Shanghai Hometown Microsystems Inc

System Design Frontier

Monthly Online Technical Journal Volume 1, Number 5, July 2004

Enabling of Your Local Globalization and Global Localization

Content

China Electronics Fair Since1964

64th China Electronics Fair (CEF Shanghai 2004)

2004年(第64届)全国电子产品展览会

Venue: Shanghai New International Expo Center,China

Date: Nov.15-18, 2004

CEF is a most authoritative, comprehensive and professional electronics fair in China, which is an electronic fair supported fully by China Ministry of Information Industry and Ministry of Commerce. CEF is also the fair with a long history, large scale, and deep influence of its kind in China. CEF Shanghai 2004 will reach 46,000 sqm in exhibition area.

Now for FREE visitor ticket:
www.chinaelec.com.cn

Together with 2004 Asia Electronics Exhibition in Shanghai (AEES 2004)
www.aeesshow.com

Hall 1: AEES 2004 (For consumer elctronics), Digital household electronic device

Hall 2: CEMAS 2004(For electronic facilities), Automobile electronics, Power & Supply

Hall 6: Testing and measurement instrument, Optical products, Sensors

Hall 7: Comprehensive electronic components, Military electronics

Sponsor:
China Electronic Appliance Corporation (CEAC)

Organizer:
China Electronic Exhibition & Information communication Co., Ltd (CEEIC)

Contact:
49, Fuxing Rd., Beijing, 100036, China
Tel: +86-10-68207732/33/34 ext. 16/22
Fax: +86-10-68132578/68189519
E-mail: nef@ceac.com.cn
www.chinaelec.com.cn

Co-organizers:
Provincial branches of CEAC
Hong Kong Trade Development Council (HKTDC)
Taiwan Electrical And Electronic Manufacturers' Association (TEEMA)
International PR, Inc. Korea
American Business Exhibition Co., Ltd

Reply Form (Fax to: 010-68132578):

I am interested in ____ exhibit, ____ visit CEF Shanghai 2004, please sent me more information.

Name:	Company name:
Position:	URL:
Addr:	email:
Tel:	Fax:
Main product:	

中电会展 电子信息业会展专家

上规模上档次　　国家规划布局内的重点软件企业　　精英灿烂！
展风采　　软件最大规模前 100 家企业　　强劲登场！

“中国软件 20 年”评选活动

INT’L SOFT CHINA 2004
2004 第八届中国国际软件博览会

主题：中国软件产业国际化与企业做强做大

开幕式/主论坛/分论坛
2004 年 7 月 8 日－7 月 10 日
北京，中国国际展览中心 1 号馆、综合楼

支持单位：中华人民共和国国家发展与改革委员会
中华人民共和国科学技术部
华人民共和国商务部
华人民共和国国务院信息化办公室
各省市、自治区信息产业主管部门
主办单位：中华人民共和国信息产业部
承办单位：中国软件行业协会
中国计算机报社
中国国际贸易中心
大会网站：赛迪网（www.ccidnet.com）

欢迎各界报名参选、参会、参展！

评选报名热线：010－88559723/9995
论坛联系方法：
联系人：邱燕娜　郝杰
电话/传真：010－88559723/9995/9809/9867，88559887（F）
地址：北京海淀区紫竹院路 66 号赛迪大厦 18 层中
国计算机报（邮编：100044）
电子邮箱：qiuyanna@ciw.com.cn
grace@ciw.com.cn

参展联系方法：
联系人：朱芸 顾长江 李璐
电话/传真：010－62186579（F），62143871(F), 51527167，51527159
地址：北京海淀区学院南路 55 号中软大厦 A401 室（邮编：100081）
网站：www.csia.org.cn　www.ccidnet.com (欢迎网上注册参展参观！)
电子邮箱：csia@css.com.cn

论坛内容

第一部分：开幕式
第二部分：颁奖仪式
奖项设置：
中国软件 20 年明星企业——拟授予在软件领域辛苦耕耘了近二十年，在产业的发展过程中不断成长壮大，起到了典范作用的软件企业
中国软件 20 年特别成就奖——拟授予在软件产业的发展过程中付出了巨大心血，并在软件领域取得突出成就的软件专家、学者及知名人士
中国软件 20 年杰出贡献奖——拟授予为推动软件产业的发展做出长期贡献并取得骄人成绩的软件企业家
中国软件 20 年最具应用价值的优秀软件产品——拟授予在长达近二十年的应用中经得住用户考验、发挥了巨大应用价值的软件产品
评选流程：
第一阶段：通过各地信息产业主管部门、各地软件行业协会推荐候选者，或由候选者主动报名，专家推荐
第二阶段：邀请软件领域知名专家、软件行业协会资深人士、IT 媒体资深人士及产业相关领导等组成评委会进行审议
第三阶段：采用群众投票初评，评委会复评最后确定获奖者
第四阶段：于软博会开幕式进行颁奖
第三部分：主论坛
主题：
软件政策与产业发展论坛（18 号文件与 WTO、行政许可证与 18 号文件、政府采购）
资本与软件企业成长性论坛
软件国际化与服务外包论坛
软件学院与人才培养论坛
技术进步与 IT 应用论坛
版权保护与国际形象论坛
开放源码论坛
高峰对话：中国软件二十年
对话嘉宾：“中国软件 20 年杰出贡献奖”及“中国软件 20 年特别成就奖”获得者代表
第四部分
一、　嵌入式专场
主题：嵌入式产业新机遇（汽车电子、消费电子、移动增值、工业控制、医疗电子、机床电子、银行终端等）
二、　中间件专场
主题：软件产业的发展后动力
三、　游戏专场
主题：做大做强国产游戏
四、　软件与传统产业专场
主题：从传统产业中诞生的软件企业
第五部分：“中国软件二十年”纪念展（软件墙）
在主论坛展示区内，以“产业篇章”的形式，总结中国软件产业二十年来的发展：产业篇、人物篇、成果篇

EDITORIAL

Never Too Late

Editor In Chief

I had my DAC trip at San Diego in early June, it shocked me in someway: due to the industry wide of cost reduction, I saw fewer attendees this year despite the fact that the industry definitely seems to be turning up, on the other hand, lots of startups showed up for the first time there demonstrating their key technologies. In late June, we set up a booth at ELECOM 2004, a leading trade show for telecommunication industry in China. I experienced the similar phenomena as I saw at DAC, one big difference between ELECOM is that, even though there are lots of startups in China, not too many startups showed up at the trade show, and multi-nationals along with local heavy-weights dominate the seen, and it makes startups hard to breathe. I was impressed by local heavyweights, on their deep pockets as well as their technology advance.

Honestly I am quite optimistic on the road ahead. Why? Both in HW engineering and SW engineering, there are so many existing problems yet to be solved or to be proved as unsolvable, and more and more new problems are emerging as well. For example, the Design for Manufacuribity problem is new, and is different as Yield Optimization problem a decade ago, and here Danny Rittman has a good paper in this issue; the system-level synthesis problem, proposed around the mid-90s, has no perfect solution so far, even with quite a few system-level specification languages being proposed, ranging from C to C++, Java, MatLab, SystemC, and most recently SystemVerilog, here AccelChip Inc contributes an interesting paper on how to automate the DSP design flow from MatLab to synthesized gate-level netlist along with generated testbenches. I strongly believe that the syntax of a specification really does not matter that much, as far as it is rich yet concise enough for system-level modeling (specification and verification), the thing matters is the semantics of the specification language from the synthesis point of view– *It is not how you draw it, it is how we view it that really makes sense.* Now if pure top-down approach never works for any practical design space, will the IP-based SoC design approach be the winner? No way, as the high NRE cost associated with SoC, nor is reconfigurable computing, and that leads to the rising of Structured ASIC

On SW development side, similar to HW, the one we care about is performance, and the other we care about is cost. When the computing paradigm migrates from client/server based to distributed based, the browser/server architecture now evolves to multi-tier architecture, and whether it is .Net based web services, or J2EE based web services (or even just IIS based web server versus Apache based web server), performance is always of paramount importance, and Port80 SW addresses this topic quite well in this issue. On the cost side, since the OO technology cannot deliver what it promised, pattern and component based technology takes over in attacking the yet to be solved SW design reuse problem, similar to HW design reuse, separating interface (communication or service) from implementation (action or computation) is the key for sure. In this issue, Stephen Barrette presents a novel way on component-based reuse under such scenario

So Never Too Late, and see you at **INT'L SOFT CHINA 2004**

Loosely Does It

Stephen Barrett

Chief Architect, Wilde Technologies
stephen.barrett@wildetechnologies.com

Abstract

Software constructed using traditional development paradigms suffers from an inherent inflexibility. This is due to both the tight coupling of the components of the system, and to the difference between the system's actual implementation and the architect's view of it. A loose coupling paradigm can overcome these problems, and deliver flexible enterprise software, but the architect must develop and deploy mechanisms that provide loose coupling if he or she wishes to architect for change. This is an expensive and risky under taking. Wilde reduces both risk and cost by supporting a loose coupling paradigm that is driven directly from application architectural modeling. It is built on and supports component technologies deployed today. It delivers more rapidly constructed, better defined and more easily maintained software than any technology currently available.

1. Introduction

A view of components has emerged that posits the component as a service provider, packaged using a component technology, and often available across the network. The service oriented architectures that we see emerging through standards such as WSDL, UDDI, and SOAP pay homage to this kind of model: the component as a service provider within a larger application. The enterprise has used this kind of client/server model as the backbone of enterprise systems for years. Over time, it has become clear that enterprise software is inflexible and very expensive to maintain, with some 70% of development effort going into the maintenance of these systems, once deployed. Leaders in the area of component-based development have identified a shortcoming in the modeling of the enterprise software system that substantially contributes to the problem.

They have proposed a solution based on a richer component model, built not around simple service provision, but rather around a more symmetric model that includes service requirements within the specification of a component [2]. I t is argued that this model better represents components in medium and larger scale software systems, as co-operating computational entities (i.e. n-tier), and not simply layers of dependent functionality. From an architectural point of view, symmetric component modeling adds clarity to the modeling of the enterprise. This makes maintenance easier to define and control, but the key benefit of a symmetric component model is its support for a component paradigm based on loosely coupled components. Coupling in software is the idea that one piece of software has dependencies on other software. This categorization includes dependencies such as a dependency on the host operating system (your spreadsheet won’t run on Linux), a dependency on simultaneous execution (as in client/server systems), and a dependency on shared data source (as with multiple applications accessing a common data base). In the context of component-based software, we focus hereon dependencies on other services, their location, and interfaces.

We thus define a loosely coupled component as one that can be bound late in the development cycle to partners that provide it with services. This means that the identity of service partners can be configured after the component has been deployed, and possibly while the component is executing. Software systems constructed as compositions of loosely coupled components are in theory more flexible, and less expensive to maintain.

