known, the touch resistance is given by:

$$R_{TOUCH} = R_X \times \frac{X_{POS}}{2^N} \times \left(\frac{Z2}{Z1} - 1 \right)$$

If both X- and Y-plate resistances are known, the touch resistance is given by:

$$R_{TOUCH} = R_X \times \frac{X_{POS}}{2^N} \times \left(\frac{2^N}{Z1} - 1 \right) - R_Y \times \left(1 - \frac{Y_{POS}}{2^N} \right)$$

Larger values of touch resistance indicate lighter touch pressure.

AD7879 Touch-Screen Controller

The AD7879 touch-screen controller is designed to interface with 4-wire resistive touch screens. In addition to sensing touch, it also measures temperature and the voltage on an auxiliary input. All four touch measurements—along with temperature, battery, and auxiliary voltage measurements—can be programmed into its on-chip sequencer. Its wide supply voltage range (1.6 V to 3.6 V), small size (12-ball, 1.6 mm × 2 mm WLCSP; or 16-lead, 4 mm × 4 mm LFCSP), and low power dissipation (480 µA while converting, 0.5 µA in shutdown mode) make it flexible for use in a wide range of products.

Wake Up on Touch

The AD7879 can be configured to start up and convert when the screen is touched and to power down after release. This can be useful for battery-powered devices where power conservation is important. After each conversion sequence, the AD7879 delivers an interrupt to the host microcontroller, waking it from its low-power mode to process the data. Thus, the microcontroller also draws little power until the screen is touched. Figure 3 shows the setup for the wake-up-on-touch function.

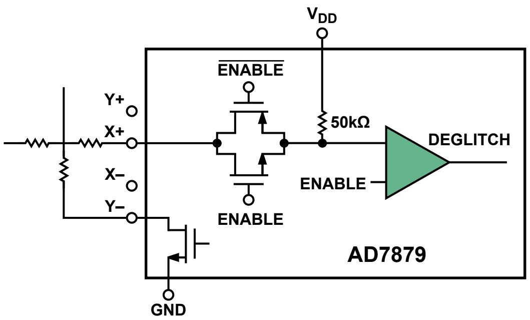


Figure 3. Wake-up-on-touch setup.

When the screen is touched, the X- and Y-plates connect, pulling the deglitch input low and waking the AD7879, which then starts converting. An interrupt is sent to the host at the end of the conversion.

Result Filtering

In a typical display, the resistive plates are placed on top of a liquid-crystal display (LCD), which contributes a lot of noise to the position measurement. This noise is a combination of impulse noise and Gaussian noise. The AD7879 offers median and averaging filters to reduce this noise. Instead of taking a single sample for position measurement, the sequencer can be programmed to take two, four, eight, or 16 samples. These samples are sorted, median filtered, and averaged to give a lower noise, more accurate result. The principle is illustrated more clearly in Figure 4. Sixteen position measurements are taken and are then ranked from lowest to highest. The four biggest and four smallest measurements are discarded to eliminate impulse noise; the remaining eight samples are averaged to reduce Gaussian noise. This has the added benefit of reducing the required amount of host processing and host-to-touch-screen controller communication.

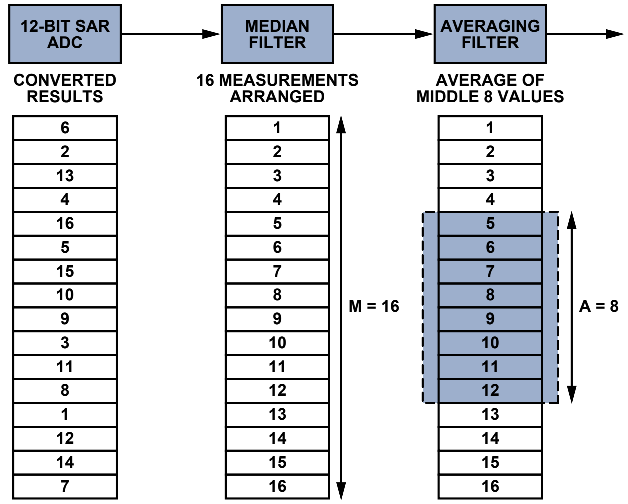


Figure 4. Median and average filtering.

References

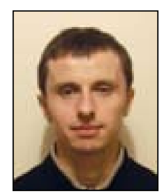
[1] www.analog.com/en/analog-to-digital-converters/touchscreen-controllers/products/index.html.

[2] www.analog.com/en/analog-to-digital-converters/touchscreen-controllers/ad7879/products/product.html.

Pratt, Susan. Ask The Applications Engineer—35, "Capacitance Sensors for Human Interfaces to Electronic Equipment." *Analog Dialogue*. Volume 40, Number 4. www.analog.com/library/analogdialogue/archives/40-10/cap_sensors.html.

Kearney, Paul. "The PDA Challenge—Met by the AD7873 Resistive-Touch-Screen Controller ADC." *Analog Dialogue*. Volume 35, Number 4. www.analog.com/library/analogdialogue/archives/35-04/touchscreen.

Author



Gareth Finn [gareth.finn@analog.com] currently works as a staff analog design engineer with the Integrated Portable Products Group in Limerick, Ireland. After graduating from University College Cork with a BE (Elec) in 1999, he spent five years as a designer with the Consumer Products Group at S3 in Cork, Ireland and two years as a designer in the Mixed-Signal Automotive Group at TI in Munich, Germany. Gareth joined the Transmit Signal Processing Group at ADI in October 2006.

Driving PIN Diodes: The Op-Amp Alternative

By John Ardizzoni

PIN diodes, which sandwich a lightly doped intrinsic (I) region between heavily doped P and N regions, are used extensively in RF and microwave applications. Common applications of PIN diodes are microwave switches, phase shifters, and attenuators, where high isolation and low loss are required. They can be found in test equipment, instrumentation, communications gear, radar, and a variety of military applications.

Every PIN diode in a switching circuit has an accompanying PIN diode driver or *switch driver* that provides a controlled forward bias current, a reverse bias voltage, and the activating interface between the control signal—typically a digital logic command—and one or more PIN diodes. This driver function can be rendered as a discrete design, or with specialized ICs to fit the application.

As an alternative, widely available op amps and specialty amplifiers, such as clamp amplifiers and differential amplifiers, can be used in place of discrete PIN-diode drive circuits and expensive PIN-diode driver ICs. Op amps in this class feature wide bandwidth, high slew rates, and more than enough steady-state current to drive PIN diodes. This article discusses three different PIN-driver circuits that employ op amps or specialty amplifiers—the AD8037, AD8137, and ADA4858-3. The circuits are designed to work with single-pole double-throw (SPDT) PIN-diode switches, but they can be adapted for other circuit configurations as well. They will be described in detail following a discussion on the nature and use of PIN diodes.

PIN Diodes

PIN diodes are used as current-controlled resistors at RF and microwave frequencies, with resistances that can range from a fraction of an ohm when forward biased, or *on*, to greater than 10 kΩ when reverse biased, or *off*. Differing from typical PN junction diodes, PIN diodes have an additional layer of highly resistive intrinsic semiconductor material (the *I* in PIN) sandwiched between the P and N material (Figure 1).

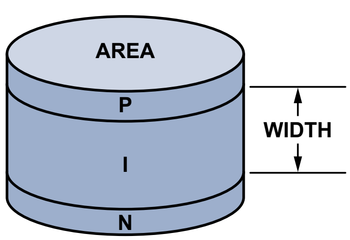


Figure 1. PIN diode.

When a PIN diode is forward biased, holes from the P material and electrons from the N material are injected into the I region. The charges cannot recombine instantaneously; the finite amount of time required for them to recombine is called the *carrier lifetime*. This causes a net stored charge in the I region, reducing its resistance to a value designated as R_S, the effective *on* resistance of the diode (Figure 2a).

When a reverse- or zero-bias voltage is applied, the diode appears as a large resistance, R_P, shunted by a capacitance, C_T (Figure 2b). By varying the diode geometry, it is possible to tailor PIN diodes to have a variety of combinations of R_S and C_T to meet the needs of various circuit applications and frequency ranges.

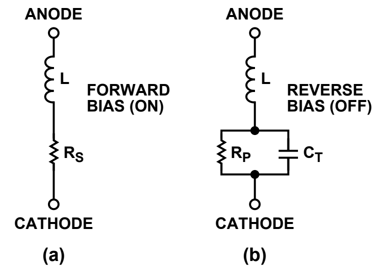


Figure 2. PIN diode equivalent circuits. a) *On*, $I_{BIAS} >> 0$. b) *Off*, $V_{BIAS} \leq 0$.

The combination of steady-state bias current, I_{SS}, and reverse voltage provided by the driver determines the final values of R_S and C_T. A set of relationships for members of a typical family of PIN diodes can be seen in Figure 3 and Figure 4—for M/A-COM MADP 042XX8-13060[1] series silicon diodes. The diode material affects its properties. For example, gallium-arsenide (GaAs) diodes require little—if any—reverse bias to achieve a low value of C_T, as shown in Figure 9.

