WP198 (v1.0) June 30, 2003

CoolRunner-II CPLDs in Cell Phone Handsets/Terminals

By: Jesse Jenkins

Cell phone handsets (or “terminals,” as they’re called in Europe) are among the most dynamic products in the electronics market today. From their original analog roots, they have evolved into nearly pure digital devices with as much functionality as complex PDAs. Consumers who once evaluated handets based on their ability to make high-quality local calls now take call clarity as a given. Their choices instead rest on characteristics ranging from a handset’s "skin" color to its ability to support streaming video. Buyers, even those shopping for low-cost handsets, increasingly demand these kinds of features: "extras" are well on their way to becoming standards. This shift puts manufacturers in a bind as they try to balance low cost with the ever-increasing consumer insistence on new features. Should customers pay for these features outright, or should their monthly payments subsidize the handset cost?

Each country has adopted an individual economic model to resolve this question. Common to all of these models, however, is a need to financially cope with increasing numbers of new features. Our goal in this white paper is to show how using CoolRunner™-II CPLDs in handsets will make it easier for handset developers to make these future changes—as well as baseline handset capability—financially viable.

Quick Look at a Handset

Figure 1 shows a fairly straightforward block diagram of a digital-type handset.

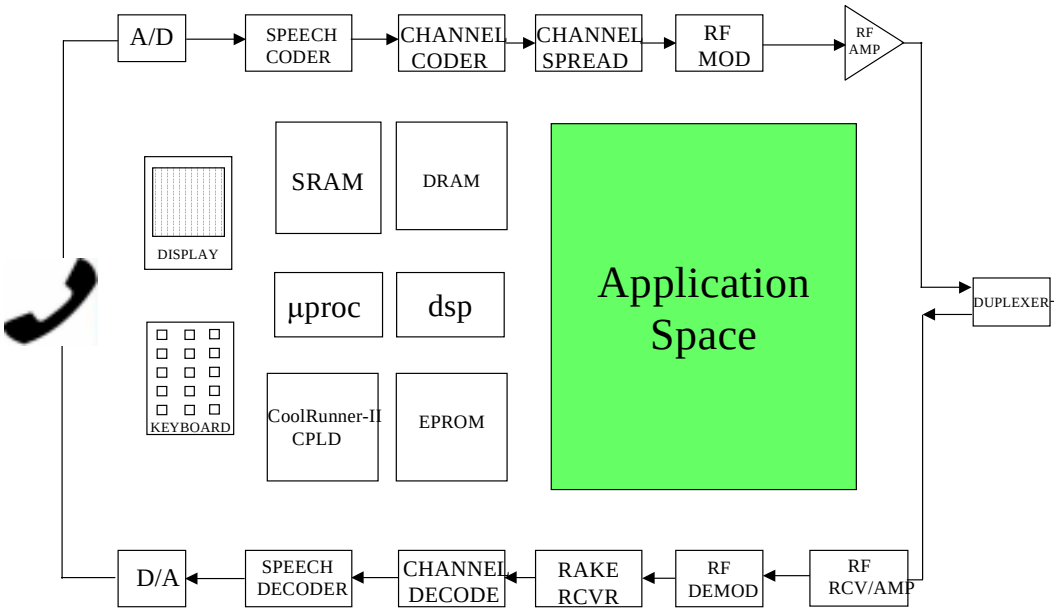

Figure 1: **Functional Components of a Digital Handset/Terminal**

In this diagram, we don't show specific connections between the microprocessor, the DSP, and the various boxes within the handset. Typically, they are all interconnected. In fact, some functions may be implemented by either processor. This saves board area at the expense of processor bandwidth. Let's assume the microprocessor handles baseline tasks like managing the display and the keyboard, but the DSP does channel coding / decoding and other signal processing tasks in the phone. The microprocessor handles dialing.

