. .eescale Semiconductor
Application Note

AN2013
Rev. 1, 10/2005

Characterizing CMOS DSP Core Current for Low-Power Applications

By David Dicarlo

As DSPs are used in battery-operated, portable applications, there is a need to conserve power and minimize heat dissipation. This application note discusses a simple, low-cost method of characterizing CMOS DSP current consumption on an application board while application code runs. The application code, code structure, and operating frequency can then be optimized for minimum power consumption.

Although only the measurement of the core supply current is presented in this application note, the methods discussed can also be applied to the I/O supply current, optimizing I/O structure and use. Additionally, several Freescale DSP563xx evaluation boards can be modified to enable core current measurements, eliminating the need to build a prototype application board with that capability.

CONTENTS

1 Power Consumption in CMOS Devices

Current in CMOS devices is the result of the current required to charge or discharge each node that switches from one power supply rail to the other. Nodes that do not switch do not contribute to the current because no additional charge is required to maintain the state of the nonswitching nodes. The amount of charge required to switch a node from one rail to the other can be mathematically expressed as follows:

Equation 1

$$Q_{node} = C_{node} \times V_{supply}$$

where:

- Q_{node} = the amount of charge in coulombs required.
- C_{node} = the capacitance of the node in farads.
- V_{supply} = the voltage difference between the supply rails.

Alternately switching a node from one supply rail to the other produces a current that is directly proportional to the switching rate, or frequency. Current is expressed in terms of charge per unit of time, and can be mathematically expressed as follows:

Equation 2

$$I_{node} = Q_{node} \times frequency = C_{node} \times V_{supply} \times frequency$$

where:

- I_{node} = current required to switch a single node.
- frequency = rate in hertz that the node is switched between rails.

By summing the current for all switching nodes, the total device current may be expressed as:

Equation 3

$$I_{device} = C_{switch} \times V_{supply} \times frequency$$

where:

- I_{device} = total device current.
- C_{switch} = total capacitance of nodes that are switching.

In a CMOS-based DSP, because the number of nodes that switch changes as application code executes and data is processed, the total capacitance of switching nodes changes as well. **Equation 2** is better expressed with the total device current and switching capacitance are functions of time, as shown in **Equation 4**:

Equation 4

$$I_{device}(t) = C_{switch}(t) \times V_{supply} \times frequency$$

where:

- $I_{device}(t)$ = total device current as a function of time.
- $C_{switch}(t)$ = total capacitance of switching nodes as a function of time.

While it is impossible to calculate the actual number of switching nodes as a function of time for any given instruction and data set, it is necessary to understand how application software code, the use of on-chip resources, and clock frequency can affect DSP current as a function of time.

1.1 DSP563xx Hardware

The DSP563xx family of CMOS-based DSPs is based on a 24-bit, fixed-point DSP core surrounded by differing amounts of internal SRAM and a complement of peripherals and general-purpose I/O (GPIO) pins. Internal processor speeds of up to 275 MHz are available in some derivatives, and because the core can execute one instruction per clock cycle, a rate of up to 275 million instructions per second (MIPS) is possible. The devices are implemented in high-density CMOS fabrication technologies that exhibit low operating currents with supply voltages in the range of 3.3 V down to 1.6 V.

The fully static design of the DSP563xx family devices allows the use of low-power operating modes in which the core and possibly all the peripherals halt when no activity is required. The full state of the processor is preserved and execution can resum without any loss in data or position in code.

The DSP clock is provided by an internal phase-locked loop (PLL) clock generation circuit, allowing the use of relatively low frequency external crystals or clock oscillators and eliminating the challenge of providing an external full-speed clock signal. The PLL output frequency is fully programmable and can be disabled during one low-power mode, all through software control. The internal clock is also available for use by on-chip peripherals.

The DSP563xx core has internal SRAM that operates with zero wait states and is organized in three banks: one bank of program memory and two banks of X and Y data memory. The size of each bank of memory depends on the particular derivative device. Three independent internal addresses and data buses allow for multiple data paths. On-chip memory capacity has increased with each new member of the DSP family, further decreasing the need for external memory for application code and data storage.

