

# Methods for Upgrading Reconfigurable Hardware Used in Mobile Phones

White Paper

Vishwas Mudagal  
Senior Software Engineer

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted, provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers, or to redistribute to lists, requires prior specific permission.  
GSPx 2006. October 30-November 2, 2006. Santa Clara, CA.  
Copyright 2006 Global Technology Conferences, Inc.

**Freescale Semiconductor, Inc.**

Rev #0  
11/2006



## OVERVIEW

Wireless technology is becoming increasingly complex, creating demand for more flexibility in hardware and software. The use of reconfigurable hardware such as field programmable gate arrays (FPGAs) in mobile phones can provide much needed flexibility to device manufacturers (OEMs) in developing new hardware features quickly and reducing time-to-market. A device can go to market quickly, and additional configuration of software and hardware can be done as soon as new features are ready.

This paper discusses methods for upgrading the reconfigurable hardware of mobile phones over the wireless network, and the uses and advantages of this technology. Primarily, the paper describes an architectural framework to download new hardware configuration data/firmware from a download server over the wireless network and reconfigure the reprogrammable hardware. This method enables OEMs to perform post-launch hardware upgrades on mobile phones. The upgrade could also include the software/firmware required to run the newly downloaded version/feature of the hardware.

The paper presents the example of interfacing an FPGA chip with the ARM® processor on Freescale's i300-30 dual core chip [8] used in mobile handsets. This will help clarify the ways in which FPGAs can be reconfigured to add new hardware features in more than one way, over the air or by other methods. The paper also describes a mechanism to store different hardware configurations on the phone memory. These hardware features can be installed by the user whenever required. Finally, the paper discusses ways to enhance this technology and its commercial advantages.

## CONTENTS

Introduction.....	2
Upgrade over the Wireless Network .....	2
Download Package Format .....	3
Download Process .....	4
Installation/ Upgrade Process .....	6
Other Upgrade Methods .....	7
Example .....	7
Hardware Profiles.....	8
Profiles Stored on Mobile Phones .....	8
Static Hardware Profiles.....	9
Dynamic Hardware Profiles .....	10
Dynamically Sharing Reconfigurable Hardware Resources.....	11
The Future of Mobile Phones .....	11
Conclusion .....	12
References .....	12

## Introduction

Upgrading the hardware of wireless communication devices such as mobile phones is one of the biggest challenges manufacturers face today. Traditionally, when a new mobile phone model with the latest technologies and features is launched by a manufacturer, competitors would be expected to launch a new phone with better technologies and features to match or outdo this latest phone as soon as possible to retain or gain market share. But, this trend may soon be forgotten.

Using reconfigurable hardware in mobile phones will give the much needed flexibility to launch new features almost any time, even after the phones have been deployed in the market. Currently available reconfigurable hardware such as field programmable gate arrays (FPGAs) meet the requirements of density and speed to allow a complete mobile platform design in a single system-on-a-programmable chip, i.e. from the main processor to all the desired peripheral functions [9]. Alternatively, hybrid platforms can be developed with a combination of FPGA and ASIC in order to keep the system (partly) programmable even after it comes out of fabrication. These platforms will allow us to statically reconfigure the finished product just prior to use. Dynamic reconfiguration can also be used to modify the behavior of an application system at several stages of the product's life.

Even though FPGAs today provide the flexibility and the capability to implement complex features, they are not suited for large scale/volume products due to their high cost and limitations with respect to power consumption, speed and size. But the hybrid (heterogeneous) platforms mentioned before can still provide a good mix of flexibility and implementation efficiency in the near future.

The advent of reconfigurable hardware mobile platforms will open avenues for new technologies and applications. Manufacturers can launch new hardware features with applications as soon as they are developed and tested. The use of reconfigurable hardware allows for updating the platform when new versions or new standards emerge. Reconfigurable hardware also introduces bug fixing capabilities for hardware systems. Moreover, end users can add new hardware features and even create hardware profiles in their phones. The dream of instantly changing the way mobile phones behave with just a press on the keypad can then be realized.

This paper describes a mechanism to store different hardware configurations on the phone memory. These hardware features can be installed by the user whenever required. This enables operators and handset manufacturers to offer a wide range of products and services to suit different budgets and customer classes.