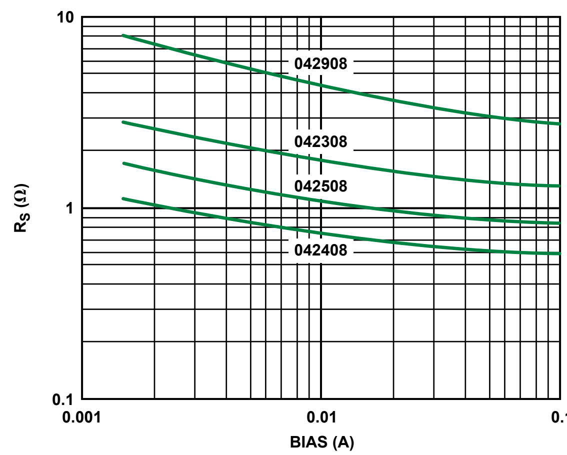


Figure 3. Silicon diode on resistance vs. forward current.

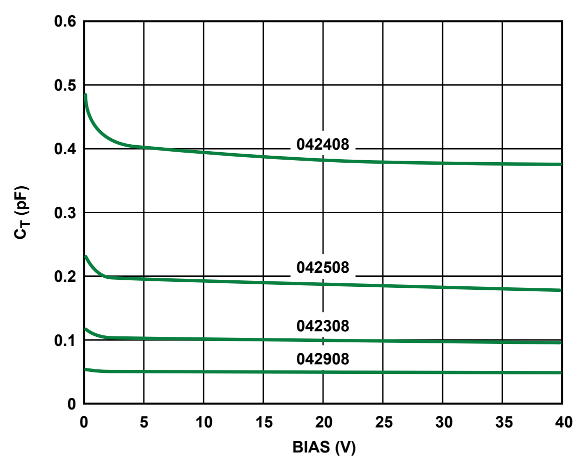


Figure 4. Silicon diode capacitance vs. reverse voltage.

The stored charge in a PIN diode can be approximated by Equation 1.

$$Q_S = \tau \times I_{SS} \qquad (1)$$

where:

Q_S = stored charge
τ = diode carrier lifetime
I_{SS} = steady-state current

To turn the diode on or off, the stored charge must be injected or removed. The driver's job is to inject or remove this stored charge very quickly. In cases where the switching time is less than the carrier lifetime of the diode, the peak current (I_P) required to effect fast switching can be approximated by Equation 2.

$$I_P = \frac{\tau \times I_{SS}}{t} \qquad (2)$$

where:

t = required switching time

I_{SS} = the steady-state current provided by the driver that sets the PIN-diode on resistance, R_S

τ = carrier lifetime

The driver injection or removal current, or *spiking current*, i, can be expressed by Equation 3.

$$i = C\frac{dv}{dt} \qquad (3)$$

where:

C = capacitance of the driver output capacitors, or *spiking caps*

v = voltage across the output capacitors

dv/dt = time rate of change of voltage across the capacitors

PIN-Diode Bias Interface

Connecting the switch driver control circuit to a PIN diode such that it can turn diodes on and off by applying a forward or reverse bias is a challenging task. The bias circuit typically uses a low-pass filter between the RF circuit and the switch driver. Figure 5 shows a single-pole double-throw (SPDT) RF switch and its bias circuit. When properly implemented, filters L1/C2 and L3/C4 allow control signals to be applied to PIN diodes D1–D4 with minimal interaction with the RF signal—which is switched from RF IN to PORT 1 or PORT 2. These elements allow the relatively lower frequency control signals to pass through to the PIN diodes but keep the high-frequency signal from escaping the RF signal path. Errant RF energy loss means undesirably higher insertion loss for the switch. Capacitors C1, C3, and C5 block the dc bias that is applied to the diodes from invading the circuitry in the RF signal path. Inductor L2, in the dc return path to ground, lets dc and low-frequency switch-driver signals pass with ease but presents a high impedance at RF and microwave frequencies, reducing RF signal loss.

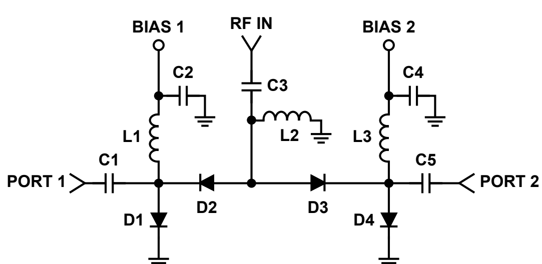


Figure 5. Typical single-pole, double-throw (SPDT) switch circuit.

Because the bias-, RF-, and switch-driver circuits all interact and affect each other's performance, trade-offs are essential, as in any design. For example, if C2 and C4 are large (>20 pF)—desirable for RF performance—the driver has a problem because large capacitances result in slower rising and falling edges. Fast switching is desired in most applications, so the capacitances must be kept to a minimum for optimum driver performance yet be high enough to meet the RF circuit requirements.

Traditional PIN-Diode Drivers

PIN-diode drivers come in a variety of shapes and sizes. Figure 6 is a schematic of a typical discrete switch driver that provides good switching speed. Such drivers can be realized with either *chip-and-wire* (hybrid) construction, which is very expensive, or with *surface-mount* (SMT) components, which are inexpensive but require more printed-circuit-board (PCB) area than a hybrid.

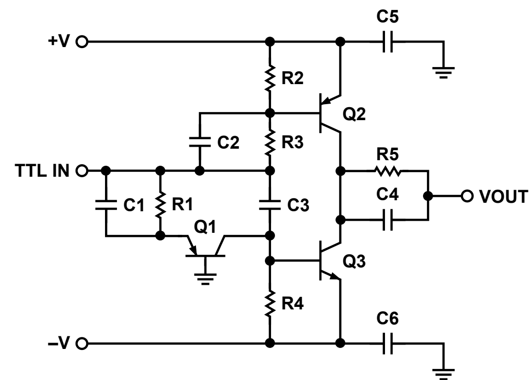


Figure 6. Discrete switch driver circuit.

Dedicated switch-driver *integrated circuits* (ICs) are also available; they are compact, provide a TTL interface, and have good performance, but their flexibility is limited, and they tend to be expensive.

Another kind of switch-driver architecture that should be considered employs *operational amplifiers*. A clear advantage of op-amp switch drivers is their inherent flexibility. They can be easily configured for a variety of applications, supply voltages, and conditions to provide the designer with a multitude of design options.

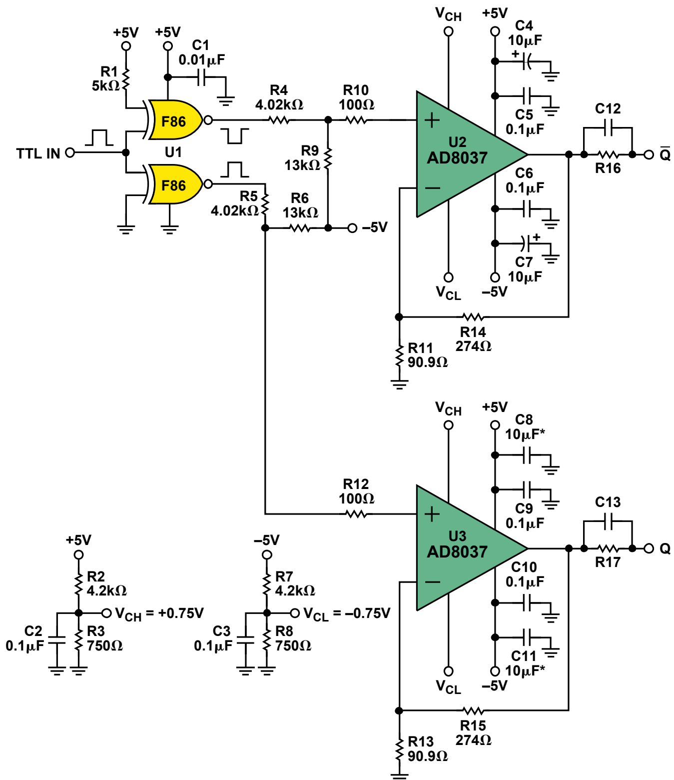
Op-Amp PIN-Diode Drivers

Op-amp circuits are an attractive alternative to traditional PIN diode drivers. Besides being flexible, they can operate with transition speeds often approaching or exceeding 1000 V/μs. Three different amplifier circuits for driving RF PIN diodes will be shown here. The amplifiers chosen are intrinsically different, yet they all perform a similar function. These amplifier circuits will drive silicon- or gallium-arsenide (GaAs) PIN diodes, but each has something different to offer.

AD8037—A Clamp Amplifier

This circuit can operate up to 10 MHz with excellent switching performance and a total propagation delay of 15 ns. The output voltage and current can be *tweaked* to fit different applications by varying either the gain or the clamp voltages.

The AD8037[2] clamp amplifier, originally designed to drive ADCs, provides a clamped output to protect against overdriving the ADC input. In the configuration of Figure 7, a pair of AD8037s (U2 and U3) are used to drive PIN diodes.



*C8 AND C11 MAY BE REDUCED OR ELIMINATED IF U2 AND U3 ARE LOCATED CLOSE TO EACH OTHER.

Figure 7. AD8037 PIN-diode driver circuit.

Here U2 and U3 are set for a noninverting gain of 4. The AD8037's unique input clamp feature allows extremely clean and accurate clamping. It amplifies the input signal linearly up to the point where the gain, multiplied by the positive and negative clamp voltages (V_{CH} and V_{CL}), is exceeded. With a gain of 4 and clamp voltages of ±0.75 V, the output voltage will be four times the input voltage for inputs smaller than ±0.75 V but will be clamped to a maximum of ±3 V when the input signal is larger than ±0.75 V. This clamping feature allows for very fast recovery (typically less than 2 ns) from overdrive. The clamp voltages (V_{CH} and V_{CL}) are derived by voltage dividers R2, R3, R7, and R8.