An external memory interface, port A, is provided for memory expansion outside the device and for off-chip memory-mapped peripherals. Port A requires at least one wait state and is therefore limited to operation at half the internal bus frequency. Port A can also be programmed to mirror internal memory addresses for use in debugging. Additionally, the port A controller can be disabled in power-sensitive applications where it is not required.

Each member of the DSP563xx family is surrounded by several on-chip peripherals that assist in moving data in, out, and around the DSP and that provide event timing and measurement support. Additionally, the DSP56311 and DSP56321 each have an on-chip filter coprocessor that can perform filter operations in hardware, therefore reducing the MIPS load on the core. The DSP5636x derivatives peripherals t specifically designed for digital audio applications. In most instances, the pins of unused peripherals can be used as GPIO. Each peripheral derives its power from the device V_{IO} supply. Low-power applications call for the judicious use of each peripheral because each one adds to the current on that power supply.

1.2 Operating Modes

There are three operating modes: Normal, Wait, and Stop. In Normal mode, the PLL and clock generation circuitry, the core, and all the enabled peripherals are active and application code executes. Wait and Stop are low-power modes of operation.

In Wait mode, the PLL and clock generation circuitry and all enabled peripherals are active, but the clock signal to the core and to the internal memory is gated off and the processor is halted. Any unmasked interrupt or external hardware reset causes the processor to wake up to process the exception condition. The advantage of Wait mode is that the on-chip peripherals remain active. Data streams can be processed but power can be conserved during idle periods of the streams.

Stop mode offers the lowest power mode of operation. The clock signal is gated off internally and all the core and peripheral activities are suspended. A hardware reset or external interrupt on IRQA brings the DSP out of Stop mode, and processing continues. Because Stop mode can interrupt peripherals during data transfers, Stop mode should be used only after careful consideration of the implications to the application. Additionally, the clock circuitry can be fully disabled in Stop mode, further reducing power, but doing so requires a clock-stabilization delay on wake-up.

1.3 Application Code

Application code plays a role in the power consumed by a CMOS DSP, both microscopically in how linear or parallel the code is and macroscopically in how the overall application code is structured.

The three independent internal buses of the DSP563xx family of devices allow up to two additional data moves to be performed in parallel with data ALU instructions, enabling data to be preloaded and moved around in the processor more efficiently. The MIPS capacity is effectively increased because fewer actual instruction cycles are required to perform an application task, resulting in a performance boost. Linear code performing the same task at a given operating frequency takes longer to complete than equivalent, highly parallel code executing at the same frequency. Because the use of parallel moves adds to the number of switching nodes, highly parallel code consumes more current than equivalent linear code at the same operating frequency.

On a higher level, the overall structure of application code can affect current consumption. The natural tendency is to use the DSP at its maximum operating frequency. In many applications, the MIPS capacity of the DSP exceeds what is required, so the processor may sit idle during periods when no processing is required. While the amount of current wasted during idle periods can be reduced by the use of the Wait and Stop modes, you can also reduce the clock frequency and organize the code so that processor capacity matches the demands of the application and there is little or no idle time.

1.4 Measuring Current

Although understanding the factors that influence DSP core current is helpful, it is more useful to quantitatively measure actual device current while varying any one or all of these factors. Of most interest to the designer of low-power applications is:

- variation of core current with clock frequency
- organization and linearity of the application code
- use of on-chip resources.

Current is usually measured with an ammeter, and the voltage drop across a known resistance is used to calculate the current using ohm’s law. In **Figure 1**, which shows a typical schematic for measuring current, R and V comprise an ammeter.

Making low-current measurements is a challenge. A small current requires larger resistance to obtain a useful measurement of voltage. However, the larger resistance reduces the voltage across the device. In addition, if the current changes in time, the voltage across the device changes as well, because the voltage drop across the measurement resistance changes. To achieve desired accuracy in the current measurement, there must be a compromise between how much the device voltage is lowered and how measurable the voltage drop is.