In practice, flexibility costs - it takes substantial effort and skills to implement loosely coupled components, and assemble them into applications. Looser coupling requires a greater up-front investment in design and implementation. You must make use of mechanisms such as connection points, directory services, or event services. This almost certainly means utilizing and building skills in one or more component technologies.

Once this has been achieved, it is then necessary to build and maintain an architectural code base that assembles and configures the components to form a system. This requires a development team that has a high degree of skill in component technologies, and a strong‚buy-in™ regarding the benefits of loose coupling and component based development by the architects and other stakeholders. Few architects doubt that loose coupling is a desirable goal in a software design, but the pragmatic recognize it as an expensive investment to be used only when its need can be predicted. The challenge then is to provide a paradigm that cost-effectively supports the implementation of loose coupling. Wilde is an architecture implementation platform that provides this kind of support. Before describing how Wilde works, and what the benefits of Wilde are,
We want to look at why loose coupling is indeed a necessary paradigm for solving the flexibility problems in the enterprise

2. Loose Coupling: A Necessary Paradigm

At its most simple, loose coupling is about reducing the dependencies between pieces of code. In the context of component based software, we focus here on dependencies on other services, their location, and interfaces, and define a loosely coupled component as one that can be bound late in the development cycle to partners that provide it with services. This means that the identity of service partners can be configured after the component has been deployed, and possibly while the component is executing. Software systems constructed as compositions of loosely coupled components are in theory more flexible, and less expensive to maintain. To understand why this is so, it is necessary to study the limitations of a traditional development approach, and what occurs as components are connected together to form the software system. Typically what results is an interconnected component set that is very hard to break apart. We examine this later in The Drawbacks of Coupling in the context of a simple example software system. Following that in Loosely Coupled Components, we apply the loose coupling paradigm to the same system where we see that loose coupling preserves flexibility in a systems implementation. This allows us to make architectural changes without the traditional expense

2.1 The Drawbacks of Coupling

All code in a software system fulfils one of two functions. Some code implements business logic algorithms, stores data, creates and responds to the button press and so forth. We cal l this behavioral code in that it provides some desirable execution behavior. The ecotype of code assembles the behavioral code, binding it together ensuring that it cooperates to achieve the desired overall system goals. We cal l this architectural code because it implements the application architecture. It typically amounts to some 30% of the overall code basin general l y takes the form of constructor parameters, directory lookups, object creation in mainlines and in class methods, distributed invocation framework, and so on. This kind of activity seems so second nature to the typical developer that it seems odd to award it equal status with real functionality. However, it t is precisely this architectural code, and its structure, that has the principal impact on system flexibility. I f you don't believe this then we suggest you peruse redesign Patterns, by Eric Gamma et al. [2] paying particular attention to io behavioral patterns such as the Mediator, and structural patterns such as isBridgelland in Compositell- it™s not what you do; i t ™s the way that you do it!

A loose coupling paradigm implies that a system it s constructed from two separate and distinct code bases: one that implements the components, and one that assembles the components together. In other words, Behavioral code and architectural code are separated. This is quite different from traditional development approaches in which a single code base is implemented that combine architectural and behavioral code in the same computational units. The traditional development process makes no distinction between architectural and behavioral code. The system model, that describes necessary behavior and the structure of the system, is used to produce a single code base. This approach to system construction results in software that is overly complex and difficult to change, hard to understand, and thus expensive to change.

The lesson to be learned from this is that architecture should, if possible, be implemented separately from behavioral code. This means avoiding the embedding of architectural code in behavioral components. This may seem like a small point, but is in fact a major issue as you scale up from the class library or application to the enterprise. At the application level, you can make decisions like we will use Javelin, or acrobat is our distributed bus choice, it but not across the complete enterprise. At that level, aspects of a software system that inhibit flexible and cost effective reconfiguration have a disproportionate impact on the cost of maintenance.

In the following subsections, we look at why the construction of code that implements a system's application architecture is more difficult than is commonly assumed, and why in the absence of a clearly defined loose coupling paradigm, this typically results in the embedding of architectural code in behavioral code. We explain why this results in expensive maintenance and why the architect's view of the software system diverges

From the real code base so that it becomes increasingly difficult to change the system to accommodate new requirements.

2.1.1 Embedding Architecture Makes it Difficult to Change

Implementing architecture is a complex business. Even if you work only with classes, the complexity inherent in the creation and maintenance of the relationship between client and object is often overlooked.

To illustrate this, let's explore how we might build the gateway component using C#: a language that lets us takes a class like view of component technology. We will assume that the method do Trade is supposed to do some internal computation before delegating the trade to the external trade service.

```
Public bolo do Trade(string arg_account, Trade Details arg_trade)
{
if(doTradeOK(arg_account))
{
return m_trade Service. doTrade(arg_account, arg_trade);
}
else
return false;
}
```

Figure 1: A simple inter face method

The first issue is how the gateway obtains a reference to the trade service - how do we create the relationship? There are many mechanisms and pat terns for doing this.

Figure 3 illustrates a constructor based approach that is typically used.
The constructor includes a reference to a trade service object as a parameter.

```
void Gateway(ITradeService arg_tradeService)
{
m_tradeService = arg_tradeService;
}
```

Figure 2: A non-default constructor for the Gateway.

Let's be clear. This constructor is architectural code. I t is about connecting behavior together. This const rector is one of the many ways to do that. This approach would have to be modified if we chose to implement the gateway as a COM component, because there can be no default constructor in a COM component. In such cases presumably we would provide an alternative **Init** method or property (note to architect: please remember to modify the UML model to take account of COM technology requirements).

Note that if we go with this kind of constructor based solution, or initialization method solution, and then the client is required to be aware of the identity of the trade service so that it can initialize the Gateway instance (see Figure 3). This may be acceptable (if you don't mind your clients configuring your services) in certain circumstances, but it would be an unacceptable solution in a real client/server scenario. In such a case, the Gateway will already be executing, before the client connects.

```
Class MyClient
{
Itrade Service m_trade service = //get the trade service please
IAction m_gateway = // get the gateway please
MyClient()
{
m_gateway.Init(m_tradeservice);
}
}
```

Figure 3**:** A trade service aware client

To solve this, we could engineer a third component (e.g. a server mainline) that manages the creation of the trade service and Gateway component instances and we would configure the Gateway component instance so that it holds a reference to the trade service instance.

This is not a bad idea, but because it constitutes additional implementation effort, it is more likely that in practice, the Gateway will be implemented to either directly create the trade service, or obtain a reference to it via some directory lookups mechanism, either a standardized mechanism such as UDDI, or some is spoke internal service. So, our code might look like Figure 4

```
public Gateway : IGateway
{
ITradeService m_tradeService = null;
Gateway()
{
m_tradeService = new (ITradeService) TradeService();
}
}
public bool doTrade(string arg_account, TradeDetails arg_trade)
{
if(m_tradeService == null)
throw new ExternalServiceError("where's my trade service?");
if(doTradeOK(arg_account))
{
return m_tradeService.doTrade(arg_account, arg_trade);
}
else
return false;
}
```

Figure 4**:** *A non-parameterized gateway*

Let's be clear here too that this is architectural code, buried within the behavior. As Figure 4 demonstrates, application code is typically writ ten so that it combines both behavioral code and architectural code. This is bad for flexibility because it is as expensive to change architecture, as it is hard to identify the scope and location of necessary change. It goes without saying that; anything that requires recoding does not bode well for flexibility in any case.

If we have a constructor-based solution such as described in Figure 2, then it might look like we can apply this architectural change without modification of the Gateway component. However this is only part of the story. If the new component is implemented using a new technology, such as Web services, then some development effort is incurred in order to pass the gateway a simple reference. Where should this code go? Who is responsible? It depends on how the developers implemented the architect's application architecture. Our model has no well-defined mapping to the distributed component code base.

If we had opted for a gateway implementation as described in Figure 4, then a degree of recoding of the Gateway itself would be necessary. Because we do not have an external code framework for the assembly of what is now a three component system, the following code segment (Figure 6) is quite likely to be embedded in the Gateway. Actually, something far more complex is likely to be needed, but there is no need to overcomplicate for this discussion. Note how application architectural code is becoming spread through behavioral code -why does the Gateway have a constructor that configures external components? If this particular three component system becomes obsolete in a couple of months, where do we look to change the architecture? Deep in the implementation of the components? If at some point, we wish to replace our Gateway component with a new implementation, this kind of architectural code will have to be replicated somewhere -- where should it go?

```
Gateway()
{
// create the trade service, and filter component
ITradeService m_ts = lookup("trade service");
IAction m_fltr = lookup("filter");
// configure so chain is this-> filter -> trade service
m_action = m_fltr;
IConfigure cnf = (IConfigure) m_fltr;
cnf.InitTradeService(m_ts);
}
```
Figure 6**:** Configuring the component set

We're not done yet. The new filter component must be configured to delegate trades to the trade service. We have to consider how it obtains references to external services. For example, it might look up such services using UDDI. The implementer of the filter component may have specified this so that it exhibits greater deployment flexibility, and so is easier to reuse in different systems. If so, then it will be necessary now to expose the trade service as a Web service, and possibly prepare a configuration file or write some code to initialize the new filter component so that an instance will perform the correct lookup. What happens if we wish to run two system types in parallel - for example, the original system, which does not perform account credit checking for internal traders, and the new version, which does? Do we have to up grade the gateway so that it accesses this Web service-based trade service implementation in systems that do not contain a filter component? Are there other components that access the trade service directly? Are different versions of components such as the Gate way component now needed?

These are the type of questions faced, with an increasingly disjointed view of the system architecture as the system develops. So far, we have considered an extremely simple system consisting of just three components, linked together in a pipeline. Scale this up, and consider perhaps three hundred components, ten legacy systems, two distributed bus architectures, Visual Basic on the desktop, COM, and Web services and so on. System assembly is less trivial than we would like.

Another argument is that the complexity isn't that hard to handle if done properly. Gamma et al. [2] have described design patterns such as the Mediator, Adaptors, Composites, and Abstract Factories and so on, which they propose can be used to handle this kind of complexity. Perhaps we are describing a worse case scenario. For example, mediators can instantiate and configure component sets, assembling them into coherent subsystems. Provided component references can be configured by this mediation code base, the complexity is manageable, and the kind of flexibility we are looking for is achievable. If you buy this, then you've just bought into loose coupling - you've advocated a separation of behavior and architecture. The degree to which you are willing to enforce the separation impacts directly on the flexibility of your software system going forward.

2.1.2 Embedding Architecture

Leads to A Design Disconnect as argued in the previous section, embedding architectural code in behavioral code makes it difficult to change a system in response to new requirements.