The digital interface is implemented by a 74F86 XOR logic gate (U1), which provides drive signals for U2 and U3 with minimal propagation-delay skew between the two complementary outputs. The network of resistors, R4, R5, R6, and R9, provides level shifting of the TTL outputs to approximately ±1.2 V, which is fed to U2 and U3 via R10 and R12.

The ±1.2-V inputs to U2 and U3 provide 60% overdrive, ensuring that the outputs will go into the clamped state (4 × 0.75 V). Thus, the output levels for the silicon PIN-diode driver are set to ±3 V. Resistors R16 and R17 limit the steady-state current. Capacitors C12 and C13 set the spiking current for the PIN diodes.

AD8137—A Differential Amplifier

Differential amplifiers, such as the AD8137 used in this example, provide exceptional high-speed switching performance at low cost and offer the designer great flexibility in driving various types of RF loads. A variety of differential amplifiers[3] are available, including faster and higher-performance devices.

The AD8137[3] high-speed differential amplifier, typically used for driving ADCs, can also serve as a low-cost, low-power PIN diode driver. Achieving typical switching times of 7 ns to 11 ns, including the propagation delays of the driver and the RF load, it features complementary outputs and is a versatile alternative to more expensive conventional drivers.

The circuit of Figure 8 converts a single-ended TTL input (0 V to 3.5 V) to a complementary ±3.5-V signal, while minimizing propagation delay. The TTL signal is amplified by a factor of 4 to produce the required ±3.5-V swing at the AD8137 outputs. The midpoint (or common-mode voltage) of the TTL signal is 1.75 V; the same value must be applied to R_2, as V_{REF}, to avoid introducing a common-mode offset error at the amplifier outputs. It is best to drive this point from a low source impedance; any series impedance will add to R_1 and affect the amplifier gain.

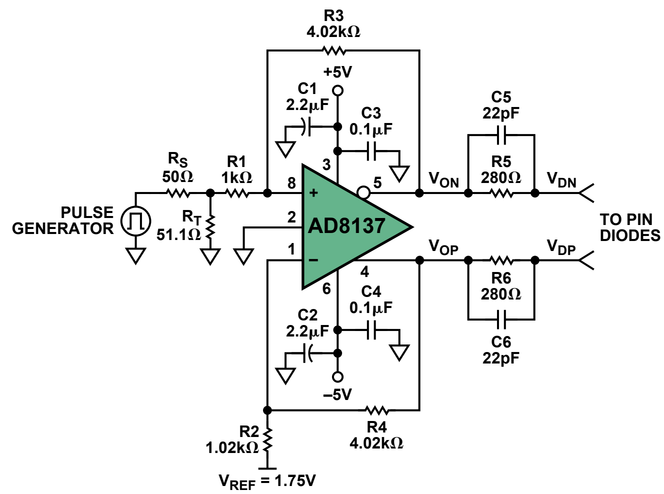


Figure 8. PIN-diode driver schematic.

The output voltage gain is established by Equation 4.

$$G = \frac{R_F}{R_G} = \frac{R_3}{R_1} = \frac{R_4}{R_2} \tag{4}$$

To properly terminate the pulse generator input impedance into 50 Ω, the input impedance of the differential amplifier circuit needs to be determined. This can be calculated using Equation 5, which gives R_T = 51.55 Ω, for which the closest standard 1% value is 51.1 Ω. For a symmetrical output swing it is important that the two input networks have the same impedance. This means that the inverting input impedance must incorporate the Thévenin impedance of the source and termination resistance into the gain-setting resistance, R_2. For a more detailed explanation see Application Note AN-1026.[5]

$$R_{IN} = \left(\frac{R_G}{1 - \frac{R_F}{2 \times (R_G + R_F)}} \right) \tag{5}$$

In Figure 8, R_2 is roughly 20 Ω larger than R_1 to compensate for the additional resistance (25 Ω) introduced by the parallel combination of the source resistor, R_S, and termination resistor, R_T. Setting R_4 to 1.02 kΩ, the closest standard value to 1.025 kΩ, ensures that the two resistor ratios are equal, to avoid introducing a common-mode error.

Output level shifting is easily accomplished using the AD8137's V_{OCM} pin, which sets the dc output common-mode level. In this case the V_{OCM} pin is tied to ground for a symmetrical output swing around ground.

Resistors R5 and R6 set the steady-state PIN-diode current as shown in Equation 6.

$$I = \frac{V_{ON} - V_{DN}}{R_5} \quad (6)$$

Capacitors C5 and C6 set the spiking current, which helps inject and remove the stored charge in the PIN diodes. Their capacitance values can be adjusted to optimize the performance required for a particular diode load. The spiking current can be determined by Equation 7.

$$i = C_{5\,\mathrm{or}\,6}\frac{dV}{dt} \quad (7)$$

ADA4858-3—A Triple Op Amp with Charge Pump

Many applications make available just a single supply. This can often be problematic for the circuit designer, especially when looking for low off capacitance in PIN circuits. In such cases an op amp that has an on-board charge pump is useful in a circuit to drive silicon or GaAs PIN diodes without requiring an external negative supply. This can provide significant savings by conserving space, power, and budget.

One such device is the ADA4858-3,[6] a high-speed current-feedback triple amplifier with the distinguishing feature that it includes an on-board charge pump that enables the output to swing –3 V to –1.8 V below ground, depending on the supply voltage and loading. It is robust enough to actually power other circuitry with up to 50 mA of negative supply current.

The ADA4858-3 provides a unique solution to the problem of driving a complementary PIN-diode microwave switch in a single-supply system. Recalling Figure 4, it can be seen that, depending on the PIN diode type, even a small amount of reverse bias helps lower the diode capacitance, C_T. GaAs PIN diodes benefit from this type of driver, as they typically don't require as much negative bias to keep their off capacitance (C_T) low (Figure 9).

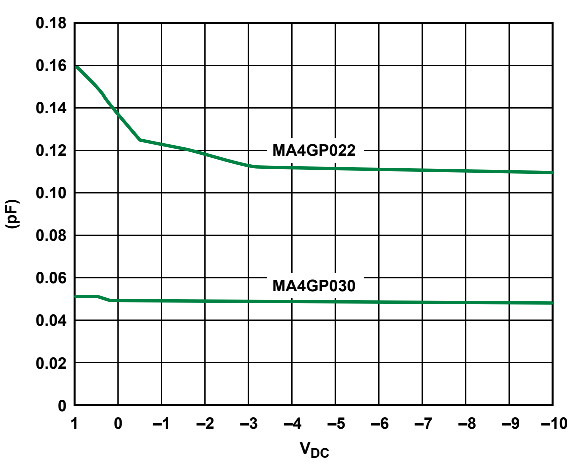


Figure 9. GaAs CT capacitance vs. voltage.

Figure 10 shows a circuit using the ADA4858-3 as a PIN-diode driver. A buffer gate can be added to the input to make the circuit compatible with TTL or other logic. For this circuit, the requirement is to convert a TTL 0-V to 3.5-V input signal swing to a complementary –1.5-V to +3.5-V swing for driving PIN diodes.

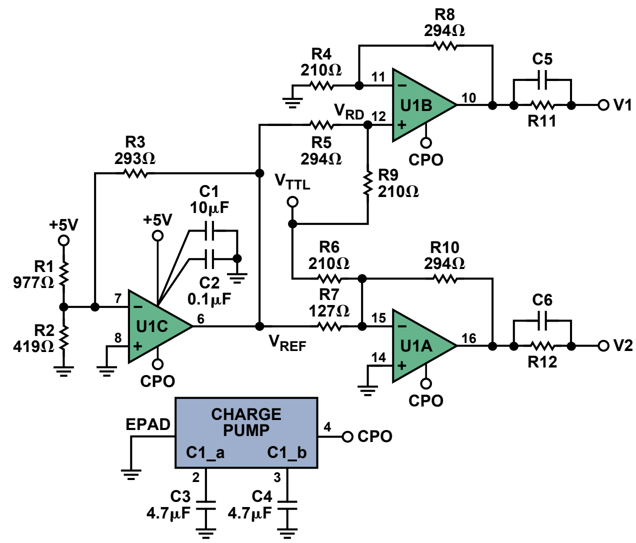


Figure 10. ADA4858-3 as a PIN-diode driver.

R1, R2, R3, and U1C form a –1.5-V reference for the circuit with the internal negative voltage, CPO, generated by the on-chip charge pump. Capacitors C3 and C4 are required for charge pump operation. The negative reference is then summed passively with the V_{TTL} input via voltage divider (R5 and R9). The resulting voltage (V_{RD}) appears at the noninverting input of U1B. The U1B output voltage can be calculated using Equation 8.

$$V_1(No\ Load) = \left[1 + \frac{R_8}{R_4}\right] V_{RD} = (2.4)\,V_{RD} \quad (8)$$

where:

$$V_{RD} = \left(\frac{R5}{R5 + R9}\right) V_{TTL} + \left(\frac{R9}{R9 + R5}\right) V_{REF} = \frac{294}{504} V_{TTL} + \frac{210}{504} V_{REF} \quad (9)$$

The negative reference is also fed to amplifier U1A where it is summed with the TTL input; the resulting output voltage, V2, can be calculated using Equation 10.

$$V2(No\ Load) = -1.4V_{TTL} - 2.3V_{REF} \quad (10)$$

Since these amplifiers employ a current-feedback architecture, attention must be paid to the choice of feedback resistance, which plays a major role in the stability and frequency response of the amplifier. For this application, the feedback resistor is set at 294 Ω, as recommended in the data sheet. Output voltages V1 and V2 can be described by Equation 8 and Equation 10, respectively. The amount of output spiking current can be determined using Equation 3 for the voltage across capacitors C5 and C6. The steady-state current, which sets the PIN diode on resistance, is established by voltage differences across R11 and R12 and depends on the PIN diode curves and system requirements.