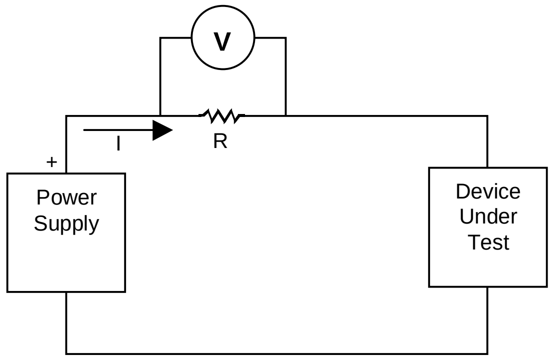

Figure 1. Typical Schematic for Current Measurement

For CMOS DSPs, current measurements are typically made with expensive current probes (very low resistance with a very sensitive volt meter) or with an ammeter with a very low series resistance using circuit boards designed specifically for measuring DSP current. The disadvantage is twofold: the cost of entry to make the measurements is high (in both equipment cost and custom hardware to make the measurement) and the measurements still suffer from the perturbation caused by making the measurement in the first place.

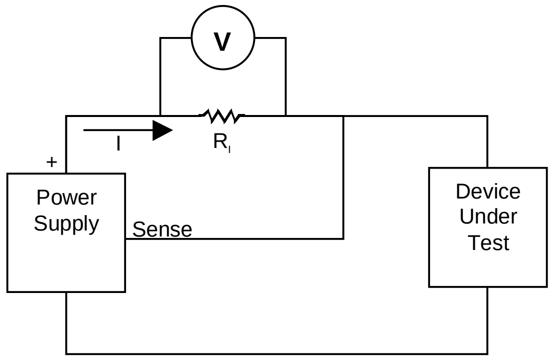

Figure 2. Schematic for Measuring Current Using a Current-Sensing Resistor

Figure 2 shows a power supply that can accurately measure DSP current while avoiding the disadvantages of the typical method. The current supplied to the device can be calculated by measuring the voltage across the current-sensing resistor, R_I. The output is held constant via the external output voltage-sensing feature of the voltage regulator. The current in R_I includes the current required by the regulator to sense and control the output voltage, but the appropriate selection of a voltage regulator and the voltage-sense circuitry can make that current much smaller than the device current. The advantages of using the power supply to make the device current measurement are that its cost is very low and that it can be included on an application board very easily. In addition, an application board can be easily designed or modified to accommodate making these measurements with an outboard current-sensing power supply.

2 Demonstrating Low-Power Measurement

This section presents the power supply prototype and test codes used to demonstrate low-power measurements methods. Also presented are the results and analysis of current data. The section concludes by presenting the details for modifying several DSP563xx evaluation boards.

2.1 Power Supply Prototype

The National Semiconductor LP2960 adjustable regulator was selected for prototyping the current-sensing power supply because of its ready availability and the following features:

- output is easily programmed externally
- operation can be configured to 1.8 V (required for the DPS56311)
- low-quiescent current

Figure 3 shows the prototype power supply schematic. Fine adjustment of the output voltage was possible using the 10-turn, 10 Ω trimpot, R_{TRIM}. The output voltage was programmed to 3.3 V, 2.5 V, and 1.8 V by replacing resistor R_{OUT}. The values for each output voltage are listed in **Figure 3**. Device supply current is measured using the voltage drop across R_I, a 1 Ω, 1% resistor, where 1 mV is equivalent to 1 mA.