The issue here is whether the architect can be certain that the implemented system accurately reflects the specified model. The key issue is certainty. The actual state of the system is not nearly as important as the accuracy of data available to the architect. Unfortunately, what tends to occur, even when architecture is implemented in code separately from behavior, is that small mistakes lead to a disconnection between what was specified and what was implemented. This is exacerbated by the difficulty in extracting a sensible view of the architecture from the implementation, as the architectural code is literally dispersed throughout the behavioral code. We call this the 'Design Disconnect'.

There is more to consider. What if we have used directory look up technology to obtain a reference to the Trade service rather than a class instance creation? Would we have a more tractable code base then? I t would depend on the nature of the lookup technology in use. If we were using a standard technology like UDDI, which is probably giving us access to a Web service, which would in effect be a Singleton, we would probably be OK. Of course if the lookup technology were be spoke, internal to the enterprise, we would have to check the documentation.

So far we've briefly examined some pitfalls in constructing architectural code. It's possible for developers to make subtle mistakes in the interpretation of the UML modeling that have architectural impact. Worse still, a fairly detailed understanding of the technologies in use in the software system is necessary in order to correctly interpret what architectural modeling is implemented by specific architectural code. So once the architect hands over the architecture, how does he ensure his modeling remains relevant and more importantly, accurate?

Because these diagrams give very little detail, we have to look at the code (and project context) of the components in order to understand what application architecture has actually been implemented. Worse look for this information.

The design disconnect has the most damaging impact at the point where the system must undergo change. As we add more and more interdependent components into the mix of our enterprise system, what results is an increasingly rigid and hard to maintain enterprise. What is often forgotten is that enterprise systems and intra-enterprise systems are inherently n-tier. The key point is that as these complex systems diverge from the original architectural viewpoint, it becomes increasingly difficult to understand and plan for change. This implies additional costs and risk.

2.2 Loosely Coupled Components

A loosely coupled component is one that can be bound to its partners late in the development life cycle. The key benefit of loose coupling paradigms is that they break the hard relationship between client and server, so allowing partners to be specified orthogonal to the implemented components set. Architectural decisions can be made late in the development cycle, even revisited inexpensively at deployment. This provides a very important kind of flexibility in the context of a component-based system.

Consider the following code segment from the Gateway component of the trading system tutorial that comes with Wilde 1.0 (Figure 7). Some detail has been omitted for clarity. It is a COM component, but one implemented to be loosely coupled. I t implements an interface, as we would expect of al l components. It also implements a connection point. The connection point is a mechanism for a COM component to express a requirement for a service provider, that service being defined in terms of a specific interface type.

```
class CGateway :
public CComObjectRootEx<CComSingleThreadModel>,
public CComCoClass<CGateway, &CLSID_CGateway>,
public IConnectionPointContainerImpl<CGateway>,
public IAction,
public CProxyIAction< CGateway >
{
public:
CCGateway()
{
}
BEGIN_COM_MAP(CGateway)
COM_INTERFACE_ENTRY(IConnectionPointContainer)
COM_INTERFACE_ENTRY_IMPL(IConnectionPointContainer)
COM_INTERFACE_ENTRY(IAction)
END_COM_MAP()
BEGIN_CONNECTION_POINT_MAP(CGateway)
CONNECTION_POINT_ENTRY(IID_IAction)
END_CONNECTION_POINT_MAP()
public:
// IAction implementation omitted for clarity
};
```

Figure 7**:** *COM implementation of Gateway Component*

The powerful thing about this type of component modeling and implementation is that because the service requirements of a component components are decoupled from their partners, and can thus be flexibly combined without expensive recoding and recompilation of the components. Note its symmetric nature it provides and requires interfaces. We've included the trade service component type, to illustrate how these components are related. We've also included one of the trade services interface requirements to hint at the complexity of a typical component. The tree view to the left of the main diagram contains a more detailed description of the trade service. Look again at the code of Figure 7. There is no attempt to create the trade service, or lookup the trade service. This is loose coupling. I t means that provided we have the appropriate underlying assembly support in place, we can bind this Gateway component to a component that implements the trade service. This might look like polymorphism, but it's far, far more powerful. It is a formalization of the decoupling of client and server at the component typing level, and it is this that leads directly to far more flexible software systems.

I f our components are implemented according to a loose coupling type modelAll that must be changed is the implementation of the application architecture. That might be implemented as a set of mediator classes, some kind of script, or constructed using some other mechanism. So, in theory, any architectural change can be achieved without affecting the behavioral components. In theory this is extremely cheap. In theory what results is agile IT capable of speedy response to changing business requirements. In practice, as mentioned before, implementing this kind of loose coupling paradigm is expensive, requiring expert knowledge of a variety of infrastructure mechanisms. In the next sect ion, we describe Wilde, a platform that supports this paradigm directly

3. Wilde

Wilde is a first of breed Architecture Implementation Platform that delivers a highly flexible cost effective platform for distributed component based development. Wilde enables a uniquely structured software system that is rapidly assembled and easily and cost effectively maintained. The product has been built from the ground up to support a Component oriented development process that takes as its starting point the following key ideas.

1) Software systems are constructed by defining and composing components that provide distinct functionality, and that are deployed across a network. These compositions are inherently n-t ire.
2) The composition must be open, meaning components can be added, changed, removed, deployed on different application servers etc.

3) A range of component technologies are used in the physical component Implementations.
4) Change is the norm; system maintenance post-deployment is the principle development activity. Greenfield construction is rare.
5) Managing change requires a full and correct understanding of a system's application architecture from which to drive change.
6) There is no such thing as an application. Every software system is potential l y a component in a larger system.

Wilde supports this feature set as a platform that supports the structuring of software as UML architect compositions of loosely coupled components. This results in a far more flexible software system than can otherwise be achieved, with substantial cost benefits both during development, but most importantly, in the maintenance phase, where most effort is expended.

3.1 Wilde Platform

Wilde is delivered as a platform with a design time UML tool for the specification of application architectures of distributed component based systems, and a runtime component that executes these specifications at runtime. Like all component-based technologies, a Wilde system is deployed and executed by combining components. The key differentiator with the Wilde platform is that it uses UML based application architecture model directly as part of the executing software system

A Wilde system is launched by passing the UML model that describes it to the Wilde runtime. This runtime interrogates the model, identifies the components and the nodes in the network on which the system is to execute, and then launches the components and configures them so that they are bound together, as described in the UML model. What results is a set of components executing on the various nodes described in the model. These components might be backend components responsible for business functions, front-end applications presenting a GUI to users, or any other type of components. Wilde supports COM, .Net and Web Services based components. Components used in Wilde need not be implemented specifically to be used with Wilde, which implies that the full existing component asset base of an organization is available for use immediately in Wilde systems, without modification.

3.2 The Benefits of Wilde

Wilde has an immediate benefit in the development process but a more important return on investment after a system is deployed. Up front, because Wilde manages the distributed execution of component-based systems, building the required system infrastructure literally at runtime, the development of components is radically simplified. Wilde eliminates the need to encode an application architecture that describes a systems composition in terms of the component set and how those components interact. This eliminates the kind of problems discussed in this article from the development process, speeding up the development process.

This key ongoing benefit of Wilde is the ennoblement of a development process and deployed system that is fundamental l y more flexible and open once deployed. The

architect is in a greatly improved position in dealing with new business requirements. The actual system architecture is visible and directly malleable. This degree of control, which is not delivered by any other technology, allows the architect to both estimates The cost of change with greatly increased accuracy, and control its delivery with greater clarity. Wilde delivers a flexible IT function.
This is explored further in Section 3.3.

3.3 Wilde --The Loose Coupling Platform

As mentioned above, one of the key inhibitors to adopting loose coupling is the cost and complexity of application architectural code that performs component based assembly. To make system assembly easier, the Wilde platform provides a GUI for the modeling of component based application architectures, as UML models. The Wilde platform uses this model directly, dispensing with the need for an application architecture code base that manually brings components together. What You Design is What You Get (WYDIWYGŽ)? This approach means that system assembly is a more rapid process upfront.

However this is only half the story. The more important attribute of a Wilde system is that the architect can make changes to the application architecture model that have a direct effect on the software system, without requiring modification of an architectural code base. This is because with Wilde, application architecture is not committed to code at all, but instead is retained in its original UML form. Modifying the model is modifying the system implementation. The Wilde platform assembles software systems directly from this modeling. There is a one to one mapping between the code necessary to assemble loosely coupled components into a software system, and the model of the application architecture of that system. Implementing an architectural code base to instantiate and configure a component based software system is unnecessary and risky, when compared to a platform approach that automates that task. I t is faster and safer than rolling your own.

System assembly is one side of the loose coupling problem. The other is the technical skill set needed to build loosely coupled components. To make component development easier, Wilde provides a component skeleton assembly technology, which delivers compiled. NET assemblies that can be used as the basis for loosely coupled .NET components. Skeleton assemblies contain a simple set of classes that provide an Object oriented view of the required component's symmetric component type. This isolates the developer from the component technologies, and deployment decisions enforced in the application architecture of the specific software system, and indeed the application architecture itself. I t is actually easier to build a loosely coupled component with Wilde than to build a traditionally coupled component, or a loosely coupled component using some bespoke mechanism. **Figure 19** illustrates the implementation of the Gateway component in C# using a skeleton generated from the Gateway type. Note how access to external services is object based, and independent of the implementation of those services.

But most importantly for the architect, it ensures that the components specified are the components delivered, without any unknown dependencies that have to be hunted down. Simplicity is the promise.

Architects define architectures and component types; developers implement components. So, if we briefly consider the filter component scenario discussed above we see that with Wilde, the modified application architecture is used directly in the executing system. Providing that a suitable (i.e. loosely coupled) component implementation of the filter component is available, there is nothing left to do. If we don't yet have a filter component, the architect or developer can generate a skeleton for it, so driving the correct implementation of the component.

We have not considered deployment so far. With Wilde, the application architecture model can specify deployment of components in addition to component relationships. If we executed this system with Wilde, the Wilde runtime would create the necessary distributed infrastructure without programmer intervention. There is no need for programmer intervention, and no requirement for the component development teams to possess distributed systems skill sets. With Wilde, you just model it.

4. Summary

Software constructed using traditional development paradigms suffers from an inherent inflexibility, that is due both t o the tight coupling of the components of the system, and to the design disconnect that opens up between the system's actual implementation, and the architect's view of it. Loose coupling is a vital and powerful paradigm for the delivery of flexible enterprise software. It succeeds both by breaking the coupling between components, and by separating out application architectural implementation so that it is less difficult to maintain. However, to implement loose coupling, you need a component model that is symmetrical; that describes components as service providers but also as consumers of services. The component technologies that bind the enterprise together lack inherent support for this kind of symmetrical type model, and thus it is up to the architect to develop and deploy mechanisms that provide loose coupling if he or she wishes to architect for change. This is an expensive and risky under taking.