For this application, the RF switch load was a MASW210B-1 silicon PIN diode SPDT switch, used in the front end of a microwave downconverter (Figure 11).

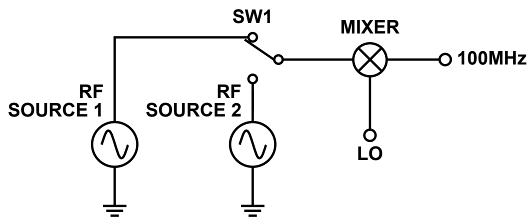


Figure 11. Downconverter block diagram.

The switch output waveform and the TTL input signal are shown in Figure 12. Note the fast rising and falling edges. This application did not use spiking caps, C5 and C6, due to the relatively slow switching time requirement of the switch, approximately 50 ns. R11 and R12, which establish the steady-state diode current, were 330 Ω resistors.

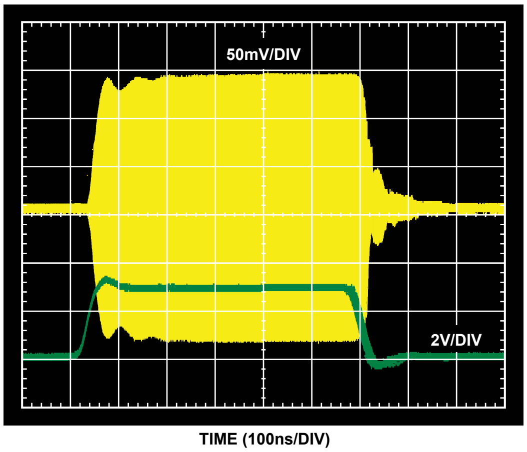


Figure 12. Waveform showing RF switching speed.

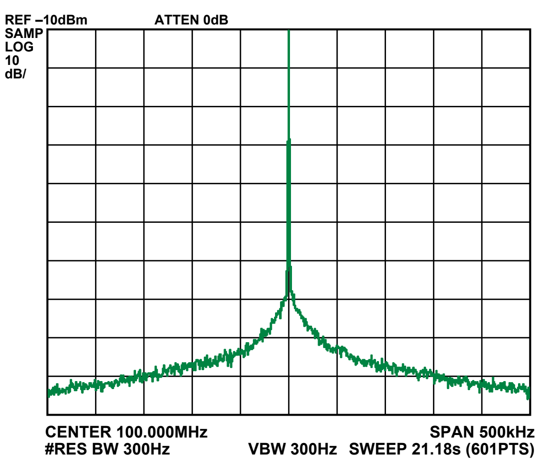


Figure 13. Spectral response of downconverter.

Figure 13 shows the spectral frequency response of the downconverter front end with switch SW1 in a fixed position to check insertion loss. Note the absence of harmonics or sidebands—a good indication that there are no perceptible 100 kHz switching artifacts emanating from the ADA4858-3's on-chip charge pump—an important consideration when using these devices in this type of application.

Conclusion

As these three examples show, op amps can provide creative alternatives to traditional drivers, with performance rivaling that of dedicated ICs designed solely for driving PIN diodes. Furthermore, op amps afford the ability to tailor gains, manipulate inputs, and—when using devices containing an internal charge pump—eliminate a negative supply, adding a dimension of flexibility to the design of drivers for PIN diodes and other circuitry. Easy to use and configure, op amps solve complex problems with relative ease.

Acknowledgment

The switching-speed- and spectral data, RF load, and test equipment were provided courtesy of Sage Laboratories, Hudson, NH. The tests were facilitated by Tony Cappello, CTO, with technical assistance provided by David Duncan, VP of Engineering.

References

Hiller, Gerald. *Design with PIN Diodes.* M/A-COM Application Note AG312.

Understanding RF/Microwave Solid State Switches and Their Applications. Agilent Application Note.

[1] www.macomtech.com/datasheets/MADP-042008_Series.pdf.

[2] www.analog.com/en/amplifiers-and-comparators/operational-amplifiers-op-amps/ad8037/products/product.html.

[3] www.analog.com/diffamp.

[4] www.analog.com/en/amplifiers-and-comparators/operational-amplifiers-op-amps/ad8137/products/product.html.

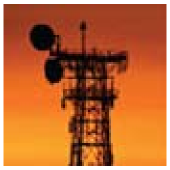
[5] www.analog.com/static/imported-files/application_notes/AN-1026.pdf.

[6] www.analog.com/en/audiovideo-products/video-ampsbuffersfilters/ada4858-3/products/product.html.

Author

John Ardizzoni [john.ardizzoni@analog.com] is a senior applications engineer in ADI's High-Speed Amplifier Group. John joined Analog Devices in 2002. Prior to that, he worked for IBM as an applications engineer and for M/A-COM as a design engineer. John graduated from Merrimack College in 1988 and has over 29 years of experience in the electronics industry.

Communications



Wireless Infrastructure

Networking

Optical

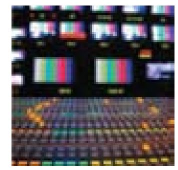
http://communications.analog.com

Consumer

Digital Imaging

Home Entertainment

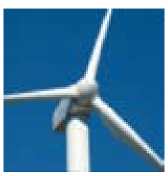
Portable



ProAV

http://consumer.analog.com

Energy



Renewable Energy Generation

Transmission/Distribution

Metering

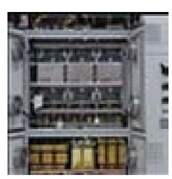
http://energy.analog.com

Motor and Power Control

Low-End Motor Control

High-End Motor Control

Servos and Robotics



Standard Inverters

UPS and Advanced Power Supplies

Wind Turbines and PV Cells

http://motorcontrol.analog.com

LEAD THE WAY ADVISOR™ BETA

Lead the Way Advisor™ will assist you in accessing the most relevant content and resources in a quick, process-driven fashion based on what you are trying to accomplish today. We invite you to try this tool and provide feedback as we evolve this service.

http://www.analog.com/advisor

Ask The Applications Engineer—39

Zero-Drift Operational Amplifiers

By Reza Moghimi

What Are Zero-Drift Amplifiers?

Zero-drift amplifiers dynamically correct their offset voltage and reshape their noise density. Two commonly used types—auto-zero amplifiers and choppers—achieve nanovolt-level offsets and extremely low offset drifts due to time and temperature. The amplifier's 1/f noise is also seen as a dc error, so it is removed as well. Zero-drift amplifiers provide many benefits to designers, as temperature drift and 1/f noise, always nuisances in the system, are otherwise very difficult to eliminate. In addition, zero-drift amplifiers have higher open-loop gain, power-supply rejection, and common-mode rejection as compared to standard amplifiers; and their overall output error is less than that obtained by a standard precision amplifier in the same configuration.

What Are Good Applications for Zero-Drift Amplifiers?

Zero-drift amplifiers are used in systems with an expected design life of greater than 10 years and in signal chains that use high closed-loop gains (>100) with low-frequency (<100 Hz), low-amplitude level signals. Examples can be found in precision weigh scales, medical instrumentation, precision metrology equipment, and infrared-, bridge-, and thermopile sensor interfaces.

How Does Auto-Zeroing Work?

Auto-zero amplifiers, such as the AD8538, AD8638, AD8551, and AD8571 families, usually correct for input offset in two clock phases. During Clock Phase A, switches labeled φA are closed, while switches labeled φB are open, as shown in Figure 1. The offset voltage of the nulling amplifier is measured and stored on capacitor C_{M1}.

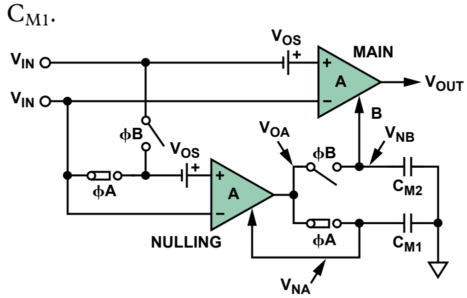


Figure 1. Phase A of auto-zero amplifier: nulling phase.

During Clock Phase B, switches labeled φB are closed, while switches labeled φA are open, as shown in Figure 2. The offset voltage of the main amplifier is measured and stored on capacitor C_{M2}, while the stored voltage on capacitor C_{M1} adjusts for the offset of the nulling amplifier. The overall offset is then applied to the main amplifier while processing the input signal.

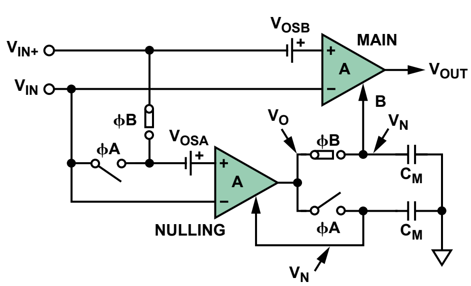


Figure 2. Phase B of auto-zero amplifier: auto-zero phase.