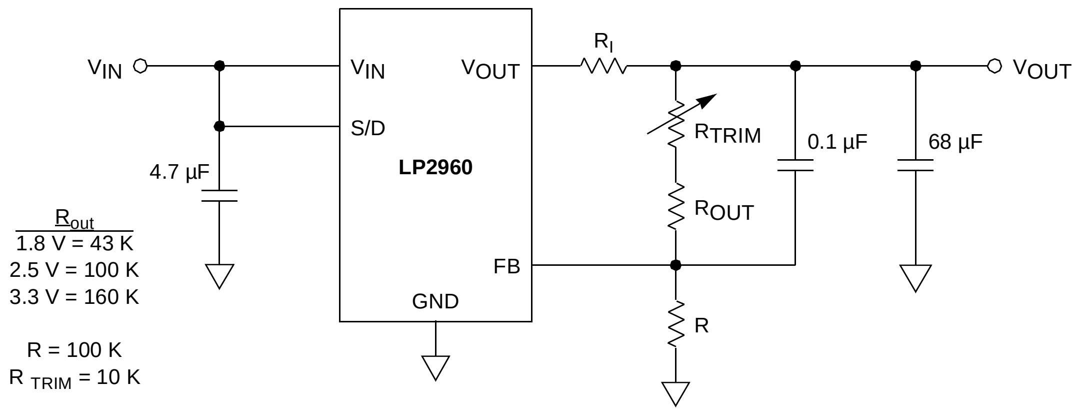

Figure 3. Schematic of the LP2960 Current-Sensing Power Supply

According to the LP2960 data sheet, at least 12 µA of current is required in the voltage programming network for stability (R_{OUT}, R_{TRIM}, and R in **Figure 3**). The values selected for the prototype circuit drew about 20 µA, meeting the requirements for stability but minimizing the contribution to the device current measured through R_I.

Figure 4 shows the supply current of a DSP56309 executing an FFT benchmark at 100 MHz (top trace) along with the output voltage of the regulator (bottom trace). The regulator output could maintain a fairly constant output voltage, even in the presence of a dynamic output current. Although there was some fluctuation in output voltage that would not be desirable in an actual application, the fluctuation was noticed only for large swings in supply current, such as at the end of the FFT kernel. The stability is more than sufficient to characterize the DSP core current during development and periods of relatively constant supply current.

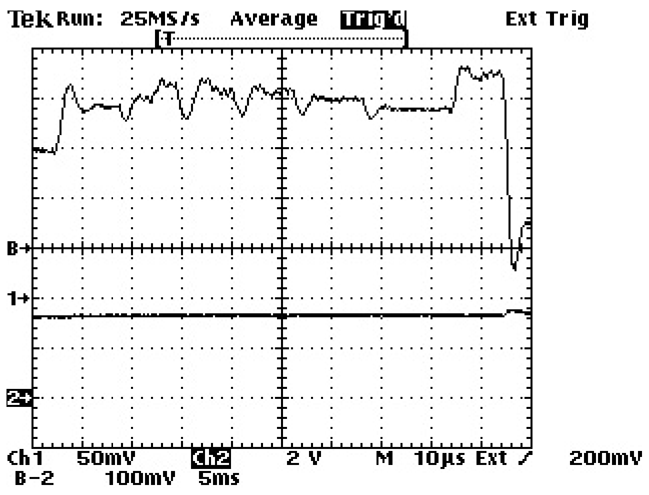

Figure 4. DSP Supply Current

The initial DSP core current values were measured on a custom PC board with a socket that allows most members of the DSP563xx family to be installed. The board was configured according to the schematic in **Figure 5**. Three separate power supplies supplied independent power to the PLL, the core, and the DSP I/O and peripheral circuitry.