Wilde reduces this risk and cost by supporting a loose coupling paradigm that is driven directly from application architectural modeling. I t is built on and supports component technologies deployed today. It delivers more rapidly constructed, better defined and more easily maintained software than any technology currently available.
It's available now for you to try.
.

Design for Manufacturibity
The impact on the Physical Design Stage and flow

Dr. Danny Rittman
danny@tayden.com

Abstract

In this paper I present the impact of sub-wavelength optical lithography for new EDA tools, IC Layout Design flows and manufacturability. We discuss the necessity of corrections for optical process effects (optical proximity correction (OPC) and phase-shifting masks (PSM)) and will focus on the implications of OPC and PSM for layout design and verification methodologies. Our discussion addresses the necessary changes in the design-to-manufacturing flow, including infrastructure development in the mask and process communities as well as opportunities for research and development in IC physical layout and verification stage.

Reticle enhancement technologies (RET) like optical proximity correction (OPC) and phase shift masking (PSM) have significantly increased the cost and complexity of sub-micron nanometer photomasks. The photomask layout is no longer an exact replica of the design layout. As a result, reliably verifying RET synthesis accuracy, structural integrity, and conformance to mask fabrication rules are crucial for the manufacture of nanometer regime VLSI designs. New EDA systems were recently developed consists of efficient wafer-patterning simulators that is able to solve the process physical equations for optical imaging, resist development and hence can achieve high degree accuracy required by mask verification tasks. These tools are able to efficiently evaluate mask performance by simulating edge displacement errors between wafer image and the intended layout. I'll discuss the capabilities for hot spot detection, line width variation analysis, and process window prediction capabilities with a sample practical layout. I'll also elaborate the potential of the new physical model simulator for improving circuit performance in physical layout synthesis.

1. Introduction

Reticle enhancement technologies for VDSM (Very Deep Sub Micron) integrated circuit manufacturing has dramatically complicated the mask data and increased the cost of advanced photomasks. The increase in pattern complexity due to optical proximity correction (OPC), the tight requirements for Critical Dimension (CD) control, and the difficulties in defect inspection and repair all contribute to the manufacturing cost increase. For phase shift masks (PSM), the problems are compounded by additional requirements such as controlling the etching of multiple materials, alignment of multiple layers, and inspecting small defect with weak signals. In addition to the added complexities in mask making, the growing array of Reticle Enhancement Technologies (RET) also put more constraints on the physical layout design and verification as physical layouts must be RET compliant and conform to the mask fabrication rules. For instance, the avoidance of phase conflicts in alternating PSM and generating OPC-friendly design layout are examples of those new constraints. Physical design and verification flow nowadays have to be overhauled to address various wafers and photomasks manufacturing issues explicitly early in the design flow to achieve high quality fabricated silicon at a reasonable point on the price-performance curve. There are tremendous amount of research efforts from the industry and academia for this issue. The complexities in mask data and manufacturing make it highly desirable to verify and optimize the mask data independently before committing to the costly fabrication process. An effective method for post-RET mask data verification is to simulate its image on the silicon wafer and compare it with the original design intent. This method places mask data in its intended operating environment and evaluate its performance metrics that have direct impact on wafer imaging. A simulation based verification system can evaluate the process for a product and give warning on certain performance limiting spots on the layout and thus significantly reduce the risk of mask data errors. Once the troubling spots are identified, localized corrections can be applied to extend the process window in an intelligent way.

The existing model based mask layout verification systems have a few areas that require further improvement. First, they are typically implemented with the same simulation engine with model based OPC. Sharing the simulation engine with OPC, the verification also inherits the errors of the OPC model. The logical dependency jeopardizes the probability of finding OPC errors, and reduces the reliability of the verification. A process window is the range of process parameter variations under which the line width remains within limits Secondly; they employ empirical modeling approaches that cannot easily track acceptable variations in process conditions. In order to sample a different condition in the process window, a different set of models has to be developed, which consumes significant effort and time.

In addition, there is no inherent reason why one set empirical models can judge the result of another if they are derived from the same set of mathematical formulation and training patterns. A full-featured photolithography simulator for mask data verification has been developed for the past decade by the major EDA vendors. (Mentor Graphics, Cadence) This type of simulators has been used extensively in lithography process development where they have demonstrated high accuracy for process predictions.

A mask data verification flow around the physical lithography simulation core that is independent from the OPC engine, thus free from the logical dependency between OPC and its verification. The use of physical models opens the possibility for achieving higher prediction accuracy on complex layout configurations. In addition, physical model can naturally predict the pattern transfer behavior under process variations such as focus change. Furthermore, a physical layout design can efficiently leverage this physical model simulator to improve circuit performance and reduce the manufacturing variations.

2. Physical Model Based Mask Layout Verification Flow

A standard flow for reticle enhancement and optical proximity correction with model based mask data verification block outlined with gray shading is shown in Figure 1. Here we consider model based OPC as an independent module because it is also needed for all other reticle enhancement techniques as well as standard binary masks. The main manufacturing flow is shown on the left hand side. The design layout from a customer is modified with reticle enhancement, followed by model based OPC to produce a set of mask geometry data that is suitable for mask manufacturing. The model generation flow is shown on the right, where a test layout is printed with the same pattern transfer process to produce an experimental data set for empirical model fitting. The resulting model can then be used in the OPC engine to predict the wafer CD error. From that, the amount of mask correction can be calculated.

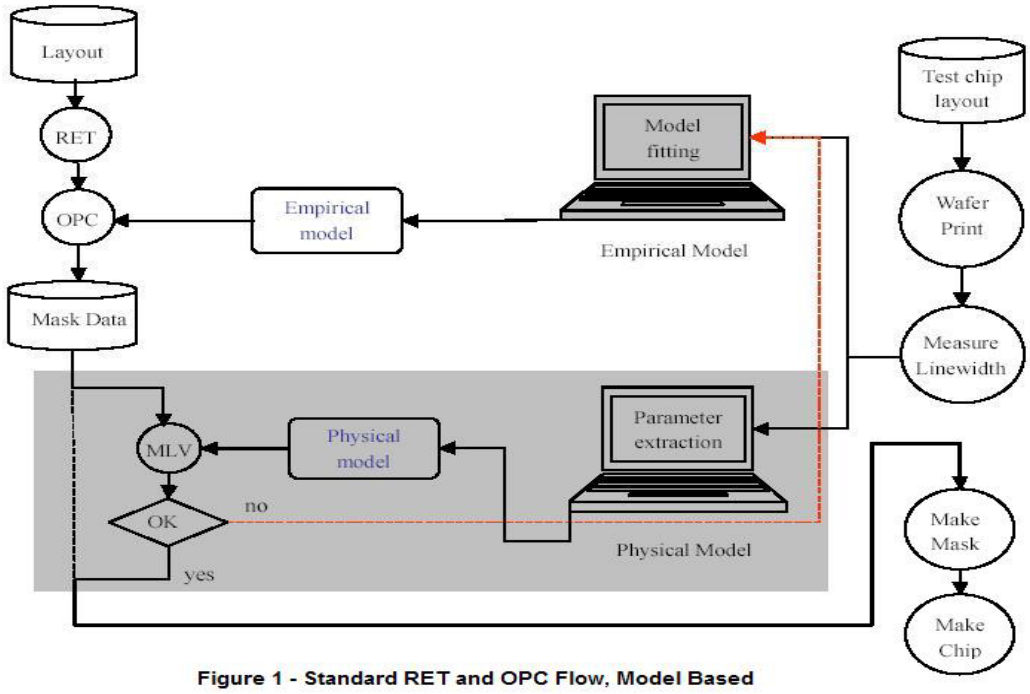


Figure 1 - Standard RET and OPC Flow, Model Based

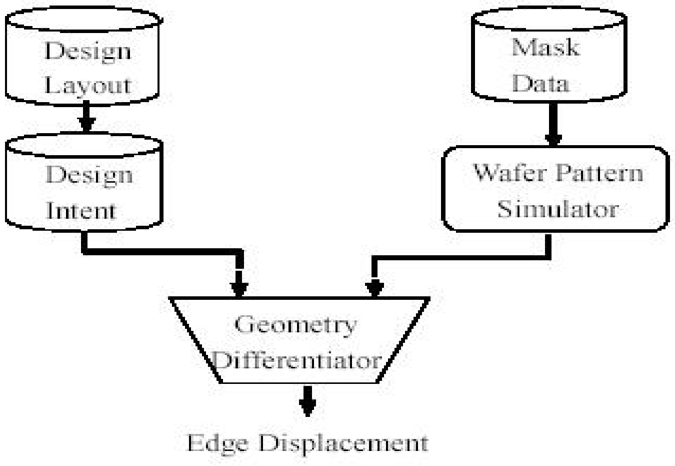
To implement physical model based mask data verification, one must calibrate the physical model by extracting the process parameters from the same data set used for empirical model fitting. The main task here is to obtain resist-processing parameters such as development rate parameters and the post exposure bake diffusion length. The model can then be used in the mask layout verification (MLV) block to check the post-OPC mask data.

The verification can be performed on the entire mask or, to save processing time, on sections of the mask that are most likely to have pattern transfer problems. In case such problems are found, the simulation pattern produced by the physical model and the corresponding mask section can be added to the data set for recalibrating the empirical model. This feed back system will gradually make the empirical model to become more predictive over time as more and more cases are added to the training set. At some point, the confidence level on the empirical model will reach a point when only occasional verification is needed in the full production mode.

A detailed Physical Model based mask verification (PM-MLV) block is sown in Figure 2. The intended layout is derived from the design data by applying appropriate geometry operations such as scaling and sizing. This design intent is used as the standard for comparison. The other path of the verification process takes the mask layout as input and run through the wafer - patterning simulator. It simulates the wafer pattern by solving the equations describing image formation; resist exposure, post exposure bake, development and etching.

The simulation parameters are set such that the resist and etching processes are accurately captured in the model. By doing so, any changes on the RET type, exposure tools settings, and thin film stack can be predicted by the physical simulator.

The output of the high accuracy wafer-patterning simulator is the outline of wafer image. The pattern differentiator in Figure 2 compares this with the design intent and outputs the difference between the two patterns. The system characterizes the pattern difference by calculating the displacement of a line segment on the intended layout to its counterpart on the wafer image.



Positive edge displacement indicates that the wafer pattern falls outside the original design polygon, and is larger than the design intent. Similarly, a negative edge displacement indicates that wafer pattern is smaller than the design intent. In order to better capture the variations along a polygon edge, the edges of the design intent polygons are subdivided into shorter segments for edge displacement calculation. The subdivided edge segments are classified in a feature specific way in the data representing the design intent. For example, the segment located on a line end will carry a special flag indicating that line end pull back will be measured for this segment. Similarly, segments at long line edge may carry another flag indicating that transistor gate or local interconnect variations will be measured at these segments. The feature specific classification flags help a user to impose different verification tolerance for each feature class of edges. By doing so, the verification process can be customized to better reflect the yield and performance of the product.