The sample-and-hold function turns auto-zero amplifiers into sampled-data systems, making them prone to aliasing and fold-back effects. At low frequencies, noise changes slowly, so the subtraction of two consecutive noise samples results in true cancellation. At higher frequencies this correlation diminishes, with subtraction errors causing wideband components to fold back into the baseband. Thus, auto-zero amplifiers have more in-band noise than standard op amps. To reduce low-frequency noise, the sampling frequency has to be increased, but this introduces additional charge injection. The signal path includes only the main amplifier, so relatively large unity-gain bandwidth can be obtained.

How Does a Chopper Work?

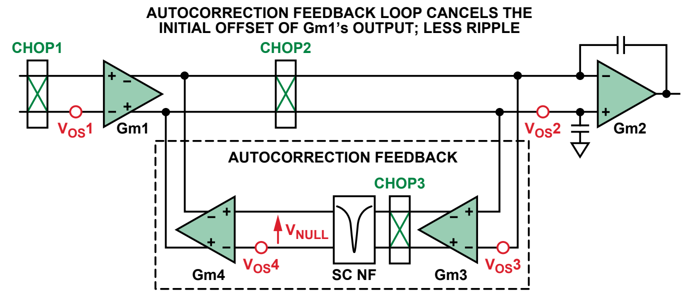


Figure 3. Chopping scheme used in the ADA4051.

Figure 3 shows the block diagram design of the ADA4051 chopper amplifier, which uses a local autocorrection feedback (ACFB) loop. The main signal path includes input chopping network CHOP1, transconductance amplifier Gm1, output chopping network CHOP2, and transconductance amplifier Gm2. CHOP1 and CHOP2 modulate the initial offset and 1/f noise from Gm1 up to the chopping frequency. Transconductance amplifier Gm3 senses the modulated ripple at the output of CHOP2. Chopping network CHOP3 demodulates the ripple back to dc. All three chopping networks switch at 40 kHz. Finally, transconductance amplifier Gm4 nulls the dc component at the output of Gm1—which would otherwise appear as ripple in the overall output. The switched capacitor notch filter (SCNF) selectively suppresses the undesired offset-related ripple without disturbing the desired input signal from the overall output. It is synchronized with the chopping clock to perfectly filter out the modulated components.

Can the Two Techniques Be Combined?

This is exactly what is done in a new series of amplifiers from Analog Devices. The AD8628 zero-drift amplifier, shown in Figure 4, uses both auto-zeroing and chopping to reduce the energy at the chopping frequency, while keeping the noise very low at lower frequencies. This combined technique allows wider bandwidth than was possible with conventional zero-drift amplifiers.

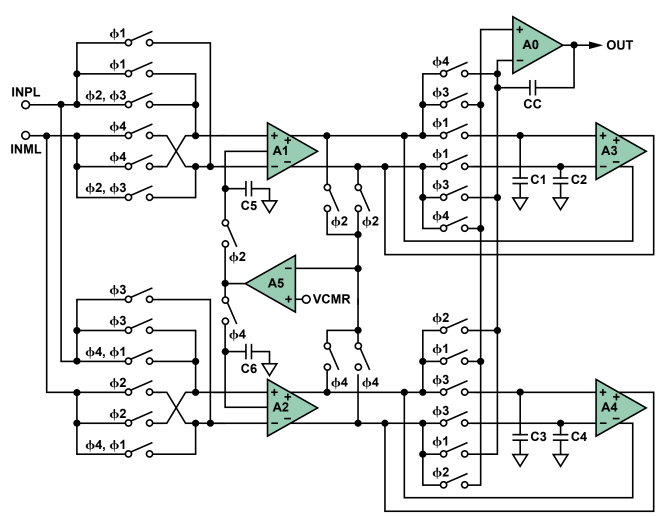


Figure 4. The AD8628 combines auto-zeroing with chopping to achieve wider bandwidth.

What Applications Issues Are Encountered When Using Zero-Drift Amplifiers?

Zero-drift amplifiers are composite amplifiers that use digital circuitry to dynamically correct for analog offset errors. The charge injection, clock feedthrough, intermodulation distortion, and increased overload recovery time that result from the digital switching action can cause problems within poorly designed analog circuits. The magnitude of the clock feedthrough increases with an increase in closed-loop gain or source resistance; adding a filter at the output or using a lower resistance on the noninverting input will reduce the effect. Also, the output ripple of a zero-drift amplifier increases as the input frequency gets closer to the chopping frequency.

What Happens to Signals at Frequencies Higher than That of the Internal Clock?

Signals with frequencies greater than the auto-zero frequency can be amplified. The speed of an auto-zeroed amplifier depends on the gain-bandwidth product, which is dependent on the main amplifier, not the nulling amplifier; the auto-zero frequency gives an indication of when switching artifacts will start to occur.

What Are Some Differences Between Auto-Zeroing and Chopping?

Auto-zeroing uses sampling to correct offset, while chopping uses modulation and demodulation. Sampling causes noise to fold back into baseband, so auto-zero amplifiers have more in-band noise. To suppress noise, more current is used, so the devices typically dissipate more power. Choppers have low-frequency noise consistent with their flat-band noise but produce a large amount of energy at the chopping frequency and its harmonics. Output filtering may be required, so these amplifiers are most suitable in low-frequency applications. Typical noise characteristics of auto-zero and chopping techniques are shown in Figure 5.

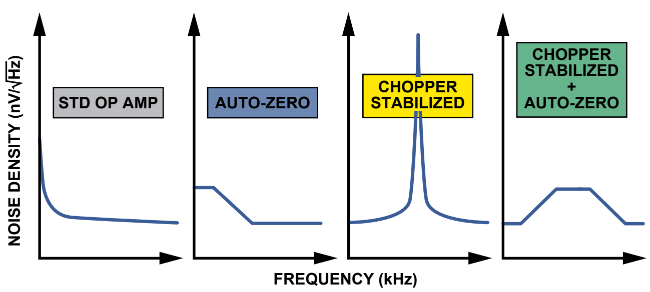


Figure 5. Typical noise of various amplifier topologies vs. frequency.

When Should I Use Auto-Zero Amplifiers? When Should I Use Choppers?

Choppers are a good choice for low-power, low-frequency applications (<100 Hz), while auto-zero amplifiers are better for wideband applications. The AD8628, which combines auto-zero and chopping techniques, is ideal for applications that require low noise, no switching glitch, and wide bandwidth. Table 1 shows some of the design trade-offs.

Table 1.

Auto-Zero	Chopper Stabilized	Chopper Stabilized + Auto-Zero
Very low offset, TCV_{OS}	Very low offset, TCV_{OS}	Very low offset, TCV_{OS}
Sample-and-hold	Modulation/demodulation	Sample-and-hold, modulation/demodulation
Higher low-frequency noise due to aliasing	Similar noise to flat band (no aliasing)	Combined noise shaped over frequency
Higher power consumption	Lower power consumption	Higher power consumption
Wide bandwidth	Narrow bandwidth	Widest bandwidth
Lowest ripple	Higher ripple	Lower ripple level than chopping
Little energy at auto-zero frequency	Lots of energy at chopping frequency	Little energy at auto-zero frequency

What Are Some of ADI's Popular Zero-Drift Amplifiers?

Table 2 shows a sample of zero-drift amplifiers offered by ADI.

References

1. "Bridge-Type Sensor Measurements Are Enhanced by Auto-Zeroed Instrumentation Amplifiers." www.analog.com/library/analogdialogue/cd/vol38n2.pdf#page=6.
2. "Demystifying Auto-Zero Amplifiers—Part 1." www.analog.com/library/analogdialogue/cd/vol34n1.pdf#page=27.
3. "Demystifying Auto-Zero Amplifiers—Part 2." www.analog.com/library/analogdialogue/cd/vol34n1.pdf#page=30.
4. MT-055 Tutorial, *Chopper Stabilized (Auto-Zero) Precision Op Amps*. www.analog.com/static/imported-files/tutorials/MT-055.pdf.

Author

Reza Moghimi [reza.moghimi@analog.com] is an applications engineer in San Jose, CA. He received a BSEE from San Jose State University in 1984 and an MBA from National University in 1990—and has also received a number of on-the-job certificates. He has worked for Raytheon Corporation, Siliconix, Inc., and Precision Monolithics, Inc. (PMI)—which was integrated with Analog Devices in 1990. At ADI, he has served in test-, product-, and project-engineering assignments. He has written many articles and design ideas—and has given presentations at technical seminars. His hobbies include travel, music, and soccer.

Table 2.