Even more exciting will be a technology that will allow users to upgrade their hardware over the air. This paper discusses methods to upgrade the reconfigurable hardware of mobile phones over the wireless network. Primarily, this paper describes an architectural framework to download new hardware configuration data/firmware from a download server over the wireless network and reprogram the reconfigurable hardware. The framework will be described with an example of interfacing an FPGA chip with the ARM® processor on Freescale Semiconductor's i300-30 dual core chip used in mobile handsets.

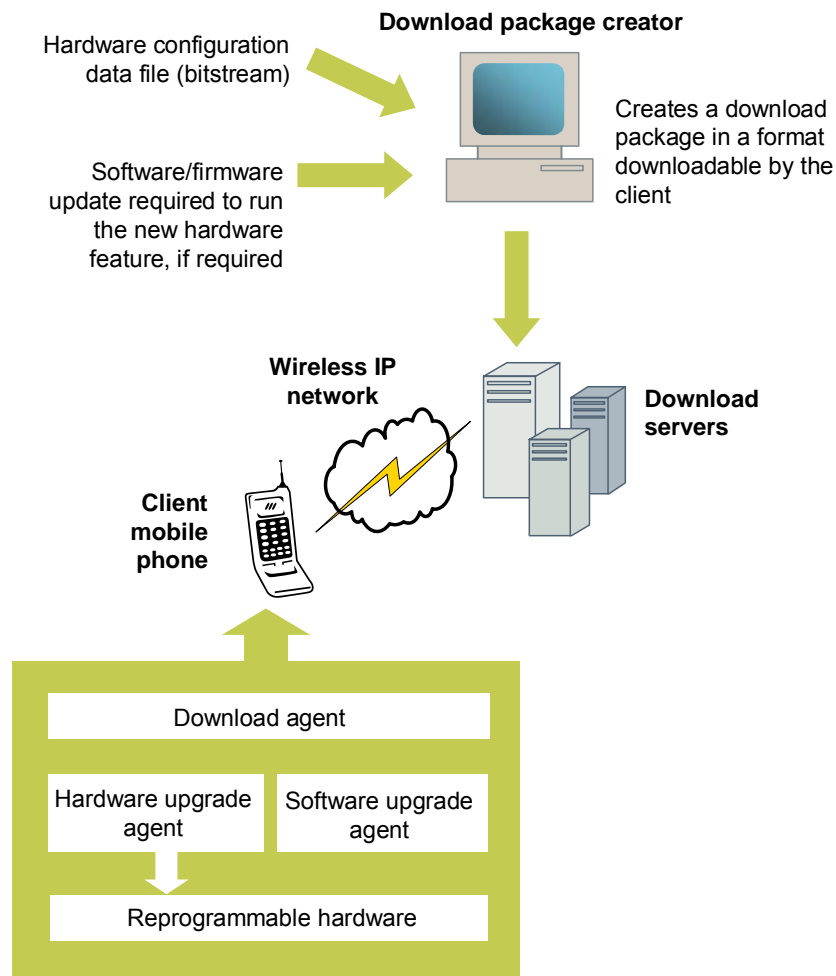
## Upgrade over the Wireless Network

This section describes an architectural framework for downloading new hardware configuration data/firmware from a download server over the wireless network and reconfiguring the reprogrammable hardware. This method can be used by manufacturers to perform post-launch hardware upgrades on mobile phones. The upgrade could also include the software/firmware required to run (or to use) the newly downloaded version/feature of the hardware.

The framework will elaborate on three main aspects:

1. Download package format
2. Download process
3. Upgrade or installation process on the client device

Figure 1 illustrates the framework graphically.



**Figure 1. System Overview: Hardware Upgrades Over the Air**

### **Download Package Format**

Manufacturers or service providers can provide hardware upgrades or new hardware features for the reconfigurable hardware used in a particular platform. This means that the downloaded data must be suitable to reconfigure the platform.

To create a download package for this service, the provider needs to provide the following:

- Configuration data file—used to program the reconfigurable hardware, usually called bitstream [2]
- Software/firmware upgrades—used to update the phone software image

The bitstream is digital information used to configure reconfigurable hardware. The software upgrades refers to any changes that the current phone software version needs to undergo in order to utilize or run the new hardware features installed by using the new bitstream. Software upgrades may not be required if the new hardware upgrade doesn't require any software changes in order to run the feature.

The bitstream files can be created in a number of formats and also depend on the type of reconfigurable hardware used—for example, the bit (.BIT) files for FPGAs. The format may even differ among hardware from different vendors in the market.