Let's simply look at sending and receiving audio. The microphone delivers audio signal to the A / D converter, creating a bit-stream. Fidelity and efficiency require redundancy elimination and compression. This is done in the speech coder (typically adaptive multirate). Forward Error Correction (FEC) occurs in the channel coder (Viterbi today, Turbo tomorrow). Coding adds back redundancy, but dropped bits can be recovered. CDMA channel spreading has been introduced in the U.S. and Korea, but others may use TDMA, SCDMA or WCDMA. The digital signal finally gets to the RF modulator and RF amplifier where it becomes one of several signal types (again, depending on the phone standard). At this point, it is analog. Today, these are in the gigahertz range and focused on octal phase shift keying. The duplexer is inserted to switch between receiving and sending at the antenna.

On the receiving end, the signal hits the antenna, slides through the duplexer and is amplified and demodulated, becoming digital. The Rake Receiver "despreads" the received bits and forwards the signal to the channel decoder, which reconstructs the possibly corrupted digital signal and passes it on to the speech decoder. The speech decoder corrects bit-dropout and should resemble the digital version of the original analog signal. This forwards to the D / A converter, which, when filtered and amplified (not shown), drives the earphone.

If the phone is going to support 3G capability, typically it must handle other standards (2G, 2.5G and GPRS) because different countries are at different levels of deployment. Why pay for a high-end phone and be able to only use it on the high end systems? Would you want to carry multiple phones, or should the manufacturer make sure your super handset just works wherever you are? Given that the latter is the case, your handset is basically three or four phones, depending on what communication link you are connecting to. It's a lot to ask. But 3G isn't about signaling, quality and compatibility, as much as it is about applications.

Applications

By now, you should have noticed the big box on the right hand side of Figure 1. That is reserved for advanced applications—things that go beyond making a call. Let's revise Figure 1 to set the stage for the kinds of things we will need to add applications. For starters, cell phone standards committees categorize applications according to parameters like guaranteed bit rate, variable / fixed delay, asymmetric / symmetric traffic and whether or not buffering is allowed. In terms of words we can relate to, those categories translate into conversational class, streaming class, interactive class and background class. These designations are based on the properties of those classes of applications. For instance, an ordinary conversation cannot tolerate long delays and echoes, as such things upset users. Lots of image dropout on a video data-stream also upsets users. So, by thinking through the needs of whole classes of applications, developers arrive at important quality factors. Listed below are some of potential cell phone applications; some may cross over more than one of the classes listed above.

1. MP3 music
2. Distance Learning
3. Online Retail
4. Travel booking
5. E-books
6. Distance Learning
7. Online Retail
8. Travel booking
9. M-commerce
10. Online trading
11. Mobile banking
12. Gambling
13. GamesInteractive toys
14. Video movie rental
15. Virtual radio station
16. Virtual TV station
17. Newspapers
18. Photo album
19. Video Conferencing
20. Field service
21. Business kiosks
22. Secure monitoring
23. telemedecine
24. Mobile clinics
25. Email
26. Secure monitoring
27. targetted advertising
28. Free trials
29. Remote metering
30. Remote control
31. Mobile speed cameras
32. Remote surveillance
33. Ticket purchases
34. Showtime information
35. Vending machines
36. Parking meters
37. Buying Crossword puzzles
38. Navigation
39. Finding ATM money machines
40. And many more . . .

Each application has its own particular needs, so it might be appropriate to consider a few of them individually. For instance, music via MP3 files could be done entirely in software, the same way that PCs can play files. If the processors are burdened with channel coding / decoding and possibly with decryption, it might be more feasible to include an MP3 decoder chip within the phone, or as an add-on. Video movie rental would probably have the files being downloaded via GPRS and linked through the Internet. Clearly, vendors renting the files would want to assure their MPEG-4 files weren't being delivered to pirates. Vendors would insist on encryption as well as compression. Adding these to the channel decoding exhausts today's processors and drains batteries . There would also be a need for buffering frames, to support dropout in the Internet protocol. Some applications might require JPEG, MPEG-4, MP3, encryption and channel coding. This suggests the need for additional offloading logic. Let's look at a commercial chipset for doing all of this, and then discuss offloading.

Cell Phone Chipsets

Figure 2 shows an Intel PXA 800F block diagram. Note that this multifaceted chipset is designed to integrate all the key functionality for the baseline, base-band operations, with a clear focus on using software to handle a broad range of other applications.