Part Number			Supply Voltage		Rail-to-Rail		BW @ A_{CL} Min	Slew Rate	V_{OS} Max	TCV_{OS} Typ	CMRR Min	PSRR Min	A_{VOL} Min	Noise @ 1 kHz	I_S/Amp Max	
Single	Dual	Quad	Min	Max	In	Out	(MHz)	(V/µs)	(µV)	(µV/°C)	(dB)	(dB)	(dB)	(nV/√Hz)	(mA)	Topology
AD8628	AD8629	AD8630	2.7	5.5	•	•	2.5	1	5	0.002	120	115	125	22	1.1	AZ, C
AD8538	AD8539		2.7	5.5	•	•	0.43	0.4	13	0.03	115	105	115	50	0.18	AZ
AD8638	AD8639		4.5	16		•	1.35	2.5	9	0.01	118	127	120	60	1.3	AZ
AD8551	AD8552	AD8554	2.7	5.5	•	•	1.5	0.4	5	0.005	120	120	125	42	0.975	AZ
AD8571	AD8572	AD8574	2.7	5.5	•	•	1.5	0.4	5	0.005	120	120	125	51	0.975	AZ
ADA4051-1	ADA4051-2		1.8	5.5	•	•	0.115	0.04	15	0.02	105	110	106	95	0.017	C

Free and Open-Source Software—An Analog Devices Perspective

By Michael Hennerich and Robin Getz

As a System Designer, Why Should I Care About *Free and Open-Source Software*?

The rapid increase in use of free and open-source software (FOSS) represents the most significant, all-encompassing, and long-term trend that the embedded industry has seen since the early 1980s.[1] FOSS software licenses make source code available and grant developers the right to study, change, and improve the software design.[2] FOSS is already playing, or will eventually play, a role in the life cycle of every major software category, influencing everything from 64-bit servers to 8-bit microcontrollers. FOSS will fundamentally change the value proposition of software for all users and developers.

So, for most embedded developers, if FOSS is not in your design today, most likely it will be soon.

What Is FOSS?

The main difference between *free software* and *open-source software* is in the philosophy of their inherent freedoms. A "free software" license is one that respects the end user's four essential freedoms:

1. The freedom to run it.
2. The freedom to study and change it.
3. The freedom to redistribute copies.
4. The freedom to improve the program and release those improvements.

Being free to do these things means (among other things) that you do not have to ask (or pay) for permission to do so. This is a matter of freedom, not commerce, so think of "free speech, not free beer."[3] Also note that the freedoms are for the *end user*, *not* the developer, *nor* the person who distributes the software.

On the other hand, *open-source software* does not always provide the end user the same freedoms, but it does provide *developers* such rights as access to the source.[4] Various open-source licenses allow developers to create proprietary, closed-source software, which includes no requirement to distribute the source code for the end work. The BSD (Berkeley Software Distribution) license is an example of those that allow binary redistributions without source code.[5]

The key real-world difference between FOSS and closed-source, or proprietary, software is the mass collaborative nature of development—the large number of people working independently on individual projects; here any user can become a developer—reporting and fixing bugs or adding new features.

The popularity of FOSS in the embedded markets is dominated by simple economic motivation[6]—it lowers software costs and hastens time to market. It turns "roll-your-own" developers into system-level integrators who can focus on adding value and differentiating features of their products rather than reproducing the same base infrastructure over and over again. It is the only proven methodology to reel in out-of-control software development costs. Irrespective of the organization, one can almost always find one of the five stages of open-source adoption taking place (our apologies to the late Dr. Kübler-Ross).[7]

Five Stages of FOSS Adoption[8]

State	Symptom of Progression
Denial: that FOSS is already in use	• No recent audits of custom software • Low awareness of popular FOSS components • No official company policy for FOSS usage
Anger: over surprise loss of control	• Software in use with no record of adoption • Management looks to assign accountability • Developers practice "don't ask, don't tell"
Bargaining: to re-establish existing controls and processes	• Crash program to identify total exposure • Program put in place to remove existing FOSS • Lawyers spend hours meeting with development teams
Depression: on realizing the point of no return has been reached	• Realization that extracting open source would bring development to a halt • Recognition that the effort involved in removing FOSS would be prohibitive
Acceptance: can't fight it, might as well prepare for it	• Implementation of a formal FOSS strategy • Adjustments to policies and procedures • An attitude to shift from tolerance to extracting value

While many people equate FOSS with the prominent Linux® kernel, or a Linux-based distribution, the use of FOSS beyond Linux in embedded development is pervasive; it is used by almost three-quarters of organizations and spans hundreds of thousands of projects. However, with the increasing popularity of embedded Linux-based systems, the interest in finding Linux drivers for embedded peripherals (ADCs, DACs, audio codecs, accelerometers, touch screen controllers, etc.) becomes increasingly compelling.

We will discuss here the contributions of Analog Devices, Inc. (ADI), to various FOSS projects, focusing on the Linux kernel and how they are being used by our customers to reduce risk and decrease product development time. We will take a look at a few popular devices, for example, the ADXL345 digital 3-axis accelerometer, and describe:

- layers of the driver created, modified, and maintained by ADI
- where things are maintained (location of driver download)
- interface code (common code for the kernel)—allowing you to use the driver on your platform
- common practice for driver development (which files can be changed or contributed, and which cannot)
- where the code can be found—how to log bug- and problem reports

Linux Device Drivers—Architecture Independence

The majority of Linux users are (happily) unaware of the underlying hardware complexity and issues involved in the Linux kernel, and are surprised to find out how much of the kernel is independent from the hardware on which it runs. In fact, the vast majority of source code in the Linux kernel is related to architecture-independent device drivers: Of the entire 7,934,566[9] lines in the Linux 2.6.32.6 kernel, an overwhelming 4,758,810 lines—over 60%—is in `./drivers`, `./sound`, and `./firmware`. Architecture-dependent code is a very small fraction of the Linux kernel—1,501,545 lines (18.9%) for *all 22* different architectures. The top 10 architectures that the kernel supports:

Architecture Directory	Lines of Source	Fraction of the Kernel
./arm	302,125	3.81%
./powerpc	188,825	2.38%
./x86	154,379	1.95%
./mips	139,782	1.76%
./m68k	106,392	1.34%
./sparc	88,529	1.12%
./ia64	85,103	1.07%
./sh	77,327	0.97%
./blackfin	74,921	0.94%
./cris	72,432	0.91%

This makes clear that architecture-*in*dependent drivers (~60% of the kernel source) play a very important role.

For each piece of Linux-supported hardware, someone has written a device driver. Since 2007, Analog Devices has ranked within the top 20 companies (from over 300+) contributing code to the Linux kernel[10]—and has a full-time team working on Linux device drivers.

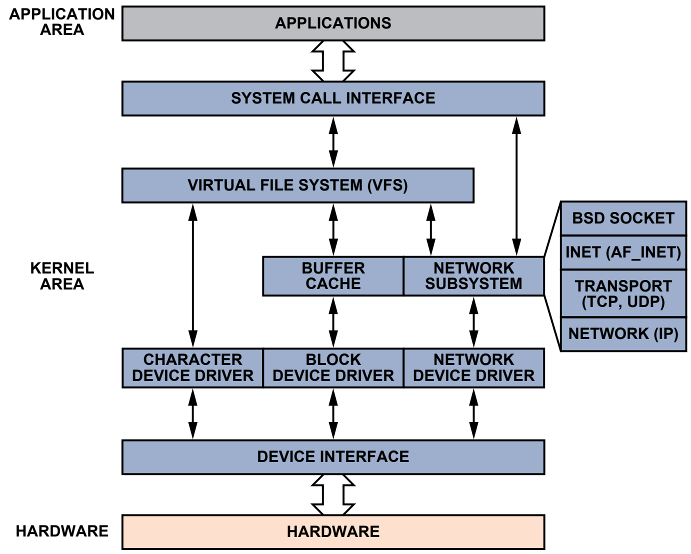
Basics of Linux Device Drivers

A device driver simplifies programming by acting as a translator between the hardware and the application (user code), or the kernel that uses it, hiding the details of how the hardware works. Programmers can write the higher-level application code using a set of standardized calls (system calls)—independent of the specific hardware device it will control or the processor it will run on. Using a well-defined internal *application programming interface* (kernel API), the application code and device driver can interface in a standard way regardless of the software superstructure or underlying hardware.

Operating systems (OS) handle the details of hardware operation for a specific processor platform. Kernel (OS) internal *hardware abstraction layers* (HALs) and processor-specific peripheral drivers (such as I^2C® or SPI bus drivers) allow even a typical device driver to be processor platform independent. This approach allows a device driver—for the AD7879 *touch-screen digitizer*, for example—to be used without modification on any processor platform running Linux, with any *graphical user interface* (GUI) package and suitable application running on top of the Linux kernel. If the hardware designer decides to change to the AD7877 touch-screen controller, (s)he can do so without input from their software team. Drivers are available for both devices; and while they differ and can connect differently (the AD7877 is SPI only, and the AD7879 is SPI or I^2C)—and they both have different register maps—the kernel API that is exposed to user code for touch screens is exactly the same. This puts control of the hardware back into the hands of the hardware architect.

Different types of device drivers in the Linux kernel provide different levels of abstraction. They are generally and historically classified into three categories.[11]

1. **Char (character) devices:** Handle byte streams. Serial port or *input device* drivers (keyboard, mouse, touch screen, joystick, etc.) typically implement the character device type.
2. **Block data devices:** Handle 512-byte or higher power-of-two blocks of data in single operations. Storage-device drivers typically implement this type of block device.
3. **Networking interface:** Any network transaction is made through an interface, that is, a device that is able to exchange data with other hosts.