A suitable format can be decided on to combine these two inputs and create a download package, which will be downloaded by a wireless client from a download server. The format should be agreed upon by the download server and the client. The client should be able to extract the necessary data from the download package and distinguish between hardware and software upgrades.

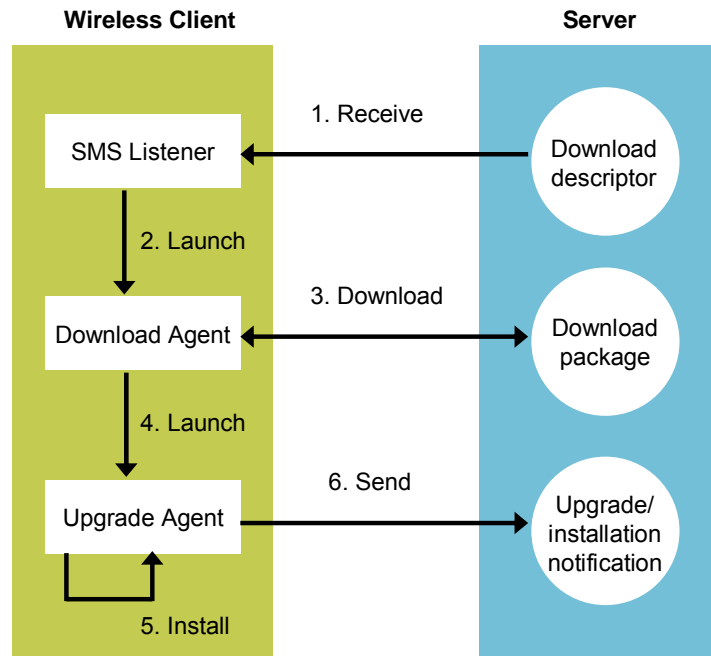
There are several ways in which the information can be sent to the client with minimal data transfer. For instance, if the hardware and software upgrades are minimal, only the delta of the previous version and the new version can be recorded in the download package. A mechanism can be implemented in the client to use this delta and make the necessary upgrades on the client's hardware and software features.

On the other hand, if completely new hardware features are made available for download, the entire bitstream can be recorded in the download package.

### **Download Process**

For the download process of the upgrade package, the client and the server architecture can be followed. The OMA's "Generic Content Download Over The Air" specifications [1] describe an excellent client-server architecture and the related protocols.

As shown in Figure 1, the client will implement a Download Agent and an Upgrade Agent. The Upgrade Agent will implement a Hardware Upgrade Agent and an optional Software or Firmware Upgrade Agent. The client can implement the Download Agent that supports W-HTTP, WSP or any other similar protocol that can provide a confirmed download. This also can be coupled with other aspects such as security and billing mechanisms. A generic approach to the steps involved in the download process is shown in Figure 2.



**Figure 2. Download Process between the Server and the Client**

The Download Agent can support the reception of the Download Descriptor using a messaging protocol such as SMS, MMS or email. The server can send a message to the client to inform the user about the download package. The package can be a hardware upgrade from the OEM or an option of adding new features to the client device.

The Download Agent will do a capability check of the download package depending on the information given in the descriptor. If the package is not suitable for the client device, it will be rejected.

Once the user approves the download, client-server negotiation can proceed for authentication, security and client-device information such as reconfigurable hardware details. If authentication is successful, the client will download the package that is identified by the URI as specified in the descriptor. The package can be retrieved by using HTTP or HTTPS schemes.

The Download Agent can typically copy the downloaded package in the nonvolatile memory (NVM) of the client device. The user must be given the status of the progress of the download on the screen.

The download may fail for of a variety of reasons, including power cuts, memory-related reasons, attribute mismatch or user cancellation. In this case, the client must inform the server and the user about the reason for the failure. If the server doesn't get the notification, then it can consider the download process as a failure.

Once the download is complete, the user should be informed that the download is successful and that the package can be installed. If the user approves the installation, the Download Agent can launch the Upgrade Agent.

Note that security can be a major area of concern in this process, but a discussion on security issues is out of the scope of this paper. Care should be taken to address this concern.

## Installation/ Upgrade Process

The Upgrade Agent will extract the necessary information from the download package, which will be stored in the NVM of the device by the Download Agent. It then basically has to distinguish between the hardware configuration data and software upgrade data.

The Upgrade Agent is comprised of two sub-entities: a Hardware Upgrade Agent and a Software Upgrade Agent.