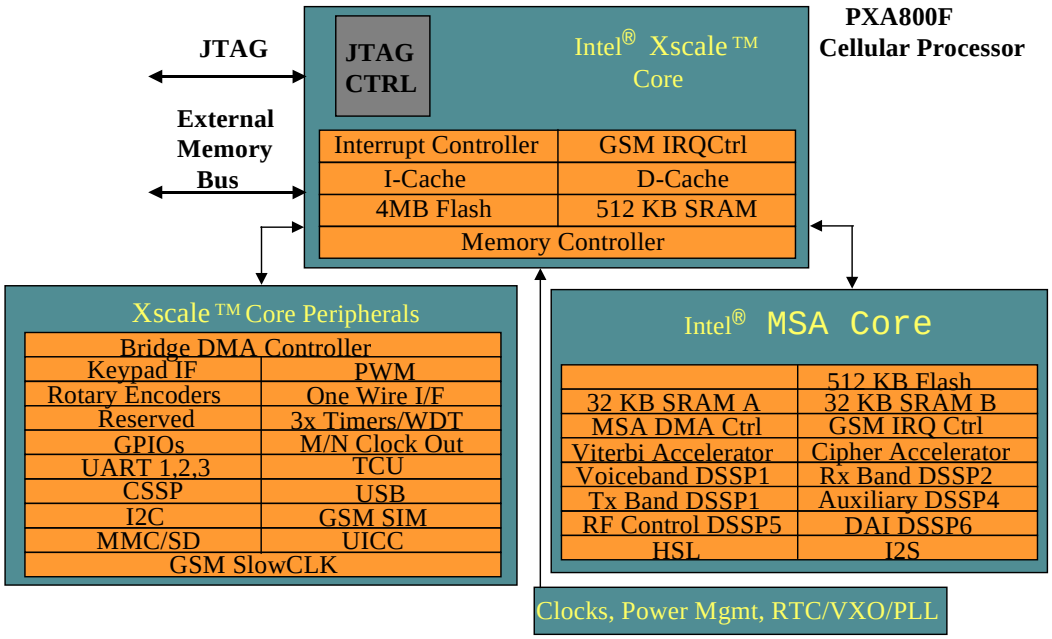

Figure 2: **Intel PXA 800F Cellular Chipset**

Manufacturers such as Motorola, Qualcomm and Texas Instruments offer their versions of chipsets that attempt do deliver the maximum capability in the minimum number of chips. Each is somewhat different in functional partitioning. The thing that is difficult for them to do is add enough flexible functionality to adapt to arbitrary future applications. We frequently seee pins referring to keyboard interfaces, or display interfaces, which is great, but they are all different. Some of the European UMTS terminals resemble pocket computers with wide screen displays and full computer keyboards as opposed to 12-16 keys on a simple cell phone. Adaptation logic is needed to use these chipsets with broader applications.

Picking the Right Mix

Figure 3 shows an application that's not on the list—Global Positioning Satellite (GPS) service. It can use the same antenna as the phone receiver / transmitter, but it will drag the satellite signals into the chip, collect the data and deliver the co-ordinates of your location to the DSP. Emergency police / ambulance calls need this.