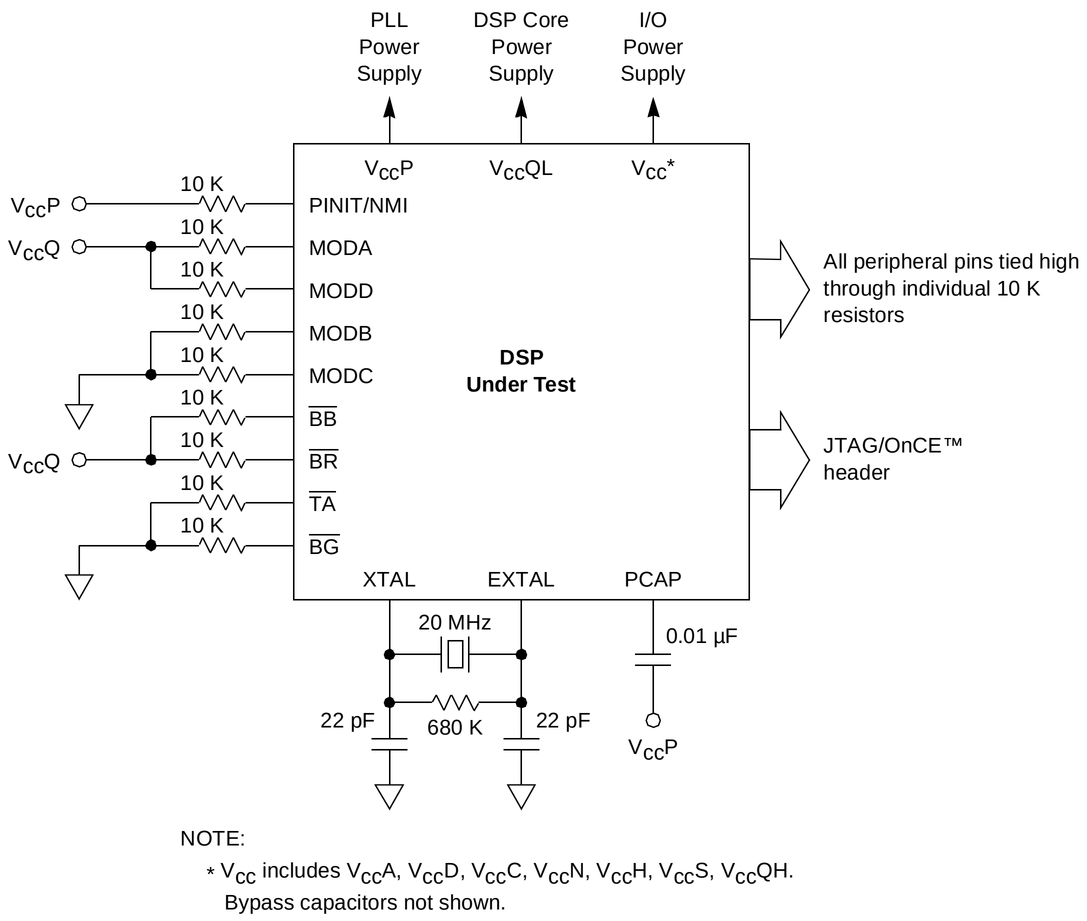

Figure 5. DSP Test Circuit Schematic

The prototype, current-sensing power supply was used as the DSP core supply. All the unused inputs were terminated with 10 KΩ resistors tied to the I/O power supply. A 20 MHz crystal was used for the clock input. The core current of five members of the DSP563xx family were measured, as shown in **Table 1**, which details the maskset information, fabrication process technology, and core supply voltage.

Table 1. Process Information for the Devices Measured

Device	Maskset	Process Technology	Core Supply Voltage
DSP56301	2K30A	0.32 µm CDR2	3.3 V
DSP56303	0K36A	0.32 µm CDR2	3.3 V
DSP56307	J22D	0.32 µm CDR2	2.5 V
DSP56309	0J17D	0.32 µm CDR2	3.3 V
DSP56311	2J23D	0.23 µm HiP4DSP	1.8 V

2.2 Test Code

We demonstrated the effect of the application test code on DSP core current using a battery of algorithms, as listed in **Table 2**. With the exception of a Freescale-developed GSM 6.10 speech encoder, all the test algorithms came from code codeveloped with Berkeley Design Technology, Inc. (BDTI) for the publication of the 2001 edition of BDTI's *Buyer's Guide to DSP Processors*. All the routines were modified to loop endlessly, and, where necessary, the test data supplied with the benchmark was reset to the original values before each new pass of the kernel. **Table 2** shows several routines that were modified to toggle the TIO0 pin signal for use in triggering an oscilloscope as the main kernel of the algorithm executed.

Table 2. Information for Test Code to Demonstrate the Current-Sensing Power Supply

Benchmark	Description	Comments
BITUPK	Bit unpacking	
BLKFIR	Block-based real FIR filter	Benchmark modified to provide oscilloscope trigger via TIO0.
CONTROL	Control benchmark	
CXFIR	Block-based complex FIR filter	Benchmark modified to provide oscilloscope trigger via TIO0.
FFT99	256-point complex FFT	Benchmark modified to provide oscilloscope trigger via TIO0.
GSM (MOT)	GSM speech encoder	
IIR	Cascaded biquad IIR filter	
LMS	LMS adaptive filter	Benchmark modified to provide oscilloscope trigger via TIO0.
SSFIR	Sample by sample real FIR filter	
VECMAX	Vector maximum	
VECPROD	Real vector product	
VECSUM	Real vector sum	
VITERBI	Viterbi decoder	

The code was downloaded to the DSP through the JTAG/OnCE™ port using a command converter. The PLL frequency was set manually with the debugger to one of three different frequencies for each processor. The frequencies measured at 50 MHz, 75 MHz, and 100 MHz, except for the DSP56311, which was also measured at 150 MHz. The DSP current was allowed to settle for several minutes, and then the current was recorded.

