An FPGA “Companion” in Smartphone Design

A Lattice Semiconductor White Paper

May 2012

Lattice Semiconductor
5555 Northeast Moore Ct.
Hillsboro, Oregon 97124 USA
Telephone: (503) 268-8000
www.latticesemi.com

Introduction

What do the advertisements of the world's leading phone makers have in common?

Answer: they almost never say anything about using their products to make an actual phone call – unless it happens to be some form of video call.

Instead, they promise to connect you with anyone so that you can share anything. They tout the hundreds of thousands of apps that you can download. These apps promise to do everything from helping you clean your carpet to navigating by the stars to a faraway land, all the while tweeting a stream of real-time updates to your faithful followers the world over.

Now, all of this might seem like just interesting marketing trivia, but if we dig a little deeper, we find a profound engineering situation that impacts every smartphone designer.

By adding an Internet browser, the iPhone created a fundamental change in the phone's use model. Instantly, the feature phone became a 24/7 mobile portal to all that the World Wide Web has to offer. Today, there are approximately 100,000 unique application developers, servicing all the various smartphones. The size of this ecosystem is unparalleled in the industry, and one of its many effects on smartphones is the rapid generation of new requirements. For example, game developers are clamoring for bigger, better displays as well as smartphone output connections to even bigger external displays. After all, probably the only thing more righteous than destroying that last pig in "Angry Birds" is destroying it on a high-definition big screen in front of all your friends.

The Effect on Smartphone Designers

This accelerated pace to meet new requirements produces very short product development cycles. And the pressure to meet these schedules causes more reliance on standard chips – i.e. fully loaded application processors (Systems on a Chip, or SoC), such as the Qualcomm Snapdragon or the TI OMAP. But now we have a

paradox: big chips like application processors take two to three years to develop, meaning that any device you can buy today was defined two or three years ago – which, at the pace of the smartphone evolution, is an eternity.

Another way to look at this is from the application processor designer's perspective. They must define tomorrow's chip using today's knowledge. But even if they succeed, as soon as the application processor is released in a new smartphone, one hundred thousand developers will start working on it in an effort to do something new and different, generating yet more requirements.

So what is a smartphone designer supposed to do? In order to meet your design schedules, you must use readily available chips. But yesterday's application processors often fail to meet today's requirements. So you need either to delay your ideas until the application processor supports them directly (when all of your competitors will also have access) or you need to supplement the application processor with something else.

Bridging the Application Processor Development Cycle Gap

One solution would be to use an FPGA. You could then do what you wanted with no waiting. But, until recently, this was not an option. FPGAs were simply too big, too expensive and too power-hungry for use in smartphones. This is because traditional FPGAs have been designed for applications that aren't limited by the price, power and space constraints of smartphones.

Increasingly, however, you can find FPGAs that are specifically targeted at the needs of small, inexpensive, power-sensitive consumer devices – and smartphones in particular. This new breed of FPGA can be used to supplement an application processor and help bridge the gap in the processor release cycle (Figure 1).

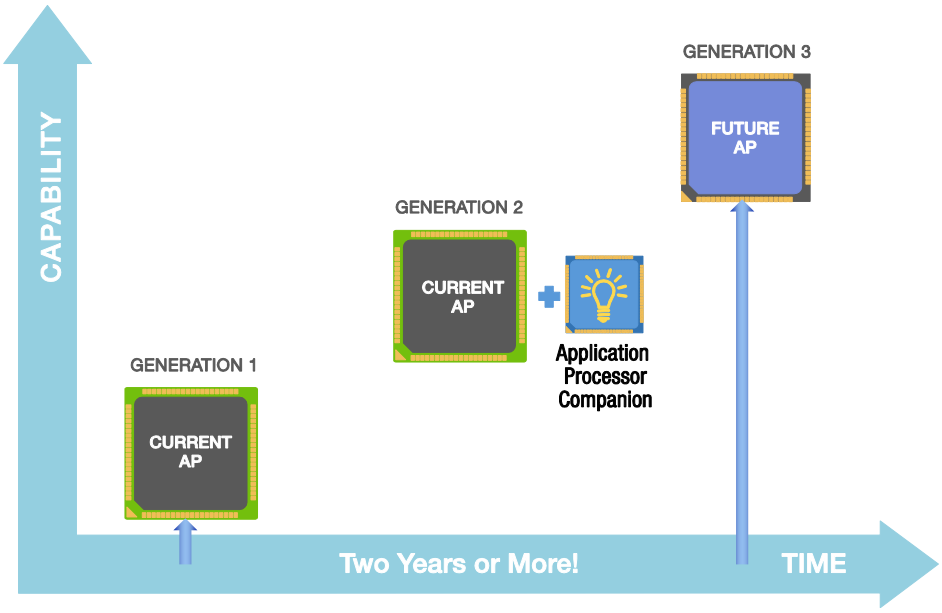

Figure 1

Sensor Surplus

No component inside a phone better illustrates the situation we've described than the sensor. Completely unnecessary for making a phone call, sensors originally were used for simple functions such as adjusting the display's backlighting based on the ambient light. But sensors provide much information on the phone's environment, and that information can be used in lots of different ways. As a result, more and more sensors are being added to smartphones: today's smartphone can have between one and two dozen different sensors in it.