3. Hotspot Detection

Processing hotspots are the locations in the design where the magnitude of edge displacement is exceptionally large. Hotspots can form under a variety of conditions such as the original design being unfriendly to the RET that is applied to this chip, unanticipated pattern combinations in rule based OPC, or inaccuracies in model based OPC. When these hotspots fall on locations that are critical to the electrical performance of a device, they can reduce the yield and performance of the device.

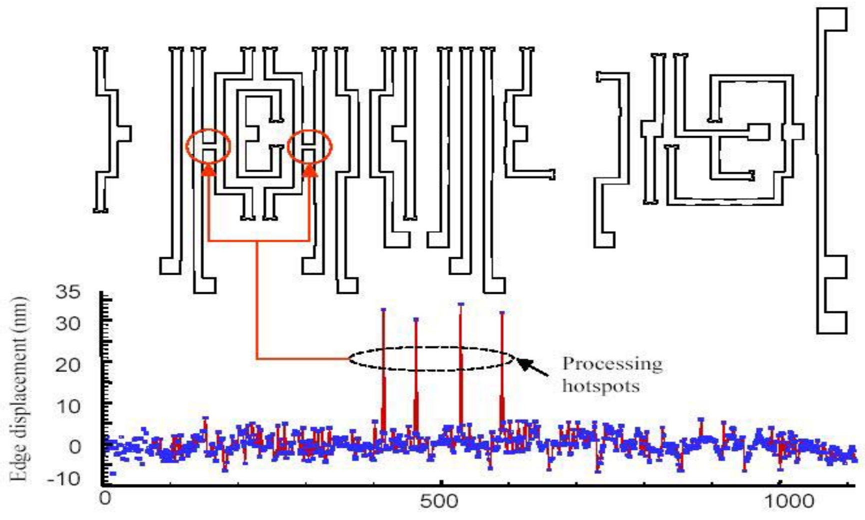


Figure 3 – Hot Spots Detection

Physical model based mask layout verification (PMMLV) can identify the hot spots and subsequently repair them by applying physical model based OPC (PM-OPC) at these locations. (Shown on Figure 3) The layout is for the poly gate layer with 90 nm target line width and dominating pitch of 300 nm. The cell size is approximately 11 um by 6.6 um. The mask layout is created by model based OPC using aerial image model only.

After OPC, the standard deviation of edge displacement error is calculated to be 0.71 nm, which confirms that the wafer pattern as predicted by the aerial image simulation is in good agreement with the design intent. The performance of this OPC mask created with simple aerial image model is verified using an optimized isofocal2 resist recipe that is a more realistic description of the patterning process. Also shown in Figure 3 is the output of the pattern differentiator. The edge displacements are evaluated on 887 line segments on this cell.

Our PMMLV process discovers four segments with large edge displacement as shown in Figure 3. Interactive exploration shows that these points are located on either side of the short horizontal bars in "H" shaped patterns. The standard deviation for edge displacement also increased 240% from 0.71 nm to 1.7 nm. This set of verification result shows that the mask data created by OPC with simple aerial image model would result in worse process and circuit performance than that suggested by the small correction residual.

4. Proximity Induced Line Width Variation Statistics

Variations in line width due to lithography and etching often limit the performance of a circuit. The line width variation pattern changes as focus varies within allowed process control limits. Existing OPC methodology is aimed at reducing the line width variability at a nominal focus point, without considering the potential impact of focus change.

In this case, physical model can be applied to obtain more complete and meaningful line width variation statistics by considering focus and other process parameter variations, the result of which can be used for performance optimization.

Figure 4 show the histogram for the edge displacement under defocus for a mask produced by physical model based OPC. At best focus, the mean edge displacement is zero, indicating an on-target CD distribution at 90 nm. The standard deviation of the edge displacement is 0.97 nm, which represents the residual of PM-OPC process. When this mask is printed under 0.15 um of defocus, the distribution broadens into a bi-modal form. We can clearly see the increase in the edge displacement envelope under defocus.

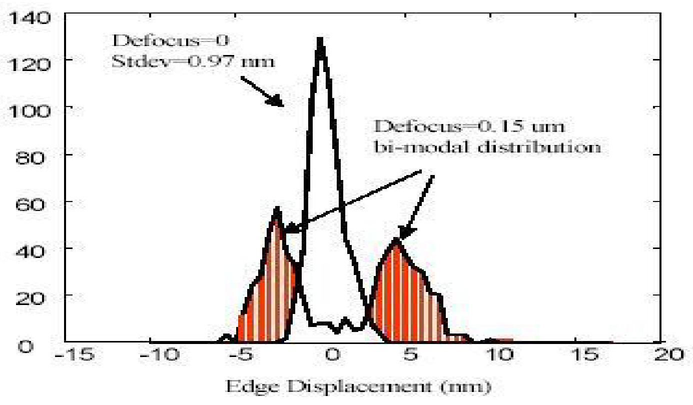


Figure 4 - Edge Displacement Error

The mean of the edge displacement, however, still stays at near zero, as in the best focus case. On average, the line width is not changed under defocus, as the number of edges with positive displacement roughly equals the number of edges with negative edge displacement. This behavior is consistent with theis focal process model we developed for this circuit. On the other hand, if the same circuit layout is corrected with aerial image model and verified using aerial image model, a -14 nm average edge displacement will result with 0.15 um defocus. The range of variation also increases by nearly a factor of 7 from 0.71 to 3.5nm. The large difference in response between this and physical model based OPC and verification shows the strong influence of models on the OPC and verification results. The edge displacement statistics produced by the physical model based OPC and verification process can be used in physical design flow to make ECAD tools manufacturability aware such that process variations can be reduced and circuit performance can be improved. This concept is illustrated in the following section.

5. Impacts on Physical Design Stage and Flows

The circuit design and mask processing are still basically separated from each other in current design and manufacturing flow. The design and process development team communicate only through a set of design rules. As we moving into VDSM technologies, we have to explicitly addressing various manufacturing issues early in the physical design flow to attain the best design performance, process window and uniformity in manufacture. Recent approach to this change in design-manufacturing interaction is through advanced process simulation that is transparent to circuit designers.
Figure 5 shows recent mode of design-process interaction. For each new technology node, the equipment community publishes tool specifications early in the process development cycle. These parameters are used to construct physical models well before an intended process becomes stable. These physical models are applied early in the design phase to ensure that the layout can be optimized for the target processing technology.

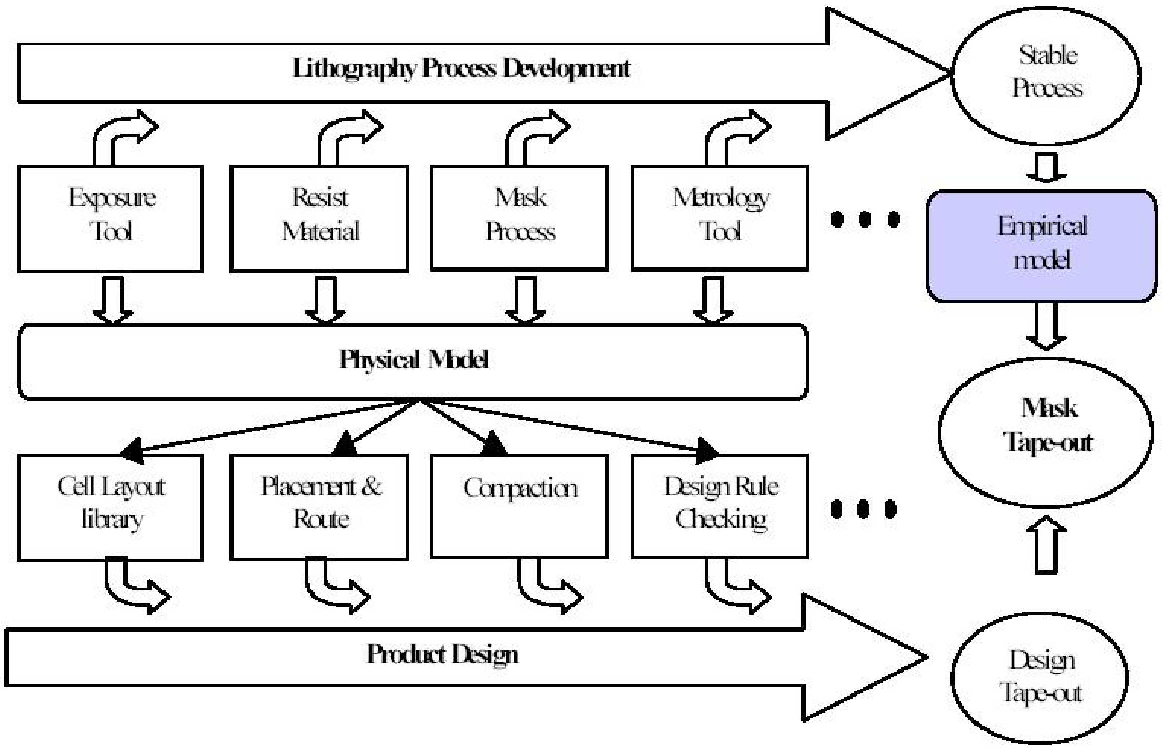


Figure 5 - Recent Design and Process Flow

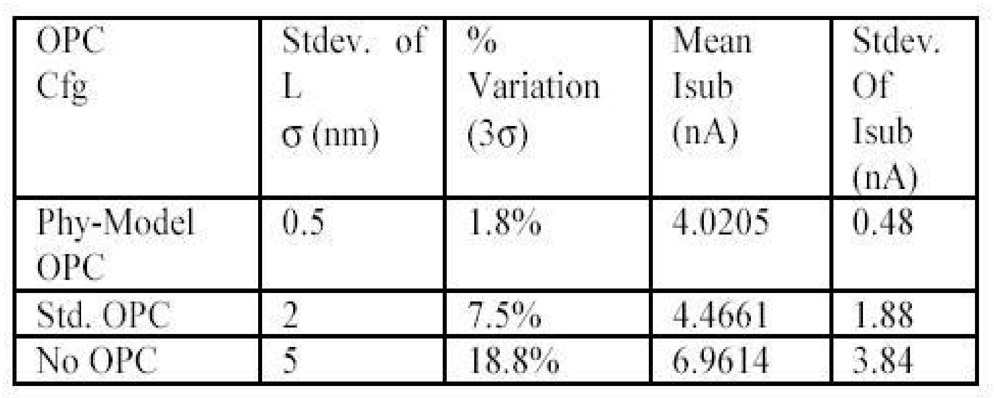
Now designers can pre-characterize OPC related information for each cell and use this information during placement to produce more OPC-friendly layouts. Specifically, we need to know how sensitive (*sensitive factor*) the critical dimensions (CDs) in a cell to its neighborhood patterns and how difficult to compensate the CDs in the cell. If an aggressive OPC is needed (like sub-resolution assistant features which may be outside of original cell layout), then the cell layout area has to be bloated. Now we can build different *OPC configurations* for each cell in the library, each of them has different layout area and OPC performance in terms of statistic errors on CDs.