3 Current Measurement Results and Analysis

The results of the current measurements for each test algorithm are detailed in **Table 3**. Each value shown is the average current of the algorithm kernel expressed in mA. All the algorithms had fairly constant core currents throughout the kernel except for the FFT99. The FFT99 profile, shown in **Figure 6**, was typical for all the DSPs measured; the only differences were in magnitude and width.

Table 3. Average DSP Core Current for Each Test Algorithm (All Currents in mA)

	DSP56301			DSP56303			DSP56307			DSP56309			DSP56311			
FREQ (MHz)	**50**	**75**	**100**	**50**	**75**	**100**	**50**	**75**	**100**	**50**	**75**	**100**	**50**	**75**	**100**	**150**
BITUPK	78	110	140	76	105	134	63	85	108	86	115	155	52	68	84	112
BLKFIR	86	120	155	88	122	155	70	94	120	94	130	160	58	76	94	130
CONTROL	68	89	115	70	93	118	55	74	92	77	105	125	44	58	72	98
CXFIR	92	134	175	98	136	175	80	108	137	105	150	190	60	82	100	145
FFT99	95	135	170	95	135	175	78	108	135	105	150	190	68	90	110	150
GSM (MOT)	75	102	130	76	105	134	60	85	108	83	118	147	52	68	86	116
IIR	76	104	132	78	105	134	64	85	106	85	115	148	52	70	84	114
LMS	90	123	160	92	125	160	75	101	130	100	140	175	60	80	100	140
SSFIR	80	110	140	84	110	144	68	90	114	90	120	155	52	70	90	124
VECMAX	88	119	150	89	120	155	73	98	125	98	135	172	58	76	97	134
VECPROD	80	110	140	85	112	146	65	90	112	89	122	154	56	76	96	128
VECSUM	88	120	155	90	121	157	75	102	130	98	137	173	60	82	104	144
VITERBI	85	116	145	85	117	150	70	98	124	94	130	165	66	86	98	134

In general, the DSP56309 consumed the most current of all the DSPs measured while the DSP56311 consumed the least. The DSP56309 probably consumed the most current because it contains the largest internal memory space of the 0.32 µm CDR2 technology processors. The DSP56311 had the lowest operating voltage at 1.8 V. Additionally, the DSP56311 is fabricated from a smaller, 0.23 µm, geometry and a newer process technology.

None of the algorithms contained code to take advantage of the enhanced filter coprocessor (EFCOP) in the DSP56307 and DSP56311, so the effect of the EFCOP on DSP core current could not be assessed.

As scanning any column in the table demonstrates, for any DSP at a given frequency, the core current varied depending on the test algorithm. The CONTROL benchmark exhibited the lowest core current, while the FFT99 and CXFIR benchmarks typically exhibited the highest current for a given processor at a given frequency (that is, the lowest and highest values, respectively, in any particular column).

To understand the difference in observed core current between test algorithms, the number of cycles that used parallel instructions was counted for the kernel of each test, excluding any modifications that were required to enable continuous looping and oscilloscope triggering. **Table 4** shows the results for the CONTROL, CXFIR, and

FFT99 algorithms. The "Cycles" column[1] shows the total number of cycles required for one pass through the kernel. The "Parallel" column shows the number of cycles in which parallel instructions occurred, and the "% Parallel" column shows the percentage of the total cycles in the kernel where parallel instructions occurred.

The CONTROL code contains no parallel instructions. Fifty percent of the cycles in the CXFIR code were parallel instructions, and eighty-five percent of the cycles in the FFT99 code used parallel instructions. Although these algorithms do not perform equivalent tasks, at a given operating frequency the extra current from non-linear code is clearly demonstrated (more nodes are switching).