This creates a problem for the designer, however: how to handle the information coming from the sensors? The obvious answer is to feed the sensor output to the application processor, but processors typically have too few of the I^2C or SPI ports that sensors use. This means finding some other way of multiplexing or using general-purpose I/Os (often with poor performance) to handle the sensor glut.

But here's the challenge: a phone is tiny, and space matters above all else – even more than power. You can't just put multiple discrete components in the phone to increase

the number of sensors. Now, if this were any other kind of non-consumer system, the obvious solution would be to use an FPGA. But, according to conventional wisdom, no one would put an FPGA in a phone: they're too big, they're too expensive and they're too power-hungry.

And that's because most FPGAs have been designed for the large number of applications that aren't bothered by these characteristics. Increasingly, however, you can find FPGAs that are specifically targeted at the needs of small, inexpensive, power-sensitive consumer devices – and phones in particular. These are the application processor companions that can bridge the promise of new hardware possibilities – like dozens of sensors – to the realities of what the application processors can do.

FPGAs in Smartphones

Managing sensors is a good example of how such an application processor companion can make tomorrow's SoC capability available today. The most basic issue to be addressed is the mismatch between the number of sensors and the number of ports available on the application processor.

For example, a Qualcomm baseband processor – say, the MSM8X55 – can only handle up to three I^2C inputs. A smartphone today will have, at a minimum, three positioning sensors – accelerometer, gyroscope and compass –along with a touch panel and a battery monitor. In order to appeal more to fitness enthusiasts, for example, you may want to add heart rate and perspiration monitors as well as an altimeter (for hikers). The touchscreen has to have its own input so that no touch events are missed while some other sensor is being polled. That means all the other sensors would have to share the remaining two I^2C busses.

An application processor companion can be programmed to add more I^2C masters and be a bridge between the sensors and Qualcomm's external bus interface (EBI2). As shown in Figure 2, all of the I^2C functionality is handled in the application processor companion, and the sensor output is then buffered in FIFOs and delivered through a high-speed EBI2 interface without using any I^2C ports at all.

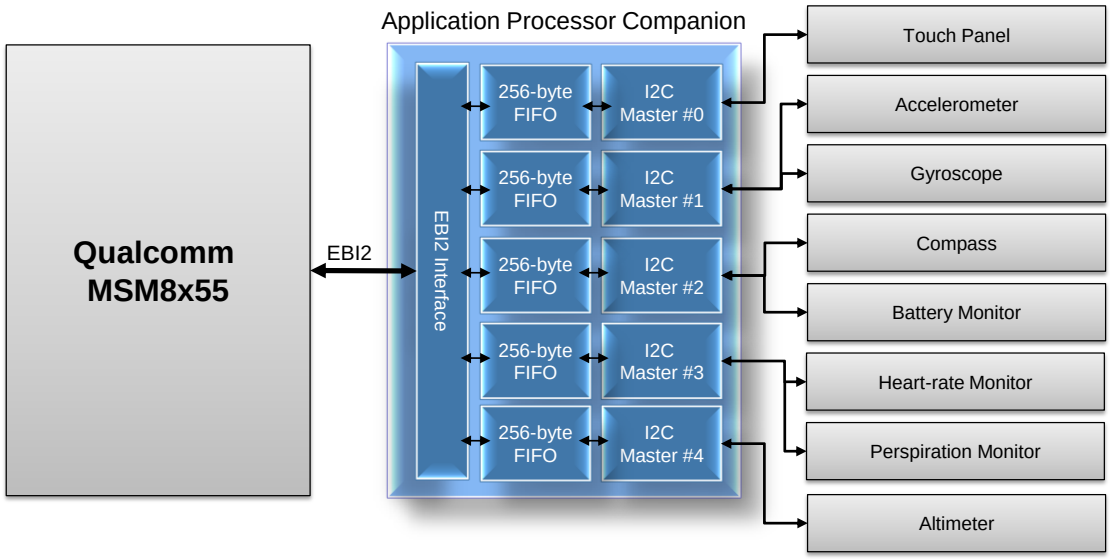

Figure 2

Offloading the Application Processor

Expanding the number of ports available to an application processor can be helpful, but it can also make a different problem worse. The application processor typically has to do work to process sensor data, and, the more sensors there are, the more work it has to do.