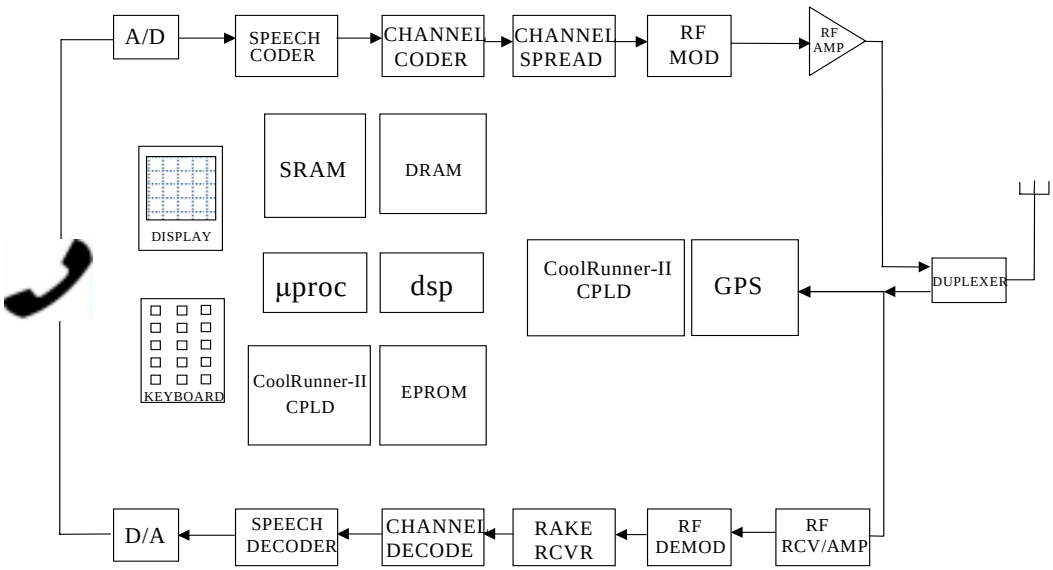

Figure 3: **GPS Added into a Phone**

In this case, it is impossible for the DSP or the microprocessor to "grow another radio chip." As long as the standard RF demodulator is being used, a second will be needed to support the GPS. The GPS will need to be interfaced to the processor buses to get data. The CoolRunner-II CPLD easily creates bus interfaces, interrupts and control operations for the processors, tying peripherals in.

Another example would be a camera. Adding a CMOS imaging chip requires interfacing it to the processor buses, along with additional memory to buffer images. Cameras come with either parallel or serial interfaces. Why be constrained to THEIR choice of peripherals in YOUR product?

For some time to come, applications will consist of adding in special application circuitry and interfacing it into the processor buses. Given that, how can a designer select the right application-specific circuits to add in, so that one ASIC could be built to interface them all? Basically, they can't. Below is an array of application modules that might be added into the phone:

- MP3
- MPEG-4
- Compact Flash +
- Video Camera
- GPS
- MMC / SDI
- More SRAM
- More DRAM
- More EPROM

But what about new applications that aren't in this group? There are applications yet to come that we can envision, but whose exact specifications we cannot wholly anticipate. Designing for their arrival is sometimes called "future proofing."

It's true that some applications are better served by microprocessor code dropped into on board EPROM, but that can only happen as long as the processor bandwidth is available for the application. If this can't be done, either more processors or additional silicon needs to be added. Either way, it will need interfacing into the phone bus network, and that will require programmable logic: very low power programmable logic.

MediPhone—A Speculative Example

To drive home some of these ideas, let's consider an idea for a product that probably doesn't exist today, but easily could in the near future. It will be marketed under the name "MediPhone" and will target segments of the population that require quick medical support. This would include the growing population of elderly citizens (frequently with enough money to buy these) as well as handicapped people needing close monitoring. See Figure 4 for an "artist" conception of this futuristic phone.