Table 4. Total and Parallel Cycle Counts for CONTROL, CXFIR, and FFT99 Algorithms

Benchmark	Description	Cycles	Parallel	% Parallel
CONTROL	Control benchmark	840	0	0%
CXFIR	Block complex FIR	2866	1442	50%
FFT99	256-point FFT	8825	7524	85%

The core current for each processor for every test algorithm was plotted against operating frequency to verify the linear relationship between core current and frequency. **Figure 6** shows the data plotted for the FFT99 algorithm and is representative of the data for all the other algorithms, which were similarly linear.

There is a linear relationship between core current and operating frequency for a given algorithm. The current at any valid operating frequency can be calculated from the slope and Y-intercept of the best-fit line to the data. The calculated slope and Y-intercept values for each test algorithm are tabulated in **Table 5**. The slope is expressed in units of mA/MHz, but it can also be expressed as mA/MIPS because the DSP563xx core can execute a single instruction every clock cycle. The Y-intercept is non-zero in all cases, and it is expressed in units of mA. We can see that the typical value for mA/MIPS published by DSP manufacturers overestimates the slope of the best-fit line because the typical published value assumes that the Y-intercept is zero. Instead, in all cases the typical published value is higher than the calculated slope.

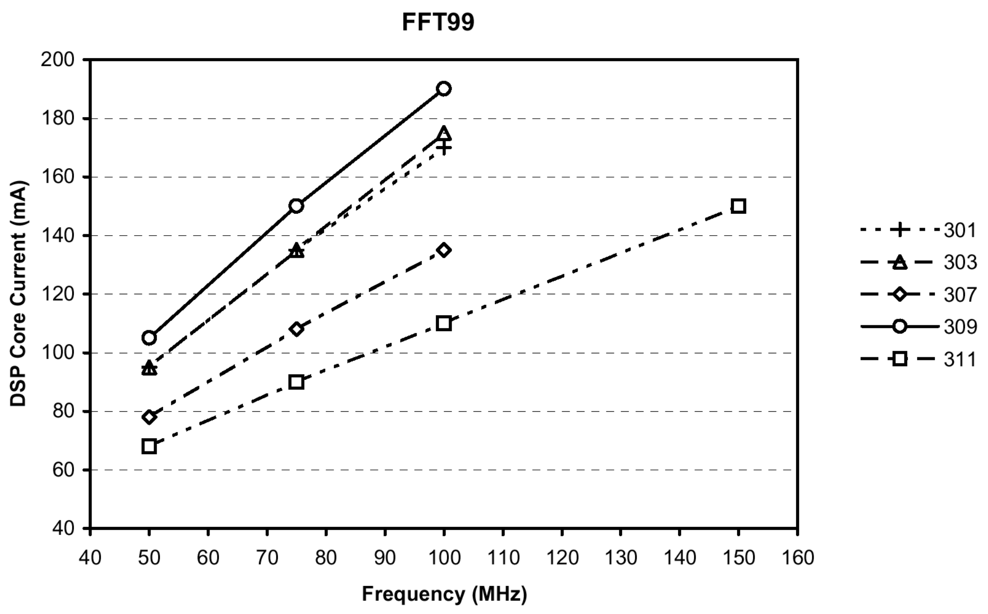

Figure 6. Linear Relationship Between DSP Core Current and Frequency

1. Benchmark cycle counts used with permission, © 2001 Berkley Design Technology, Inc. www.bdti.com

Table 5. Slope and Y-Intercept of the Best-Fit Line for Each Algorithm and Processor