6. Power Leakage

In the following, I'll show how different OPC configurations can affect the power leakage of a CMOS device. Static power leakage becomes a major concern for designers today, as it accounts for an increasing and significant portion of the total power budget in high-end microprocessors. This situation will become even worse with further reduction of threshold voltage (Vth) of MOS devices. CMOS device leakage currents I_{sub} varies exponentially with the change of channel length L as shown in the formula below:

$$Isub = K(1/L)exp(-CL)$$

where K and C are device dependent constants. As a result, the sub-threshold leakage currents are extremely sensitive to the channel length variations. The mean leakage current of a chip under process variations can deviate significantly from the nominal leakage current in a typical 0.18 um COMS process. The mean leakage current and the standard deviation of a PMOS transistor vary with the changes of its channel lengths due to different OPC configurations. The process used here is TSMC high-performance 0.13 um technology. Table 1 shows our calculated mean leakage and standard deviation of mean leakage of sub-threshold current of PMOS device under different OPC configurations.



OPC Cfg	Stdev. of L σ (nm)	% Variation (3σ)	Mean Isub (nA)	Stdev. Of Isub (nA)
Phy-Model OPC	0.5	1.8%	4.0205	0.48
Std. OPC	2	7.5%	4.4661	1.88
No OPC	5	18.8%	6.9614	3.84

Table 1 - Leakage Current Variation - PMOS Device
Channel Length Variation Range: L=0.08 um W/L=5

It is shown that average leakage will significantly deviate from nominal value if no OPC is used. If OPC is employed, but is not optimized due to poor modeling or unexpected presence of neighborhood patterns, the average leakage will still %10 higher than the nominal value. With physical model based OPC and predictable neighborhood patterns, the channel length variation can be well controlled. As a result, the mean leakage current and its variation are reduced. The statistical performance information of library cells can be leveraged during physical layout synthesis. For example, in the detailed placement phase, all the timing critical or leakage cells (called *critical cells* in the sequel), which are also OPC sensitive are instantiated with their best OPC configurations. Placement will legalize the added areas of those cells during refinement. If a cell is no longer a critical cell, its original layout will be used again. For OPC high sensitive critical cells, a fast on-line OPC process can be invoked to estimate the statistic errors for its neighborhood patterns. If the errors are still too large, some local cell swapping may be applied to get different neighborhood patterns or get more open area (adding dummy cells) around the critical cells or even re-synthesis the corresponding logics to make the resulting cell less OPC sensitive.

This process is repeated until OPC CD errors on all the high sensitive critical cells are under control. Such cell-based OPC and the manufacturability-aware placement strategy bring many advantages: First, it will improve the circuit performance and reduce the performance variations and thus unnecessary guard-banding, and lead to much more predictable circuit performances and manufacture yield. Second, with each layout pre-certified and OPC optimized by physical models, the final tape-out process would likely to be much simpler than the whole chip-wide, essentially flattened OPC and verification processes used today.

7. Conclusions

We observed mask data verification flow in order to prevent data problems to propagate to the expensive mask making and wafer printing stage. New flows, presented to the industry by the major EDA vendors leverages the high accuracy of a wafer patterning simulator that predicts the wafer image by solving the equations that describe the physics and chemistry of the pattern transfer process. These systems address the problems of the existing empirical model based OPC/RET flow and can be applied in parallel to improve the reliability and quality of the mask data. We discussed how edge displacement statistical information obtained from the new model simulator can be leveraged during physical synthesis flow to reduce the performance variations and improve the device manufacturability. No Doubt, There is a tremendous impact on the IC Physical Design and verification phase due to the sub-wavelength optical lithography and this will get more critical as progressing towards ultra sub-micron process. EDA vendors are in constant race to implement manufacturability flows for advanced process. In the next decade we will witness a major increase in efforts from industry and academy in the RET arena.

References

[1] P. Gilbert et al., A High Performance 1.5V, 0.10um Gate Length CMOS Technology with Scaled Copper Metalization, IEDM 1998, pp. 1013-1016.

[2] W. B. Glendinning and J. N. Helbert, Handbook of VLSI Microlithography: Principles, Technology, and Applications, Noyes Publications, 1991.
[3] L. Gwennap, IC Vendors Prepare for 0.25-Micron Leap, Micro-processor Report, September 16 (1996), pp. 11{15.
[4] F. O. Hadlock, Finding a Maximum Cut of a Planar Graph inPolynomial Time, SIAM J. Computing, 4 (1975), pp. 221{225.
[5] A. B. Kahng, S. Muddu, E. Sarto, and R. Sharma, InterconnectTuning Strategies for High-Performance ICs, in Proc. Confer-ence on Design Automation and Test in Europe,.
[6] A. B. Kahng, G. Robins, A. Singh, H. Wang, and A. Zelikovsky,Filling and Slotting :Analysis and Algorithms, in Proc. Inter-national Symposium on Physical Design, 1998, pp. 95{102.
[7] A. B. Kahng, H. Wang, and A. Zelikovsky, Automated Layoutand Phase Assignment Techniques for Dark Field Alternatingtomask Technology, 1998.
[8] H. Landis, P. Burke, W. Cote, W. Hill, C. Hoffman, C. Kaanta,C. Koburger, W. Lange, M. Leach, and S. Luce, Integration ofChemical-Mechanical Polishing into CMOS Integrated CircuitManufacturing, Thin Solid Films, 220 (1992), pp. 1{7.
[9] M. D. Levenson, Wavefront engineering from 500 nm to 100nm CD, in Proceedings of the SPIE - The International Societyor Optical Engineering, vol. 3049, 1997, pp. 2{13.
[10] M. D. Levenson, N. S. Viswanathan, and R. A. Simpson,Improving Resolution in Photolithography with a Phase-Shifting Mask, IEEE Trans. on Electron Devices
[11] L. Liebmann, A. Molless, R. Ferguson, A. Wong, and S. Mans-field, Understanding Across Chip Line Width Variation: TheFirst Step Toward Optical Proximity Correction, in SPIE,vol. 3051, 1997, pp. 124{136.
[12] L. W. Liebmann, T. H. Newman, R. A. Ferguson, R. M. Martino,A. F. Molless, M. O. Neisser, and J. T. Weed, A Comprehen-sive Evaluation of Major Phase Shift Mask Technologies forIsolated Gate Structures in Logic Designs, in SPIE, vol. 2197,1994,
[13] H.-Y. Liu, L. Karklin, Y.-T. Wang, and Y. C. Pati, The Ap-plication of Alternating Phase-Shifting Masks to 140 nm GatePatterning (II): Mask Design and Manufacturing Tolerances,in SPIE Optical Microlithography XI, vol. 3334, Feb. 1998, pp.1-14.
[14] H.-Y. Liu, L. Karklin, Y.-T. Wang, and Y. C. Pati, The Appli-cation of Alternating Phase-Shifting Masks to 140 nm GatePatterning: Line Width Control Improvements and DesignOptimization, in SPIE 17th Annual BACUS Symposium on Pho-tomask Technology, vol. SPIE 3236, 1998, pp. 328{337.[15] Y. Liu, A. Zakhor, and M. A. Zuniga, Computer-Aided PhaseShift Mask Design with Reduced Complexity, IEEE Transac-tions on Semiconductor Manufacturing, 9 (1996), pp. 170{181.
[16] W. Maly, Computer-aided design for VLSI circuit manufac-turability, Proceedings of IEEE, 78 (1990), pp. 356{392.[17] W. Maly, Moore's Law and Physical Design of ICs, in Proc.International Symposium on Physical Design, Monterey, Califor-nia, April 1998. special address.
[18] SEMATECH, Workshop Notes, in 3rd SEMATECH Litho-Design Workshop, Skamania Lodge, February 1996.
[19] A. Moniwa, T. Terasawa, N. Hasegawa, and S. Okazaki, Algo-rithm for Phase-Shift Mask Design with Priority on ShifterPlacement, Jpn. J. Appl. Phys., 32 (1993),
[20] A. Moniwa, T. Terasawa, K. Nakajo, J. Sakemi, and S. Okazaki,Heuristic Method for Phase-Conict Minimization in Auto-matic Phase-Shift Mask Design, Jpn. J.
[21] J. Nistler, G. Hughes, A. Muray, and J. Wiley, Issues Asso-ciated with the

Commercialization of Phase Shift Masks, inSPIE 11th Annual BACUS Symposium on Photomask Technol-ogy, vol. SPIE 1604, 1991, pp. 236{264.
[22] K. Ooi, S. Hara, and K. Koyama, Computer Aided Design Soft-ware for Designing Phase-Shifting Masks, Jpn. J. Appl. Phys.,32 (1993), pp. 5887{5891.
[23] K. Ooi, K. Koyama, and M. Kiryu, Method of Designing Phase-Shifting Masks Utilizing a Compactor, Jpn. J. Appl. Phys., 33(1993), pp. 6774{6778.
[24] G. I. Orlova and Y. G. Dorfman, Finding the Maximum Cutin a Graph, Engr. Cybernetics, 10 (1972), pp. 502{506.
[25] P. Rai-Choudhury, Handbook of Microlithography, Microma-chining, and Microfabrication, vol. 1: Microlithography, SPIEOptical Engineering Press, Bellingham,
[26] F. M. Schellenberg, H. Zhang, and J. Morrow, Evaluation ofOPC E_cacy, in Proc. Intl. Symp. on Aerospace/Defense Sens-ing and Dual-Use Photonics, vol. 2726, 1996.
[27] Alfred Wong, “Resolution enhancement techniques in optical lithography”, SPIE Oct. 2001
[28] M. Rieger, L. Stirniman, “TCAD physical verification for reticle enhancement techniques”, Solid State Technology Vol. 43, No 7, (134) 2000.
[29] W. G. Oldham, S. N. Nandgaonkar, A. R. Neureuther, and M. O’Toole, “A general simulator for VLSI lithography and etching processes: Part I – application to projection lithography,” IEEE Trans. Electron Devices ED-26, No. 4 (717) 1979.
[30] C. A. Mack, “PROLITH: a comprehensive optical lithography model,” Proc. SPIE.
[31] Qi-De Qian, F. A. Leon, “Fast Algorithms for 3D high NA lithography simulation”, Proc. SPIE Vol. 2440 (372) 1995.
[32] Chris A. Mack, Ching-Bo Juang, “Comparison of scalar and vector modeling of image formation in photoresist”, Proc. SPIE Vol. 2440 (381) 1995.
[33] A. Sekiguchi, M. Isono, and T. Matsuzawa, ”Measurement of parameters for simulation of 193 nm lithography using Fourier transform Infrared baking system.” Jpn. J. Appl. Phys. Vol 38 4936 1999
[34] A.B. Kahng and Y.C. Pati, “Subwavelength optical lithography: challenges and impact on physical design”, Proc. ACM intcl. Symp. On Physical Design, pp.112—119, April 1999.
[35] W. Grobman, M. Thompson, R. Wang, C. Yuan, R. Tian and E. Demircan, “Reticle Enhancement Technology: Implications and Challenges for Physical Design”, Proc. ACM/IEEE 38th Design Automation Conference. pp.73- 78 , June 2001.
[36] W. Maly, “Computer-aided design for VLSI circuit manufacturability”, Proc. of IEEE, 78, pp.356—392, 1999.
[37] A. Srivastava, R. Bai, D. Blaauw and D. Sylvester, “Modeling and analysis of leakage power considering within-die process variations”, Proc. Intl. Symp. on Low Power Electronics and Design (ISLPE), pp.64—67, Aug. 2002.
[38] F.M. Schellenberg, L. Capodieci and B. Socha, “Adoption of OPC and the impact on design and layout”, Proc. ACM/IEEE 38th Design Automation Conference. Pp.89 -- 92, June 2001.
[39] TSMC 0.13um CMOS Process Technology