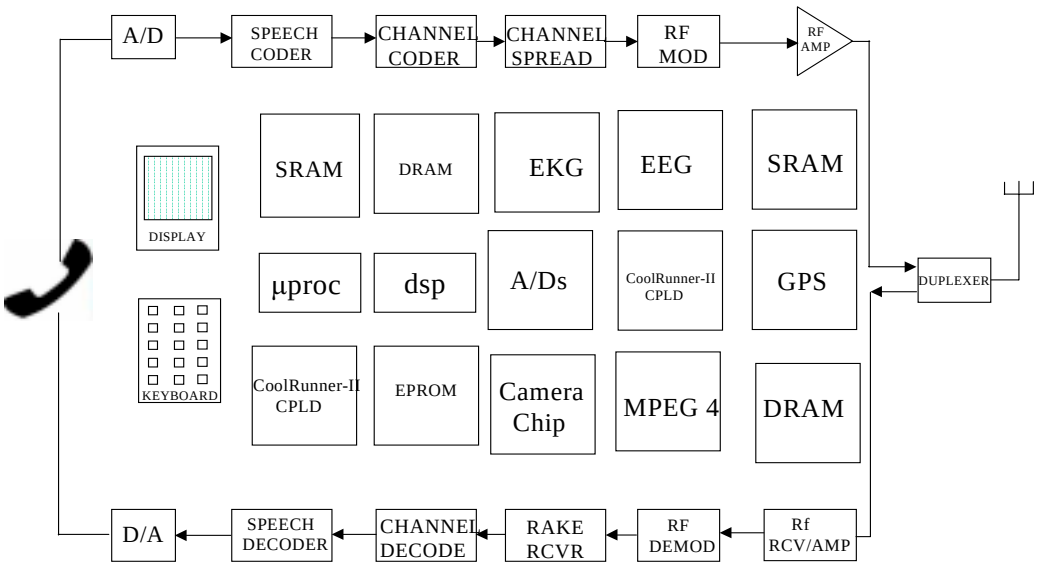

Figure 4: **MediPhone Block Diagram**

MediPhone works like this:

1. A heart attack (or other medical emergency occurs)
2. The victim or friend dials Emergency (911 in the U.S.)
3. Personnel receiving the call at a medical facility recognize the phone is "MediPhone" equipped and extract the geographic location of the emergency using GPS
4. The medic directs the friend to place the cell phone on the victim's face
5. A video camera scans the Iris for dilation to determine shock level
6. The friend is directed to attach small electrodes to the forehead / ear and chest of the victim, where pulse is taken and EEG / EKG measurements are driven into the Internet. Everything is attached to the phone.

7. Temperature is measured by physical contact of the forehead with the victim off the back of the phone
8. Microphone gain is adjusted to listen to the victim's breathing
9. It is determined that a heart attack has occurred and the friend is directed to elevate the feet and is given directions for CPR from the medic while an ambulance is dispatched
10. Ten minutes later, the ambulance team takes over and the MediPhone medic is released. For insurance reasons, all transaction data are encrypted, compressed and stored on CD for future reference.
11. The victim is saved and lives happily ever after . . .

In order to do this, MediPhone requires greater application functionality than discussed so far. Adding instrumentation amplifiers, A/Ds, transducers and support circuitry go beyond just having video and GPS as mentioned earlier. CoolRunner-II CPLDs would interface the various items needed to support MediPhone. The associated processor buses and memory interfaces areneeded to support the capture of the information taken. Other ideas quickly come to mind (WeatherPhone, ToxicGas Phone, etc.)

Power and Packaging are Key!

We have not yet mentioned power in this discussion, but it may be a key element in figuring out a solution, at least at the outset. Power management is a careful balance of dynamic operations within a cell phone. Do we add in specialty silicon, or do we just use a bigger EPROM and do it in software? As mentioned earlier, some choices are made for us. It would be ideal to just do it in the software, unless the bandwidth requirement on the processor(s) exceeds what could be done with a separate chip interfaced into the phone bus. Nonetheless, adding power consuming chips must be balanced against adding incremental code space, which also increases the power consumption. Whether you are interfacing more EPROM or additional ASSP silicon, CoolRunner-II is the ideal low power interface, with standby currents as low as 14 microamps. For power management, CoolRunner-II CPLDs are frequently used to enable/disable the power delivery to other chips within a system. So, if you are not using your camera or GPS capability, they can be automatically shut down to minimize power draw.

In addition to the unsurpassed power savings at high performance (clocking in the 300+ MHz range) provided by CoolRunner-II RealDigital technology, it is also critical to note that CoolRunner-II devices are offered in a wide range of very small packages. Table 1 shows part densities, available packages and other key features of CoolRunner-II CPLDs..

Table 1: **CoolRunner-II CPLD Family Parameters**

	XC2C32	XC2C64	XC2C128	XC2C256	XC2C384	XC2C512
Macrocells	32	64	128	256	384	512
Max I/O	33	64	100	184	240	270
T_{PD} (ns)	3.0	4.0	4.5	5.0	5.5	6.0
T_{SU} (ns)	1.7	2.0	2.1	2.2	2.3	2.4
T_{CO} (ns)	2.8	3.0	3.4	3.8	4.2	4.6
$F_{SYSTEM1}$ (MHz)	385	270	263	238	217	217

Looking Forward to 4G

The fourth generation cell phones are already being referred to as Software Defined Radio (SDR), because so many things can be more easily (in theory) changed with software. Processor speeds in the gigahertz range result in swift, agile flexibility. However, as mentioned earlier, software can't "grow" a camera. Software can't add in a bus interface to Compact Flash +. Software can't grow another radio channel. Only hardware can do some things, and when developers know neither when nor what the next new "thing" will be, only programmable logic can solve their development quandary. CoolRunner-II CPLDs provide the future proofing to help drive 4G cell phones.