The Hardware Upgrade Agent will convert the hardware configuration data into the bitstream used by the reconfigurable hardware of the client device. Let us consider an FPGA chip to be attached to the client device. In this case, the bitstream may be the bit (.BIT) file. The FPGA can be programmed in a number of ways—using a hardware controller, a boot-PROM or a similar method [2]. If an on-board controller is used to program the FPGA at the boot process, it will read the .BIT file from the NVM and program the FGPA. Figure 3 illustrates this graphically.

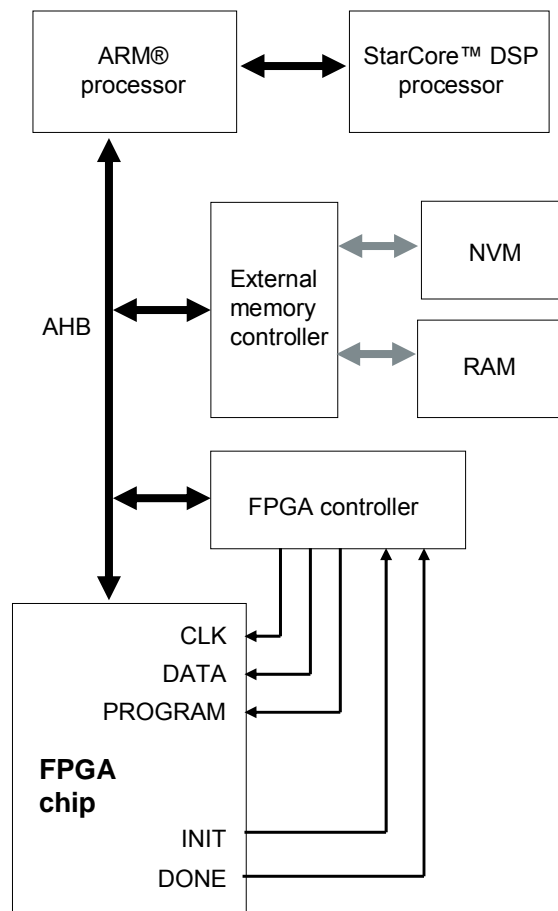


Figure 3. An FPGA Chip Attached to the ARM Processor of the Freescale i.300-30

The Hardware Upgrade Agent will replace the old .BIT file with the new .BIT file in the NVM. Other fault-tolerant mechanisms can be added in this process to ensure that the copying is successful. In the next boot process, the FPGA will be configured with the new .BIT file and the downloaded hardware feature will be ready for use. But if the changes are minimal and the existing client software can handle these

changes, then the software upgrade may not be needed. For example, the firmware to run the hardware feature remains the same. Hence, the reconfiguration of the FGPA can be done without a reboot. The software tasks handling the FPGA can be suspended during the reconfiguration process, and then be activated after the process is completed.

The method can also change the behavior of the FGPA in such a way that a software upgrade may be needed to run the new FGPA behavior. In such a case, after the Hardware Upgrade Agent has completed its job, the Upgrade Agent will convert the software upgrade data from the download package into a format used by the client device to execute the software (for example, .SHX file format). The Upgrade Agent will store this new software image into the NVM and ask the user to re-boot the system.

The download package can also contain information of only the delta (d) between the new (say v2) and the old (say v1) versions of the phone software image. In this case, the Upgrade Agent can extract the necessary data required to upgrade the old software image.

To upgrade the phone software image, a reboot may be required because write access to the phone software image in the NVM may not be possible when the phone is active. Hence in the next boot process, the Software Upgrade Agent needs to be launched. This entity can be loaded by the boot loader and should be designed to execute without the support of the operating system. Usually, the phone software image is typically stored in the on-board NVM and is copied onto the RAM to execute it. Hence, Software Upgrade Agent needs to take the new image and copy that to the NVM address space in which the phone software image is stored. The process can be made fault tolerant to events such as power loss and memory inefficiencies with additional mechanisms. After the image is changed successfully, the Software Upgrade Agent can reboot the device. In the next boot process, the boot loader can launch the normal phone image, but now the phone image will have the required upgrades. The Upgrade Agent can then inform the server that the installation is complete.

The user can now use the newly downloaded hardware feature. The same process can be repeated if the user wants to download another upgrade that the manufacturer or the service provider provides. Further, these downloads can be stored on the phone memory as hardware profiles, as described later in the paper, and can be activated whenever required.