Fundamentals of Web Site Acceleration Performance Starts at the Web Server

By Port80 Software

Abstract

This paper outlines a common sense, cost-effective approach to lowering total cost of ownership and improving Web site and Web application performance according to two simple principles:

- Send as little data as possible
- Send it as infrequently as possible

We will explore "best practice" strategies that can be systematically employed in Web front-end source code and at the origin server in order to achieve performance improvements. These basic strategies, which all avoid expensive hardware solutions in favor of software and business process enhancements, include:

- Code optimization
- Cache control
- HTTP compression

We assume that our typical reader is responsible in some way for development and/or management of a Web site or application running on one or more Windows servers with Internet Information Services (IIS) and that he or she has an interest in improving its performance as much as possible without deploying additional hardware (such as dedicated acceleration appliances) or services (such as Content Distribution Networks).

As we examine each strategy, we will explore the potential benefits to a variety of different Web sites and applications in terms of three vital metrics:

- Faster Web page loads and an improved user experience, translating into higher revenue and increased efficiencies
- Reduction of bandwidth utilization and increased, ongoing savings
- Consolidation of the number of server resources required to deliver existing sites and applications

We will suggest relatively inexpensive software tools that will leverage common Web standards in order to maximize hardware and network resources, thus improving Web site and application performance while lowering the total cost of ownership of Web infrastructure.

Measuring Web Site Performance

The most valuable measurement of Web site performance from an end user's perspective is the amount of time needed to display a requested Web page. The most important metric of page load time is Time to First Byte (TTFB). Defined as the amount of time it takes to deliver the first byte of the requested page to the end user, TTFB represents the visitor's initial confirmation that a Web site or application is responding. Following Time to First Byte is the metric of Throughput, or how many requests can be served by a Web site or application in a given time period. A user expects text, images, and other elements to load swiftly and methodically — failure in any of these metrics results in the perception of poor performance, which can very quickly lead to visitor frustration and abandonment of the site.

With a large enough budget for heavy infrastructure improvements, any server's connection to the Internet can always be improved. However, we are interested in cases in which these common measures of Web site performance degrade due to uncontrollable network conditions and in which expensive, complex hardware solutions are not feasible or desirable. As Web sites and applications grow increasingly complex with bulky code and third party applications, users — many of whom are still using dial-up connections — must download an increasing amount of data to access a Web site or application. Even when highly efficient applications are made available on fast Wide Area Networks (WANs) like corporate extranets, segments of the network will always be susceptible to bottlenecks, and the user may experience unacceptably long page load times. If the Web server's source code and management software are not optimized to keep pace with rising site traffic and application complexity, administrators will waste server resources and bandwidth, and users will be presented with slower, easier-to-abandon Web sites.

Optimizing content and content delivery has been proven to improve Web page delivery speed in case studies conducted by the World Wide Web Consortium and has been espoused by Web optimization experts such as Andy King. Until recently, content optimization required difficult and time-consuming manual coding, but this can now be achieved through the implementation of inexpensive software tools and unobtrusive changes in development and deployment processes. Similarly, while optimizing content delivery once demanded expensive hardware and infrastructure investments, today it can be accomplished with affordable, easy-to-deploy server software tools.

Code optimization, cache control, and HTTP compression are strategies that focus on sending as little data as necessary as infrequently as possible to optimize performance in an existing Web application's front-end code and on the origin Web server delivering that content to Web browsers.

Code Optimization

Source code optimization is an important but often overlooked application of the "send as little data as possible" principle. As a Web site or application acceleration strategy, its chief benefit is that the technique can provide substantial reductions in network payloads without requiring any additional processing on the origin server. Source code optimization should be implemented as a pre-deployment step through which all additions and changes to the front-end source code (including markup, style sheets, and client-side scripts) normally pass before being uploaded to the "live" production site or application.

Beyond White Space Removal

While traditional "white space removal" (the elimination of superfluous spaces, tabs, new lines, and comments) can result in modest savings of 10 to 15 percent in typical HTML and CSS files, a code optimizer modeled after a traditional software compiler can achieve a much higher level of optimization on all text-based files. Because it possesses a site-wide contextual awareness, such a tool can employ more aggressive tactics, including condensing function and variable names, re-mapping lengthy identifiers and URL or URI paths, and through the use of a real JavaScript parser, even streamline lines of code.

Until recently, this approach has been the exclusive domain of JavaScript experts with the time and patience to write very terse code. In addition, hand coding still presented limitations to site expansion and code readability because it maintains a single optimized code base for both development and deployment. It is both safer and more efficient to use next generation development tools like the w3compiler that preserve developer-friendly versions of Web site files as well as the highly optimized versions exclusively for delivery on the Web.

When integrated into the normal developer workflow as a final pre-deployment process, optimizations can be applied to growing Web sites without breaking the external references that abound in modern Web front-ends, including function and variable names defined or initialized in one file and referenced in another. This highly aggressive optimization approach can lead to average savings of 20 to 30 percent in JavaScript files and as high as 70 percent in tested, "real world" cases.

A sensible objection to aggressive source code optimization is that it diminishes the readability of the code when one "views source" in a Web browser. This is a particularly outdated objection as it adds essentially no value for end users and may, in fact, represent a security threat by making it trivially easy for competitors to copy unique client-side features. Even more troubling is the fact that un-optimized source code also clears the way for hackers to conduct reconnaissance on a Web application by "source sifting" for clues to its potential vulnerabilities. Readable and well-commented code is an obvious advantage during initial development, but it is not an appropriate format in which to deliver business-critical Web applications.

Code optimization represents the first step in improving Web site and application performance by reducing the initial network payload that a server must deliver to the end user. Once that content has been optimized on the developer side, attention shifts to how that content can be delivered in its most optimized and efficient form.

Cache Control

What Is Caching? How Does It Apply to the Web?

Caching is a well-known concept in computer science: when programs continually access the same set of instructions, a massive performance benefit can be realized by storing those instructions in RAM. This prevents the program from having to access the disk thousands or even millions of times during execution by quickly retrieving them from RAM. Caching on the Web is similar in that it avoids a roundtrip to the origin Web server each time a resource is requested and instead retrieves the file from a local computer's browser cache or a proxy cache closer to the user.

The most commonly encountered caches on the Web are the ones found in a user's Web browser such as Internet Explorer, Mozilla and Netscape. When a Web page, image, or JavaScript file is requested through the browser each one of these resources may be accompanied by HTTP header directives that tell the browser how long the object can be considered "fresh," that is for how long the resource can be retrieved directly from the browser cache as opposed to from the origin or proxy server. Since the browser represents the cache closest to the end user it offers the maximum performance benefit whenever content can be stored there.

The Common HTTP Request/Response Cycle
When a browser makes an initial request to a Web site — for example, a GET request for a home page — the server normally returns a 200 OK response code and the home page. Additionally, the server will return a 200 OK response and the specified object for each of the dependent resources referenced in that page, such as graphics, style sheets, and JavaScript files. Without appropriate freshness headers, a subsequent request for this home page will result in a "conditional" GET request for each resource. This conditional GET validates the freshness of the resource using whatever means it has available for comparison, usually the Last-Modified timestamp for the resource (if Last-Modified is not available, such as with dynamic files, an unconditional GET will be sent that circumvents caching altogether). Through a roundtrip to the server, it will be determined whether the Last-Modified value of the item on the server matches that of the item cached in the browser, and either a fresh version will be returned from the server with a 200 OK response, or a 304 Not-Modified response header will be returned, indicating that the file in the cache is valid and can be used again.

A major downside to validations such as this is that, for most files on the Web, it can take almost as long to make the roundtrip with the validation request as it does to actually return the resource from the server. When done unnecessarily this wastes bandwidth and slows page load time. With the dominant Internet Explorer browser under its default settings and without sufficient cache control headers, a conditional GET will be sent at the beginning of each new session. This means that most repeat visitors to a Web site or application are needlessly requesting the same resources over and over again.

Fortunately, there is an easy way to eliminate these unnecessary roundtrips to the server. If a freshness indicator is sent with each of the original responses, the browser can pull the files directly from the browser cache on subsequent requests without any need for server validation until the expires time assigned to the object is reached. When a resource is retrieved directly from the user's browser cache, the responsiveness of the Web site is dramatically improved, giving the perception of a faster site or application. This cache retrieval behavior can be seen clearly when a user revisits a Web page during the same browser session - images render almost instantaneously. When the correct cache control headers are incorporated into a server's responses, this performance improvement and reduction of network traffic will persist for any number of subsequent sessions as long as the object is not expired or until a user manually empties his or her cache.

Somewhere Between a Rotting Fish and Burning Currency
Web caching via HTTP headers is outlined in HTTP/1.1, section 13 of RFC 2616. The language of the specification itself helps to define the desired effects and tradeoffs of caching on the Web. The specification offers suggestions for explicit warnings given by a user agent (browser) when it is not making optimal use of caching. For cases in which a browser's settings have been configured to prevent the validation of an expired object, the spec suggests that the user agent could display an image of a stale fish. For cases in which an object is unnecessarily validated (or re-sent, despite its being fresh) the suggested image is of burning currency, so that the user "does not inadvertently consume excess resources or suffer from excessive latency".

Cache control headers can be used by a server to provide a browser with the information it needs to accurately handle the resources on a Web site. By setting values to prevent the caching of highly volatile objects, one can help to ensure that files are not given the "rotting fish treatment." In the case of objects that are by design kept consistent for long periods of time, such as a Web site's navigation buttons and logos, setting a lengthy expiration time can avoid the "burning currency treatment." Whatever the desired lifetime of an object is, there is no good reason why each resource should not have cache-related headers sent along with it on every response.