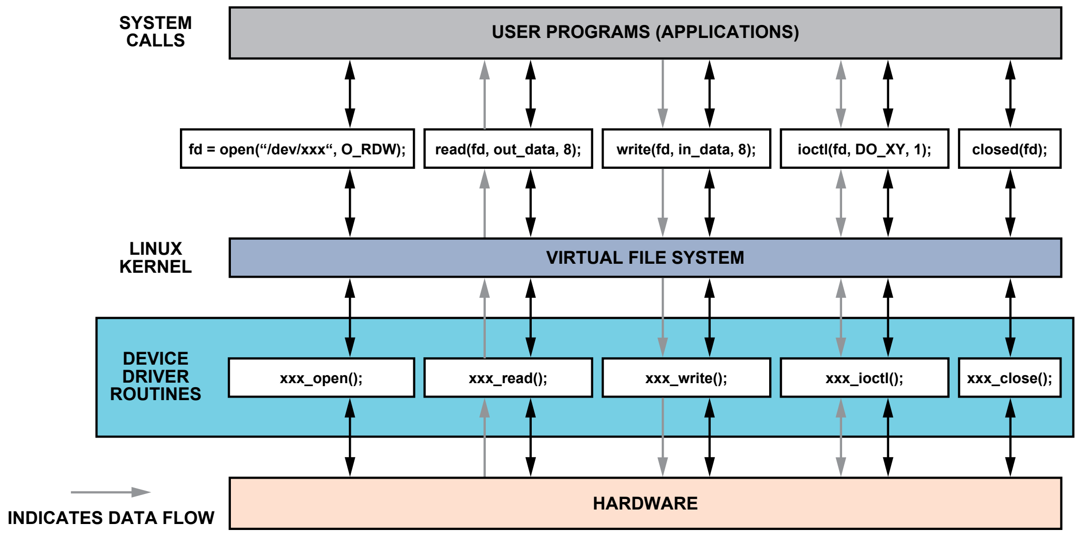
Each particular category may feature several independent device core layers within the Linux kernel, helping developers to implement drivers that serve standardized purposes—such as video, audio, network, input device, or backlight handling. Typically, each one of these subsystems has its own directory in the Linux kernel source tree. This *device driver core approach* removes code that would otherwise be common to all device drivers of a specific class and builds a standardized interface to the upper layer. Each class device, or bus device core driver, typically exports a set of functions to its child. Drivers register with such core drivers and use the API exported by the core driver instead of registering a character/block/network driver of their own. This typically includes support and handling for multiple instances—and the way data is distributed between layers. Huge portions of the system have very little interest in how devices are connected, but they need to know what kinds of devices are available. The Linux device model also includes a mechanism to assign devices to a specific class, such as input, RTC (real-time clock), net (networking), or GPIO (general-purpose input/output). These class names describe such devices at a higher, functional level and allow them to be discovered from user space.

There may be several device-driver subsystems associated with a particular piece of hardware. A multifunction chip, like the ADP5520 backlight driver with I/O expander, concurrently leverages the Linux backlight, LED, GPIO, and input subsystems for its keypad functionality.

As noted earlier, user applications are not allowed to communicate with hardware directly because that would require supervisor privileges on the processor, such as executing special instructions or handling interrupts. Applications utilizing a specific hardware device typically operate on kernel drivers exposed via nodes in the `/dev` directory.

Device nodes are called pseudofiles: they look like files; applications can also `open()` or `close()` them; but when they are read or written, the data comes from or is passed to the device nodes' associated driver. This level of abstraction is handled by the virtual file system (VFS) inside the Linux kernel. Besides `read()`, `write()`, or `poll()`, user applications may also interact with a device using `ioctl()` (input/output control).

In addition to the device nodes, applications may also utilize file entries in `/sys`, a sysfs virtual file system that exports information about devices and drivers, including parent/child relationship or association to a specific class or bus, from the kernel device model to user space. `/sys` is also heavily used for device



configuration, especially when the driver in question registers with a device driver core, which exports only its standardized set of functionality to the user.

Device drivers can register `/sys` *hooks* or *entries*; a specially registered callback function from the device driver gets executed when they are read or written. These callback functions—running in supervisor mode—may accept parameters, initiate bus transfers, invoke some processing, modify device-specific variables, and return integer values or character strings back to the user. This allows additional functionality; for example, the temperature sensor or auxiliary ADCs on the AD7877 touch-screen digitizer can be made available to user space.

Device drivers can be statically built into the kernel, or dynamically installed later as loadable modules. *Linux kernel modules* (LKMs) are dynamic components that can be inserted and removed at run time. This is especially valuable to driver developers since time is saved by quicker compilation and by not having to reboot the system to test the module. By letting the hardware drivers reside in modules that can be loaded into the kernel at any time, it is possible to save RAM when the specific hardware is not in use.

When a module is loaded, it can also be given configuration parameters. For a module that is built into the kernel, parameters are passed to it during the kernel boot. For example:

```
root:~> insmod ./sample_module.ko argument=1
root:~> lsmod
Module                    Size  Used by
sample_module 1396 0 - Live 0x00653000
root:~> rmmod sample_module
```

Drivers can also be instantiated multiple times, with different settings, with the target device sitting on a different I^2C slave ID, connected to a different SPI slave select, or mapped to a different physical memory address. All instances share the same code, which saves memory, but will have individual data sections.

Since Linux is a preemptive multitasking, multiuser operating system, almost all device drivers and kernel subsystems are designed to allow multiple processes (possibly owned by different users) to leverage the devices concurrently. Popular examples are the *network*, *audio*, or *input* interfaces. Key-press or -release events of an ADP5588 QWERTY keypad controller are time-stamped, queued, and sent to all processes that opened the *input event device*. These event codes are the same on all architectures and are hardware independent. There is no difference between reading a USB keyboard and reading the ADP5588 from user space. Event types are differentiated from codes. A keypad sends key-events (`EV_KEY`), together with codes identifying the key and some state value representing the press- or release action. A touch screen sends absolute coordinate events (`EV_ABS`) with a triplet consisting of x, y, and touch pressure, while a mouse sends relative movement events (`EV_REL`). An ADXL346 accelerometer may send key events for tap or double taps while it sends absolute-coordinate events for the acceleration.

In some applications, it could also make sense if the ADXL346 accelerometer generated relative events, or sent a specific key code—very application-specific settings. In general, there are two ways of driver customization: during run time or during compile time.

Device characteristics that are likely to be customized during run time use module parameters or `/sys` entries.

Using an Open-Source Linux Driver—Customization for a Specific Target

For compile time configuration, it's common Linux practice to keep board- and application-specific configuration out of the main driver file, instead putting it into the *board support file*.

For devices on custom boards, as typical of embedded and SoC-(system-on-a-chip) based hardware, Linux uses `platform_data` to point to board-specific structures describing devices and how they are connected to the SoC. This can include available ports, chip variants, preferred modes, default initialization, additional pin roles, and so on. This shrinks the board-support packages (BSPs) and minimizes board and application specific `#ifdefs` in drivers. It is up to the driver's author to decide which tunables go into `platform_data` and which should have access during run time.

Digital accelerometer characteristics are highly application-specific and may differ between boards and models. The following example shows a set of these configuration options. These variables are fully documented in the header file, adxl34x.h (`include/linux/input/adxl34x.h`).

```
#include <linux/input/adxl34x.h>
static const struct adxl34x_platform_data
adxl34x_info = {
 .x_axis_offset = 0,
 .y_axis_offset = 0,
 .z_axis_offset = 0,
 .tap_threshold = 0x31,
 .tap_duration = 0x10,
 .tap_latency = 0x60,
 .tap_window = 0xF0,
 .tap_axis_control = ADXL_TAP_X_EN | ADXL_
TAP_Y_EN | ADXL_TAP_Z_EN,
 .act_axis_control = 0xFF,
 .activity_threshold = 5,
 .inactivity_threshold = 3,
 .inactivity_time = 4,
 .free_fall_threshold = 0x7,
 .free_fall_time = 0x20,
 .data_rate = 0x8,
 .data_range = ADXL_FULL_RES,

 .ev_type = EV_ABS,
 .ev_code_x = ABS_X,             /* EV_REL */
 .ev_code_y = ABS_Y,             /* EV_REL */
 .ev_code_z = ABS_Z,             /* EV_REL */

 .ev_code_tap = {BTN_TOUCH, BTN_TOUCH, BTN_
TOUCH}, /* EV_KEY x,y,z */

.ev_code_ff = KEY_F,             /* EV_KEY */
.ev_code_act_inactivity = KEY_A,/* EV_KEY */
 .power_mode = ADXL_AUTO_SLEEP | ADXL_LINK,
 .fifo_mode = ADXL_FIFO_STREAM,
};
```

To attach devices to drivers, the *platform and bus model* eliminates the need for device drivers to contain hard-coded physical addresses or bus IDs of the devices they control. The platform and bus model also prevents resource conflicts, greatly improves portability, and cleanly interfaces with the kernel's power-management features.

With the platform and bus model, device drivers know how to control a device once informed of its physical location and interrupt lines. This information is provided as a data structure passed to the driver during probing.

Unlike PCI or USB devices, I^2C or SPI devices are not enumerated at the hardware level. Instead, the software must know which devices are connected on each I^2C/SPI bus segment and what address (slave selects) these devices are using. For this reason, the kernel code must instantiate I^2C/SPI devices explicitly. There are different ways to achieve this, depending on the context and requirements. However, the most common method is to declare the I^2C/SPI devices by bus number.