## Other Upgrade Methods

The reconfiguration of the reprogrammable hardware can be done using several other methods. Other wireless technologies such as Bluetooth®, WLAN, Wi-Fi™ or UWB can be used to download hardware configuration data and software upgrades to the mobile phone. The Upgrade Agent mentioned in the previous section can then be used to reconfigure the hardware and upgrade the software.

Alternatively, the mobile phone can be connected to a host machine such as a computer, using USB or UART. A host-based tool can then be used to download the hardware configuration data and software upgrades to the mobile phone. The Upgrade Agent mentioned in the previous section can then be used to reconfigure the hardware and upgrade the software.

### Example

A hybrid platform of partly reconfigurable hardware system-on-chip can be designed according to the needs of the applications, the protocols and the standards the platform supports. An analysis has to be done as to which functionalities will be implemented in software, hardware and reconfigurable hardware. In [3] and [7], the authors have extensively investigated several design and modeling methodologies to achieve a fine balance in the functionalities distributed among software, hardware and reconfigurable hardware. These methodologies are not covered in this paper; for more information please refer to [3], [5] and [7].

As described by the authors of [7], coupling with a host microprocessor is one of the reconfigurable architectures. In a closely coupled system, reconfigurable units are placed on the data path of the processor, acting as execution units. For the sake of simplicity, an example of coupling or interfacing an FPGA chip with a microprocessor is considered.

Freescale's i300-30 [8] is a dual processor platform used in mobile handsets. It has an ARM11™ application processor and a StarCore™ DSP processor, with different dedicated hardware peripherals. This platform is a classic ASIC SoC. An FPGA chip is interfaced to the AHB, the main bus, of the ARM processor as shown in Figure 3. A specially designed controller is introduced in the platform to configure the FPGA. The controller can be designed in a number of ways [2], but basically it loads the bitstream into the FPGA in serial or parallel mode. The controller is attached to the AHB and is controlled by the phone software running on the ARM processor. The controller can be activated during the boot process or any other time the configuration of the FPGA is required. The controller can be instructed to fetch the desired bitstream from a specified location of the NVM or the RAM.

Consider that the FPGA is configured to behave as a hardware accelerator for a special-purpose application such as games, video, graphics, etc. during the boot process and that the current phone software is sufficient to run this hardware accelerator.

Now, the user downloads a new hardware accelerator with advanced features from the service provider, using the process described previously. The new hardware accelerator requires new software that will also be sent with the download package.

The Download Agent strips the downloaded package and extracts the bitstream for the FPGA configuration and the software upgrade information from it. The Hardware Upgrade Agent is then launched, which replaces the old bitstream with the new one in the NVM at a specified location from which the FPGA controller fetches the bitstream to configure the FPGA. In the next boot process, the Software Upgrade Agent is launched that upgrades the phone software image with the information received. The user can now use the downloaded hardware feature directly or download various other applications from the service provider to run with the new hardware accelerator. This process is similar to the static reconfiguration wherein all the tasks are halted for the upgrade.

Dynamic reconfiguration and partial reconfiguration [6] can also be achieved. By implementing hardware profiles, as described in the next section, these features can be used effectively.

## Hardware Profiles

### Profiles Stored on Mobile Phones

Using reconfigurable hardware in mobile platforms will facilitate storing different hardware configuration files in a mobile phone. A hardware configuration file, together with the software/firmware and applications to run the feature, may be called a hardware profile.

Whenever a user chooses a hardware profile, the Upgrade Agent can configure the FPGA with the corresponding bitstream. The challenge here is to have flexible software to run different profiles. Basically, the need is to have the firmware required to control the feature configured in the FPGA and applications to run that firmware. For example, if an image processing hardware accelerator program is loaded into FPGA, a software driver to run that particular hardware accelerator is needed. The FPGA can be controlled by the software driver as a memory mapped device attached to the main bus of the processor. This software driver in turn is used by applications such as a camera. Hence, the hardware configuration file (bitstream), the firmware (software drivers) and the applications together constitute a hardware profile.

Consequently, a framework is needed to change the drivers and applications depending on the profile the user chooses. This framework can be designed in a number of ways, but basically the aim is to

enable the required hardware and software features and disable the rest. This framework will get more complicated if more profiles are made available for the user and if more FPGA chips are attached to the phone. If the mobile phone has multiple FPGA chips attached to its main processor, multiple profiles can be chosen.