For example, inertial measurement units, like gyroscopes and accelerometers, simply provide a value at their output, either rotational or translational acceleration, respectively. They don't tell you when their value changes. That means that something – typically the application processor–has to poll constantly to confirm whether or not there has been a change. Even if the application processor has nothing else to do, this can prevent the application processor from going into sleep mode, increasing power consumption.

What's needed is a way to offload the sensor management tasks from the application processor, and an FPGA that can act as an application processor companion is an effective way to do this (Figure 3). You can create an auto-polling block and track the values, alerting the application processor when there has been a change that requires

some action. This effectively transforms the sensor reading from a “pull” to a “push,” allowing the application processor to ignore the sensor until interrupted.

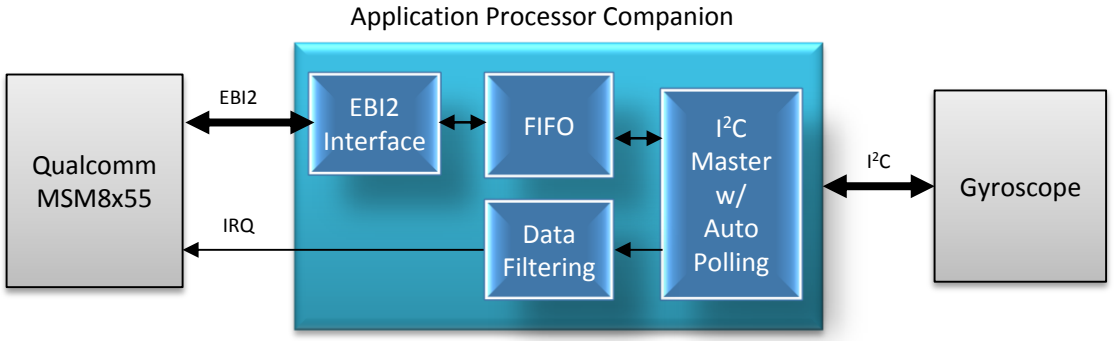

Figure 3

On the other end of the spectrum, some sensors, like touch sensors, already provide an interrupt signal so that the application processor doesn’t have to deal with it until there’s an event of interest. The problem is, when you get multiple fingers touching the sensor and swiping, the huge number of interrupts – potentially thousands of them – easily overwhelms the application processor. Here again, an application processor companion can manage the flood of interrupts, as shown in Figure 4, sorting through them to filter out the noise and deliver only those of interest to the application processor. This frees the application processor from meaningless distractions, allowing it to focus on more substantive tasks.

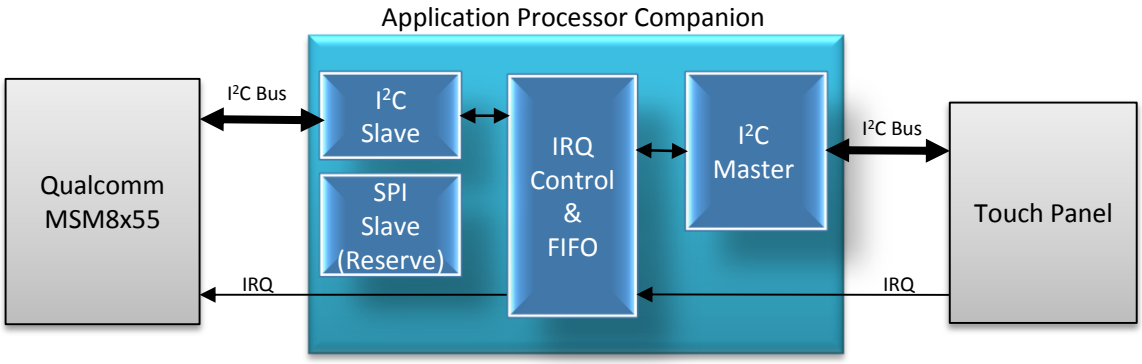

Figure 4

Note that you can also handle interface conversion at the same time. While the application processor shown above may accept an I^2C input, by including an SPI interface in the application processor companion, you can reuse the same design for a different application processor requiring an SPI interface, or perhaps for an application processor whose I^2C ports have already been committed to some other purpose.

These are examples of ways in which an FPGA – functioning as an application processor companion – can be paired with an application processor, designed three years ago, to allow you to take advantage of your new ideas today… not three years from now.

Lattice Semiconductor's iCE40™ mobileFPGA™ is a good example an FPGA specifically designed for the small footprint, low power and low cost requirements of smartphones. The FPGA makes it possible to keep up with the dizzying pace of innovation as smartphone life spans continue to shrink, while at the same time giving the mere conversation a back seat to all the other incredible things a "phone" can do.

###