This method is appropriate when the I^2C/SPI bus is a system bus, as in many embedded systems, wherein each I^2C/SPI bus has a number which is known in advance. It is thus possible to pre-declare the I^2C/SPI devices that inhabit this bus. This is done with an array of `struct i2c_board_info / spi_board_info`, which is registered by calling `i2c_register_board_info()/spi_register_board_info()`

```
static struct i2c_board_info __initdata bfin_
i2c_board_info[] = {
#if defined(CONFIG_TOUCHSCREEN_AD7879_I2C) ||
defined(CONFIG_TOUCHSCREEN_AD7879_I2C_MODULE)
 {
   I2C_BOARD_INFO("ad7879", 0x2F),
   .irq = IRQ_PG5,
   .platform_data = (void *)&bfin_ad7879_ts_
info,
 },
#endif
#if defined(CONFIG_KEYBOARD_ADP5588) ||
defined(CONFIG_KEYBOARD_ADP5588_MODULE)
 {
   I2C_BOARD_INFO("adp5588-keys", 0x34),
   .irq = IRQ_PG0,
   .platform_data = (void *)&adp5588_kpad_
data,
 },
#endif
#if defined(CONFIG_PMIC_ADP5520) ||
defined(CONFIG_PMIC_ADP5520_MODULE)
 {
   I2C_BOARD_INFO("pmic-adp5520", 0x32),
   .irq = IRQ_PG0,
   .platform_data = (void *)&adp5520_pdev_
data,
 },
#endif
#if defined(CONFIG_INPUT_ADXL34X_I2C) ||
defined(CONFIG_INPUT_ADXL34X_I2C_MODULE)
 {
   I2C_BOARD_INFO("adxl34x", 0x53),
   .irq = IRQ_PG0,
   .platform_data = (void *)&adxl34x_info,
 },
#endif
};

static void __init blackfin_init(void)
{
 (...)
 i2c_register_board_info(0, bfin_i2c_board_
info, ARRAY_SIZE(bfin_i2c_board_info));
 spi_register_board_info(bfin_spi_board_info,
ARRAY_SIZE(bfin_spi_board_info));
 (...)
}
```

So, to enable such a driver one need only edit the board support file by adding an appropriate entry to `i2c_board_info` (`spi_board_info`).

It has also been noted that the driver needs to be selected during kernel configuration. Drivers are sorted by subsystems they belong to. The ADXL34x driver can be found under:

```
Device Drivers  --->
  Input device support  --->
    [*]   Miscellaneous devices  --->
     <M>   Analog Devices AD714x Capacitance
Touch Sensor
        <M>     support I2C bus connection
        <M>     support SPI bus connection
     <*>   Analog Devices ADXL34x Three-Axis
Digital Accelerometer
        <*>     support I2C bus connection
        <*>     support SPI bus connection
```

Selected drivers will be compiled automatically once the user has started the kernel build process.

The above code declares four devices on I^2C Bus 0, including their respective addresses, IRQ, and custom platform_data needed by their drivers. When the I^2C bus in question is registered, the I^2C devices will be instantiated automatically by the `i2c-core` kernel subsystem.

```
static struct i2c_driver adxl34x_driver = {
 .driver = {
   .name = "adxl34x",
   .owner = THIS_MODULE,
 },
 .probe     = adxl34x_i2c_probe,
 .remove    = __devexit_p(adxl34x_i2c_
remove),
 .suspend   = adxl34x_suspend,
 .resume    = adxl34x_resume,
 .id_table = adxl34x_id,
};

static int __init adxl34x_i2c_init(void)
{
 return i2c_add_driver(&adxl34x_driver);
}
module_init(adxl34x_i2c_init);
```

At some point during kernel startup, or at any time later, a device driver named "`adxl34x`" may register itself, using `struct i2c_driver`—which is registered by calling `i2c_add_driver()`. Members of `struct i2c_driver` are set with pointers to ADXL34x driver functions, connecting the driver with its bus master core. (The `module_init()` macro defines which function (`adxl34x_i2c_init()`) is to be called at module insertion time.)

If the name of the driver that is filed matches the name given with the I2C_BOARD_INFO macro, the `i2c-core` bus model implementation invokes the driver's `probe()` function, passing it the associated `platform_data` and `irq` from the board support file to the driver. This only happens in cases without recourse conflicts, such as when a previously instantiated device uses the same I^2C slave address.

The `adxl34x_i2c_probe()` function then starts to do what its name implies. It checks if either an ADXL345 or ADXL346 device is present and functional by reading the manufacturer and device ID. If this succeeds, the driver's probe function allocates device-specific data structures, requests the interrupt, and initializes the accelerometer.

It then allocates a new input device structure using `input_allocate_device()` and sets up input bit fields. In this way, the device driver tells the other parts of the input systems what it is and what events can be generated by this new input device. Finally the ADXL34x driver registers the input device structure by calling `input_register_device()`.

This adds the new input device structure to linked lists of the input driver—and calls device-handler modules' connect functions to tell them a new input device has appeared. From this point on, the device may generate interrupts. The interrupt service routine, once executed, reads the status registers and event FIFOs from the accelerometer and sends appropriate events back to the input subsystem using `input_event()`.

Drivers Maintained by Analog Devices

A complete list of Linux drivers maintained by Analog Devices can be found in the mainline Linux kernel (at kernel.org) or within ADI's own Linux distribution website at https://docs.blackfin.uclinux.org/doku.php?id=linux-kernel:drivers. It includes a wide variety of drivers, from audio, digital potentiometers, touch-screen controllers, and digital accelerometers to ADCs and DACs.

To get help with these drivers, in the standard open-source fashion, Analog Devices sponsors a website that includes web forums and mailing lists at http://blackfin.uclinux.org/gf/—where a full-time ADI team is responsible for answering questions and handling requests about FOSS drivers in a timely manner.

References

(Information on all ADI components can be found at www.analog.com.)

[1] IDC study/survey from over 5000 developers in 116 countries. *Open Source in Global Software: Market Impact, Disruption, and Business Models*. 2006.

[2] http://en.wikipedia.org/wiki/Free_and_open_source_software.

[3] www.gnu.org/philosophy/free-sw.html.

[4] www.opensource.org/docs/osd.

[5] www.opensource.org/licenses/bsd-license.php.

[6] Riehle, Dirk. "The Economic Motivation of Open-Source Software: Stakeholder Perspectives." IEEE Computer, vol. 40, no. 4 (April 2007). pp 25–32. http://dirkriehle.com/computer-science/research/2007/computer-2007.pdf.

[7] Kübler-Ross, Dr. Elisabeth E. *On Death and Dying*. Routledge. ISBN 0415040159.

[8] Forrester Research. 1973. http://i.i.com.com/cnwk.1d/i/bto/20090521/Picture_2_610x539.png.

[9] All lines of source measurements were counted with David A. Wheeler's SLOCcount from http://www.dwheeler.com/sloccount/.

[10] Kroah-Hartman, Greg. SuSE Labs/Novell Inc., Jonathan Corbet, LWN.net, and Amanda McPherson. Linux Foundation; "*Linux Kernel Development: How Fast It Is Going, Who Is Doing It, What They Are Doing, and Who Is Sponsoring It*." www.linuxfoundation.org/publications/whowriteslinux.pdf.

[11] Corbet, Jonathan, Alessandro Rubini, and Greg Kroah-Hartman. *Linux Device Drivers, Third Edition*. http://lwn.net/Kernel/LDD3/.

Authors

Michael Hennerich [michael.hennerich@analog.com] joined Analog Devices in 2004. As a systems and applications design engineer he worked on a variety of DSP- and embedded-processor-based applications and reference designs. Michael now works as an open-source systems engineer in Munich. In this role he is the leading device driver and kernel developer for the Blackfin architecture Linux support. He holds an MSC degree in computer-based engineering and Dipl.-Ing. (FH) degree in electronics and information technologies from Reutlingen University.

Robin Getz [robin.getz@analog.com] joined ADI in 1999 as a senior field applications engineer, and he currently leads the Free and Open-Source efforts at Analog Devices, which include an entire Linux distribution for ADI processors, the GNU Toolchain, and platform independent device drivers for a variety of processors and operating systems. Prior to joining ADI, Robin held various positions at other multinational semiconductor manufacturers, where he obtained multiple patents. He received a B.Sc. in 1993 from the University of Saskatchewan.

Analog Devices, Inc.
Worldwide Headquarters
Analog Devices, Inc.
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9106
U.S.A.
Tel: 781.329.4700
(800.262.5643,
U.S.A. only)
Fax: 781.461.3113

Analog Devices, Inc.
Europe Headquarters
Analog Devices, Inc.
Wilhelm-Wagenfeld-Str. 6
80807 Munich
Germany
Tel: 49.89.76903.0
Fax: 49.89.76903.157

Analog Devices, Inc.
Japan Headquarters
Analog Devices, KK
New Pier Takeshiba
South Tower Building
1-16-1 Kaigan, Minato-ku,
Tokyo, 105-6891
Japan
Tel: 813.5402.8200
Fax: 813.5402.1064

Analog Devices, Inc.
Southeast Asia Headquarters
Analog Devices
22/F One Corporate Avenue
222 Hu Bin Road
Shanghai, 200021
China
Tel: 86.21.2320.8000
Fax: 86.21.2320.8222

©2010 Analog Devices, Inc. All rights reserved.
Trademarks and registered trademarks are the property
of their respective owners.

Printed in the U.S.A. M02000441-0-5/10

www.analog.com/analogdialogue

ANALOG DEVICES