Further, the framework can interact with the Download Agent and the Upgrade Agent to store the packages (or profiles) that the user downloads over the wireless network. The framework can have a mechanism to store all the profiles in a database in the on-board NVM. Whenever the user chooses a profile, the Upgrade Agent can upgrade the FPGA with the corresponding bitstream and the framework can then dynamically enable the software required for the profile.

### Static Hardware Profiles

Hardware profiles that require the system to halt for the reconfiguration of reprogrammable hardware can be called static hardware profiles. This scenario arises when the current phone software version cannot support all the hardware profiles. For instance, when the user downloads Hardware Profile 1 that cannot be supported by the existing phone software version, the software upgrade will also be sent in the download package. To activate this profile, the system has to be rebooted in order to upgrade the hardware and the software as described previously. The same approach has to be considered even when the functionality implemented in the reconfigurable hardware is critical for the system. In this case, static reconfiguration becomes inevitable.

Figure 4 illustrates the procedure of switching between static hardware profiles. The blue lines indicate that the corresponding bitstreams are loaded into the reconfigurable hardware. The red lines indicate that the software image file stored in the NVM is actually changed when the profiles change.

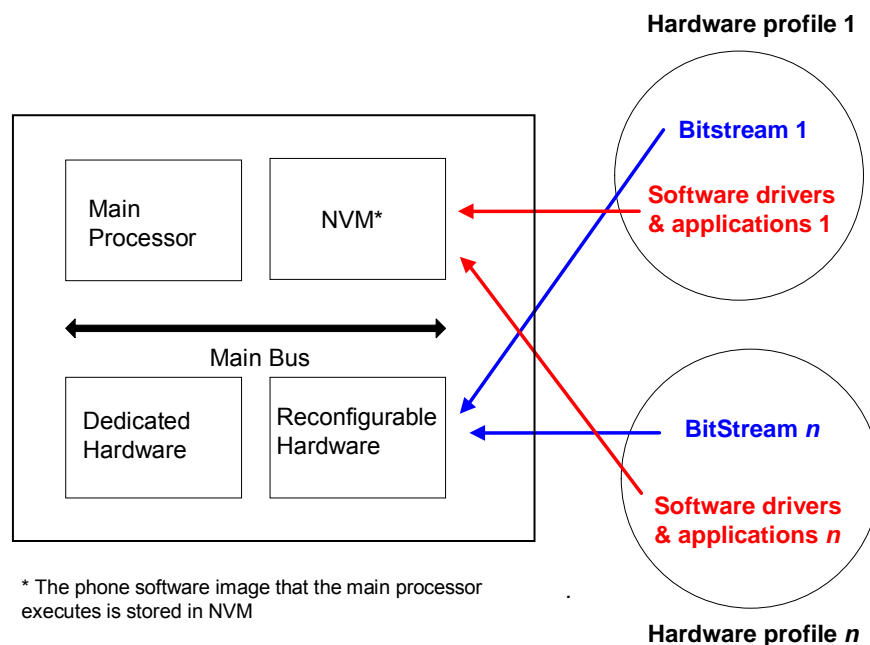


Figure 4. Static Hardware Profiles

## Dynamic Hardware Profiles

Hardware profiles that can be dynamically configured can be called dynamic hardware profiles. This means that the system need not be halted for switching between hardware profiles. This scenario arises when the current phone software version can support all the hardware profiles. Dynamic hardware profiles require advanced system designs that offer very high software flexibility. Manufacturers need to design a framework that can dynamically enable and disable software features. The framework should also have the capability to store the hardware profiles downloaded over the air.

Figure 5 illustrates the procedure of switching between dynamic hardware profiles. The blue lines indicate that the corresponding bitstreams are loaded into the reconfigurable hardware. Here the phone software image need not be changed, because the same version has the required software capabilities to support all the hardware profiles. On the other hand, the framework needs to enable the software required to run the chosen profile, as shown.