	Slope (mA/MHz)					Y-Intercept (mA)				
	56301	**56303**	**56307**	**56309**	**56311**	**56301**	**56303**	**56307**	**56309**	**56311**
BITUPK	1.24	1.16	0.90	1.38	0.60	16.3	18.0	17.8	15.2	22.9
BLKFIR	1.38	1.34	1.00	1.32	0.72	16.8	21.2	19.7	29.0	22.0
CONTROL	0.94	0.96	0.74	0.96	0.54	20.2	21.7	18.2	30.3	17.4
CXFIR	1.66	1.54	1.14	1.70	0.84	9.2	20.8	22.8	20.8	17.6
FFT99	1.50	1.60	1.14	1.70	0.82	20.8	15.0	21.5	20.8	28.0
GSM (MOT)	1.10	1.16	0.96	1.28	0.64	19.8	18.0	12.3	20.0	20.3
IIR	1.12	1.12	0.84	1.26	0.61	20.0	21.7	22.0	21.5	22.6
LMS	1.40	1.36	1.10	1.50	0.80	19.3	23.7	19.5	25.8	20.0
SSFIR	1.20	1.20	0.92	1.30	0.72	20.0	22.7	21.7	24.2	16.3
VECMAX	1.24	1.32	1.04	1.48	0.76	26.0	22.3	20.7	24.0	19.6
VECPROD	1.20	1.22	0.94	1.30	0.72	20.0	22.8	18.5	24.2	21.7
VECSUM	1.34	1.34	1.10	1.50	0.84	20.5	22.2	19.8	23.5	18.9
VITERBI	1.20	1.30	1.08	1.42	0.67	25.3	19.8	16.3	23.2	33.4

The slope of the best-fit line varies in magnitude depending on the algorithm, with CONTROL having the shallowest slope and CXFIR and FFT99 the steepest. All the other algorithms more or less follow the same ranking as the DSP core current values. This, again, appears to relate directly to the linearity of the application under execution, and it is consistent with the relationship expressed by **Equation 2**. With more switching nodes, the value of C_{switch} increases, effectively increasing the slope of the current with respect to frequency.

The average of every column and row of the Y-intercept data, as well as every value in **Table 5**, is approximately 20 mA, with the average of all values being 20.9 mA. These averages indicate that the Y-intercept is a constant across all the processors measured, meaning that all current-versus-frequency lines pivot on the point at 20 mA and 0 MHz, with the slope of the line being dependent on the degree of non-linearity of the code under execution.

The measurements for this application note reveal that several Freescale DSP evaluation boards can easily be modified to perform DSP core current measurements. In particular, the DSP56303EVM (100 MHz version only), DSP56307EVM, and DSP56309EVM can be used if the current-sensing power supply output is applied to the center pin of jumper block J11. The DSP56303EVM requires a trace to be cut on the bottom side of the board because no jumper is present and the connection is hard-wired. The DSP56311EVM can be used by removing jumper J10 and applying the external power supply output to the L5 side of the jumper block. In all cases, the externally applied voltage must be appropriate for the particular DSP and should follow the application of power to the board power supply input. Better still, the input for the current-sensing power supply should be derived from the output of the EVM bridge rectifier so that the core of the DSP and the I/O power supplies come up in unison.

4 Summary and Conclusion

The use of a current-sensing power supply was successfully demonstrated by measuring the core current of five members of the DSP563xx family. While there was some fluctuation in output voltage with DSP core current, the power supply is suitable for use in optimizing an application for sensitive, low-power applications. It avoids the expense of low series resistance current probes and problems with using a traditional ammeter. The additional current from the output voltage programming resistors was only about 20 µA, which is negligible compared to the DSP core current. While no attempt was made to optimize the prototype power supply, there are undoubtedly other power supply designs that would reduce the amount of noise for very small currents and that would better respond to large and quick changes in current.

When DSP core current of five members of the DSP563xx family was measured using the prototype current-sensing power supply with several different test algorithms, the core current was found to vary in magnitude in relation to the non-linearity of the application code —that is, the amount of code that used parallel moves. The core current was also found to vary linearly with frequency for each combination of processor and test algorithm. The slope of the best-fit line varied in magnitude in the same way current did in relation to the test algorithm for any given combination. These findings are consistent with the relationship between core current, frequency, switching node capacitance, and voltage. The Y-intercept of all the best-fit lines for core current versus frequency appeared to be a constant across all the processors, frequencies, and test algorithms used. This indicates that the core-current-versus-frequency relationship for any application code intersects at a common point at 20 mA at 0 MHz.

Document Order No.: AN2013
Rev. 1
10/2005