Conclusion

We have focused, at a high level, on requirements of cell phones and on the capabilities of CoolRunner-II CPLDs for meeting those requirements. Given the current trend towards packing as much functionality as possible into handsets without increasing their power consumption, CoolRunner-II is an ideal choice for building phones that need to quickly adapt to new applications still on the horizon.

Product differentiation will ultimately result in winning designs. If developers want to cinch their winning positions, they must ensure that their designs have distinctive, useful applications that outdo those of their competitors. Using CoolRunner-II CPLDs to plan for future feature changes can contribute enormously to the speedy and cost-effective introduction of such applications.

References

1. *UMTS Networks, Architecture, Mobility and Services*, H. Kaaranen, A. Ahtiainen, L. Laitinen, S. Naghian, V. Niemi, 2001, John Wiley & Sons, Ltd.
2. *Services for UMTS (Creating Killer Applications in 3G)*, T. Ahonen, J. Barrett, editors, 2002, John Wiley & Sons, Ltd.
3. *3G Wireless Demystified*, L. Harte, R. Levine, R. Kikta, 2002, McGraw-Hill
4. "The Mobile Phone Meets the Internet," M. W. Oliphant, August 1999, *I.E.E.E. Spectrum*
5. "Evolving Cellular Handset Architectures but a Continuing, Insatiable Desire for DSP MIPs," M. L. McMahan, March 2000, Texas Instruments Application Report SPRA650

Further Reading

CoolRunner-II Design Kit

http://www.xilinx.com/products/cr2/design_kit.htm

Application Notes

http://www.xilinx.com/xapp/xapp375.pdf (Timing Model)
http://www.xilinx.com/xapp/xapp376.pdf (Logic Engine)
http://www.xilinx.com/xapp/xapp377.pdf (Low Power Design)
http://www.xilinx.com/xapp/xapp378.pdf (Advanced Features)
http://www.xilinx.com/xapp/xapp379.pdf (High Speed Design)
http://www.xilinx.com/xapp/xapp380.pdf (Cross Point Switch)
http://www.xilinx.com/xapp/xapp381.pdf (Demo Board)
http://www.xilinx.com/xapp/xapp382.pdf (I/O Characteristics)
http://www.xilinx.com/xapp/xapp383.pdf (Single Error Correction Double Error Detection)

http://www.xilinx.com/xapp/xapp384.pdf (DDR SDRAM Interface)
http://www.xilinx.com/xapp/xapp387.pdf (PicoBlaze Microcontroller)
http://www.xilinx.com/xapp/xapp388.pdf (On the Fly Reconfiguration)
http://www.xilinx.com/xapp/xapp389.pdf (Powering CoolRunner-II CPLDs)
http://www.xilinx.com/xapp/xapp393.pdf (8051 Microcontroller Interface)
http://www.xilinx.com/xapp/xapp394.pdf (Interfacing with Mobile SDRAM)

CoolRunner-II Data Sheets

http://direct.xilinx.com/bvdocs/publications/ds090.pdf (CoolRunner-II Family Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds091.pdf (XC2C32 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds092.pdf (XC2C64 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds093.pdf (XC2C128 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds094.pdf (XC2C256 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds095.pdf (XC2C384 Datasheet)
http://direct.xilinx.com/bvdocs/publications/ds096.pdf (XC2C512 Datasheet)

CoolRunner-II White Papers

http://www.xilinx.com/publications/products/cool2/wp_pdf/wp165.pdf (Chip Scale Packaging)
http://www.xilinx.com/publications/whitepapers/wp_pdf/wp170.pdf (Security)
http://www.xilinx.com/publications/whitepapers/wp_pdf/wp197.pdf (Cipher Stream Protocol)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/30/03	1.0	Initial Xilinx release.