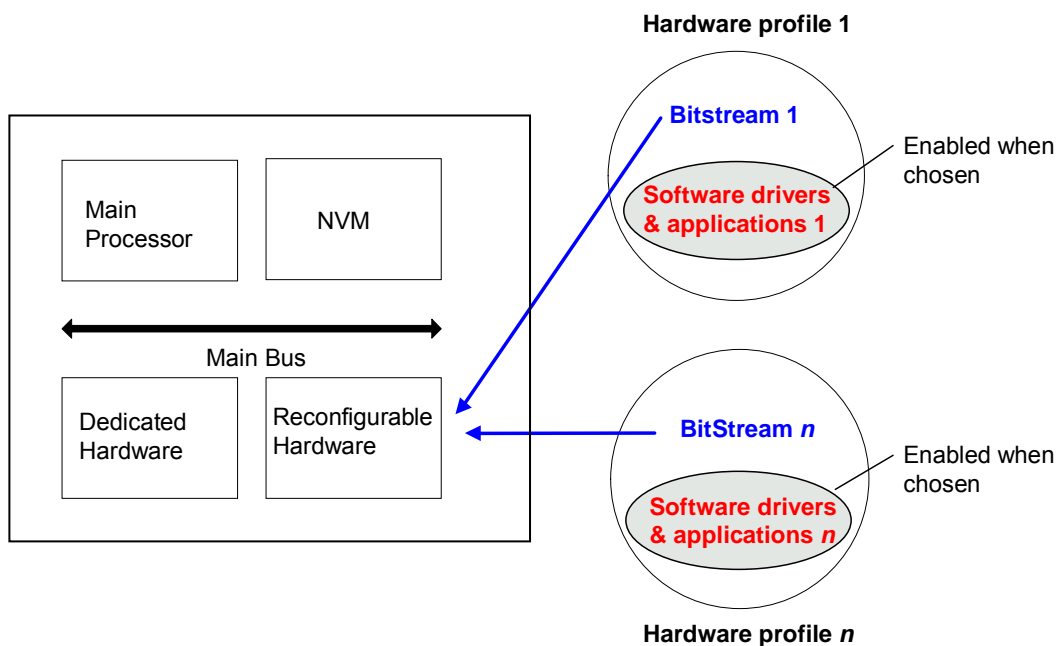


Figure 5. Dynamic Hardware Profiles

This framework will allow the manufacturers to provide in-built hardware profiles with the new mobile handsets to enhance the customer experience with greater flexibility and advanced features. This is analogous to the software profiles currently offered by mobile handsets. Providing the built-in hardware profiles is comparatively easy because manufacturers can design and test these profiles before the product is launched in the market. However, downloading dynamic profiles over the air can be complex because upgrading the software becomes inevitable. One solution to this problem is use static profiles as described before, but that process needs the entire system to halt, which is undesirable. Another solution could be to download several dynamic profiles in one download package and use the static reconfiguration process, which will eliminate the need to use static profiles.

Dynamic profiles can also leverage the latest advances in the field of reconfigurable hardware. For instance, partial FPGA reconfiguration, which is the ability to reconfigure select areas of an FPGA at any time [6], can be used to add multiple applications to a single FPGA chip. This can be done while the FPGA is operational and active (known as active partial reconfiguration). The advantage partial

reconfiguration provides is that while a portion of the design is being reconfigured, the rest of the system can continue to operate. There is no loss of performance or functionality with unaffected portions of a design—no down time. This also facilitates sharing the reconfigurable hardware resource among applications, thus reducing device count, power consumption, board size and costs.

### **Dynamically Sharing Reconfigurable Hardware Resources**

Even though dynamic hardware profiles may provide flexibility, they may not be able to implement system-critical functions. System-critical functions will render the entire system paralyzed when they are disabled. Dynamic profiles can support critical functions only if the system can shift the execution of certain features from reconfigurable hardware to software (i.e. execute in the main processor) and vice-versa, during runtime. Features can be shifted from reconfigurable hardware to software without any down time. The advances in system-level design methodologies and language-oriented design flows can help achieve this flexibility. An example of language-oriented design flows is SystemC [4].

Moreover, applications and features that require high computational resources can use reconfigurable hardware to improve speed and efficiency. Because all features can't run at optimal levels on the same hardware, these features can use reconfigurable hardware and configure it at runtime to achieve maximum performance depending on the tasks they are handling. For example, Application 1 is currently executing in the FPGA. The user starts Application 2, which has a higher priority and higher resource requirements than Application 1. In this scenario, Application 1 can be shifted to software execution. The FPGA can be reconfigured for Application 2 and it starts using the FPGA to achieve the required performance.

## **The Future of Mobile Phones**

The mobile phones of the future will need greater flexibility and more complex features. Completely replacing the ASIC with reconfigurable hardware cannot happen overnight. But eventually, as reconfigurable hardware technology improves and costs come down, it will make economic sense to replace ASIC with reconfigurable hardware to the maximum extent possible. This technology, together with advances in wireless technology, will help both manufacturers and consumers.

In the electronics business, companies have lost market share and profits to rivals just because they couldn't launch new versions of their products in time. This happens in spite of having the complete technology ready. But this scenario will seldom arise when reconfigurable hardware is used in products such as mobile phones. Manufacturers need not wait for the launch of the next product. They can just upgrade the reconfigurable hardware with the latest features over the air for all users, if needed. The advantage for the user is that the newest versions are always available and that mobile handsets do not become obsolete quickly. The advantage for manufacturers is that they can profit by launching the latest versions as quickly as possible to match rival products, helping them retain or gain market share.

Reconfigurable hardware also introduces bug fixing capabilities for hardware systems. With the help of advances in wireless technology, all these services can be done over the air. Not only can new features be added to mobile handsets over the air, but handsets also can be debugged over the air to fix hardware and software bugs. This technology will open up a huge market for "on-demand" hardware feature downloads over the air. Of course, making this a reality will require investments in research, time and deployment. But it will be worth the investment.

The experience end users will gain through this technology will be nothing short of magical. The user can add new hardware features and even create hardware profiles in a phone, with the ability to change the behavior of the phone on the fly. With the ability to partially reconfigure and to dynamically share reconfigurable hardware, hardware profiles can even be customized to add required features to the available reconfigurable hardware resources. More generic mobile handsets can then be sold in the market, giving users the choice to add the features that they desire or require the most. Of course, users can change the features over the air when they want to.

## Conclusion

This paper has presented an approach for upgrading the reconfigurable hardware used in mobile phones over the wireless network. A download package containing the hardware configuration data and the software upgrade data is transferred over the wireless network to the mobile handset client from the download server. A general approach for the download process is described. Various methods of upgrading the reconfigurable hardware and the software of the mobile handset are described.

Partly reconfigurable hardware platforms are identified as sufficient to provide enough flexibility and implementation efficiency. The paper presents a simplified example of an FPGA chip attached to the Freescale i300-30 to explain the upgrading process over the air.

The configuration bitstream files together with the software/firmware and applications to run the reconfigurable hardware feature may be called a hardware profile. Various types of hardware profiles are presented and areas of improvement are mentioned.

## References

- [1] "Generic Content Download Over The Air," Open Mobile Alliance (OMA-Download-OTA-V1\_0-20040625-A), <http://www.openmobilealliance.org/>
- [2] "The Low-Cost, Efficient Serial Configuration of Spartan FPGAs," Application Notes, <http://www.xilinx.com/>
- [3] Pelkonen A, Masselos K, Čupák M "System-Level Modeling of Dynamically Reconfigurable Hardware with SystemC", Proceedings of 10th Reconfigurable Architectures Workshop, March 2003
- [4] <http://www.systemc.org/>
- [5] Rissa T., Vasilko M., Niittylahti J., "System-Level Modeling and Implementation Technique for Run-Time Reconfigurable Systems", Proc. of FCCM, April 2002
- [6] "Benefits of Partial Reconfiguration," [http://www.xilinx.com/publications/xcellonline/xcell\\_55/xc\\_reconfig55.htm](http://www.xilinx.com/publications/xcellonline/xcell_55/xc_reconfig55.htm)
- [7] Compton K., Hauck S., "Reconfigurable Computing: A Survey of Systems and Software"
- [8] <http://www.freescale.com/>
- [9] K. Masselos, A. Pelkonen, M. Cupak, "Realization of Wireless Multimedia Communication Systems on Reconfigurable Platforms"

**How to Reach Us:**

**Home Page:**

[www.freescale.com](http://www.freescale.com)

**e-mail:**

[support@freescale.com](mailto:support@freescale.com)

**USA/Europe or Locations Not Listed:**

Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
1-800-521-6274  
480-768-2130  
[support@freescale.com](mailto:support@freescale.com)

**Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[support@freescale.com](mailto:support@freescale.com)

**Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064, Japan  
0120 191014  
+81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

**Asia/Pacific:**

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate,  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

**For Literature Requests Only:**

Freescale Semiconductor  
Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447  
303-675-2140  
Fax: 303-675-2150  
[LDCForFreescaleSemiconductor@hibbertgroup.com](mailto:LDCForFreescaleSemiconductor@hibbertgroup.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright license granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.



Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM is the registered trademark of ARM Limited. ARM11 is the trademarks of ARM Limited.

© Freescale Semiconductor, Inc. 2006.