A variety of cache-related headers are superior to the Last-Modified header because they can do more than merely provide a clue about the version of a resource. They can allow administrators to control how often a conditional GET, or validation trip to the Web server, occurs. Well-written cache control headers help to ensure that a browser always retrieves a fresh resource when necessary or tells the browser how long a validation check can be forestalled. This can dramatically reduce network chatter by eliminating superfluous validations and help keep the pipe clear for essential trips to retrieve dynamic or other files.

The Expires header (for HTTP 1.0) and the Cache-control header with its max-age directive (for HTTP/1.1) allow a Web developer to define how long each resource is to remain fresh. The value specifies a period of time after which a browser needs to check back with the server to validate whether or not the object has been updated.

By using a cache control header inspection tool like this Cacheability Engine, it is easy to see by the Last-Modified header value how long many images, style sheets, and JavaScript files go unchanged. Oftentimes, a year or more has passed since certain images were last modified. This means that all validation trips to the Web server over the past year for these objects were unnecessary, thus incurring higher bandwidth expenditures and slower page loads for users.

Cache Control Does the Heavy Lifting in Web Server Performance

Formulating a caching policy for a Web site — determining which resources should not get cached, which resources should, and for how long — is a vital step in improving site performance. If the resource is a dynamic page that includes time sensitive database queries on each request, then cache control headers should be used to ensure that the page is never cached. If the data in a page is specific to the user or session, but not highly time sensitive, cache control directives can restrict caching to the user's private browser cache. Any other unchanging resources that are accessed multiple times throughout a Web site, such as logos or navigational graphics, should be retrieved only once during an initial request and after that validated as infrequently as is possible.

This begs the question: "Who knows best what the caching policies should be?" On Microsoft IIS Web servers, currently the Web site administrator has sole access to cache control-related headers through the MMC control panel, suggesting that the admin is best suited to determine the policies. More often, however, it is the application developer who is most familiar with the resources on the Web site, and is thus best qualified to establish and control the policies. Giving developers the ability to set cache control headers for resources across their site using a rules-based approach is ideal because it allows them to set appropriate policies for single objects, for a directory of objects, or based on MIME types. An added benefit of developer cache management is less work for site administrators. Try out CacheRight for Microsoft IIS cache control management and centralized cache rules management.

No matter what your Web site or application does, or what your network environment includes, performance can be improved by implementing caching policies for the resources on your site. By eliminating unnecessary roundtrips to the server, cache control can go a long way towards achieving the dual objectives of sending as little as possible as infrequently as possible. However, in cases where resources are dynamic or rapidly changing, a roundtrip to the server is unavoidable for each request. In these situations, the payload can still be minimized by using HTTP compression.

HTTP Compression

Low Cost, High Impact Performance Tuning Should Not Be Overlooked

HTTP compression is a long-established Web standard that is only now receiving the attention it deserves. Its implementation was sketched out in HTTP/1.0 (RFC 1945) and completely specified by 1999 in HTTP/1.1 (RFC 2616 covers accept-encoding and content-encoding). Case studies conducted as early as 1997 by the WC3 proved the ability of compression to significantly reduce bandwidth usage and to improve Web performance (see their dial-up and LAN compression studies).

In the case of HTTP compression, a standard gzip or deflate encoding method is applied to the payload of an HTTP response, significantly compressing the resource before it is transported across the Web. Gzip (RFC 1952) is a lossless compressed data format that has been widely used in computer science for decades. The deflation algorithm used by gzip (RFC 1951) is shared by common libraries like zip and zlib and is an open-source, patent-free variation of the LZ77 (Lempel-Ziv 1977) algorithm. Because of the proven stability of applying compression and decompression, the technology has been supported in all major browser implementations since early in the 4.X generation (for Internet Explorer and Netscape).

While the application of gzip and deflate is relatively straightforward, two major factors limited the widespread adoption of HTTP compression as a performance enhancement strategy for Web sites and applications. Most notably, isolated bugs in early server implementations and in compression-capable browsers created situations in which servers sent compressed resources to browsers that were not actually able to handle them.

When data is encoded using a compressed format like gzip or deflate, it introduces complexity into the HTTP request/response interaction by necessitating a type of content negotiation. Whenever compression is applied, it creates two versions of a resource: one compressed (for browsers that can handle compressed data) and one uncompressed (for browsers that cannot). A browser needs only to accurately request which version it would like to receive. However, there are cases in which some browsers express, in their HTTP request headers, the ability to handle a gzip compressed version of a certain MIME type resource when they effectively cannot. These browser bugs have given rise to a number of third-party compression tools for Microsoft IIS Web servers that give administrators the ability to properly configure their servers to handle the exceptions necessary for each problematic browser and MIME type.

The second factor that has stunted the growth of HTTP compression is something of a historical and economic accident. Giant, pre-2001 technology budgets and lofty predictions of widespread high-bandwidth Internet connections led to some much more elaborate and expensive solutions to performance degradation due to high latency network conditions. Despite the elements being in place to safely implement HTTP compression, Akamai and other Content Distribution Networks employing edge caching technology captured the market's attention. While HTTP compression should not be viewed as a direct alternative to these more expensive options, with the help of a reliable configuration and management tool, compression should be employed as a complementary, affordable initial step towards performance enhancement.

Compression Will Deliver Smaller Web Transfer Payloads and Improved Performance
Most often, HTTP compression is implemented on the server side as a filter or module which applies the gzip algorithm to responses as the server sends them out. Any text-based content can be compressed. In the case of purely static content, such as markup, style sheets, and JavaScript, it is usually possible to cache the compressed representation, sparing the CPU of the burden of repeatedly compressing the same file. When truly dynamic content is compressed, it usually must be recompressed each time it is requested (though there can be exceptions for quasi-dynamic content, given a smart enough compression engine). This means that there is trade-off to be considered between

processor utilization and payload reduction. A highly configurable compression tool enables an administrator to adjust the tradeoff point between processor utilization and compressed resource size by setting the compression level for all resources on the Web site, thereby not wasting CPU cycles on "over-compressing" objects which might compress just as tightly with a lower level setting as with a higher one. This also allows for the exclusion of binary image files from HTTP compression, as most images are already optimized when they are created in an editor such as Illustrator. Avoid the needless recompression of images as this may actually increase their file size or introduce distortion.

Compression results in significant file size savings on text-type files. The exact percentage saved will depend on the degree of redundancy or repetition in the character sequences in the file, but in many cases, individual files may be reduced by 70 or even 80 percent. Even relatively lumpy or less uniform text files will often shrink by 60 percent. Keep in mind that when looking at overall savings on the site, these extremely high compression rates will be counterbalanced by the presence of image MIME types that cannot usefully be compressed. Overall savings from compression is likely to be in the range of 30 to 40 percent for the typical Web site or application. This free online tool will help you to see the file size savings and transfer improvement garnered from compressing any Web resource.

If bandwidth savings is the primary goal, the strategy should be to compress all text-based output. Ideally, this should include not only static text files (such as HTML and CSS), but files that produce output in text media MIME types (such as ASP and ASP.NET files), as well as files that are text-based but of another media type (such as external JavaScript files). This free online tool will help you to gauge the bandwidth cost savings to be realized from compressing any Web resource.

Case studies like this one from IBM have proven the performance improvement that can be realized by using compression, especially in WAN environments that include areas of high traffic and high latency. The general principle is clear: the more crowded the pipe and the more hops in the route, the greater the potential benefits of sending the smallest possible payloads. When the size of the resource is drastically reduced, as a text file is when gzip compression is applied, the number of packets sent is also reduced, decreasing the likelihood of lost packets and retransmissions. All of these benefits of HTTP compression lead to much faster and more efficient transfers.

On Microsoft IIS 4.0 and 5.0 Web servers, httpZip is the best solution for compression as it addresses a number of shortcomings in functionality, customization, and reporting on Windows NT/2000 that gave rise to a third party tools market. With the launch of Windows Server 2003 and IIS 6.0, Microsoft chose to make compression functionality a priority, and their internal IIS 6.0 compression software works — though you must delve into the IIS metabase to manage it beyond a simple "on/off" decision. You can use a tool like ZipEnable to unlock and greatly enhance the configuration and management options for IIS 6.0 built-in compression.

Make Your Web Server a Faster, More Efficient Resource

Having explored these three techniques for Web server performance optimization, how can you put this learning into practice? We have mentioned a few of our Port80 Software tools above, but the following guidelines will help you find the right solutions for your own Web server performance strategy.

Selecting a Code Optimization Tool

An effective source code optimizer should combine an awareness of Web standards with the ability to undertake the aggressive optimizations that can only be done if the entire site is treated as a whole. The capability to fully parse the code is the key here. This is particularly true in the case of JavaScript and DHTML optimizations, in which external references can easily be broken if optimization is done file by file.

Selecting a Cache Control Tool

A cache control tool should provide the ability to implement cache control policies in a rule-based approach so that each resource is cached and validated in the most appropriate and efficient manner. It should also include a mechanism that allows site and application developers to contribute to the generation of those rules without requiring intervention by server administrators.

Selecting a HTTP Compression Tool

An origin server compression solution should be, above all else, highly configurable. Although compression is widely supported by modern browsers, several known bugs remain. Such potential problems are best addressed on the server side by properly configuring exceptions for the MIME types problematic for each specific browser. Similar exceptions may be made for files that are very small or very large, or which, for other specialized reasons, should not be compressed. It is also helpful to be able to adjust the level of compression for different file types and to be able to monitor and log the results of compression.

Strong Performance is the Foundation of Your Web Presence

In order to achieve maximum Web site and application performance, it is vital to view the complete chain — from source code development, server-side processes, and the connection between server and end user, all the way to the end user's Web browser — and to examine each link in that chain for both potential pitfalls and opportunities for improvement. Implementing the three-fold strategy of code optimization, cache control, and compression will dramatically reduce the amount of Web site and application data transmitted and ensures that your data is sent in the most efficient manner possible. Leverage the resources of your existing Web infrastructure with the strategies presented here, none of which are particularly expensive. Implementing these unobtrusive changes in development and deployment practices will aid your Web developers, site administrators, and users. The results will speak for themselves: a more efficient Web server, lower bandwidth expenses, and a faster experience for your users, all helping to drive revenue growth, lower costs, and modernize your Web systems.

Top-Down DSP Design Flow to Silicon Implementation

By AccelChip, Inc

Introduction

We're on the threshold of the next wave of rapid growth in high technology. During the 1970s, we witnessed the proliferation of semiconductors that enabled the digital generation. In the 1980s came the decade of dynamic memories (DRAMs) as semiconductor vendors perfected their manufacturing technologies to allow dramatic increases in memory capacity at previously unheard of prices. The 1990s will be remembered as the era of microprocessors as even the casual consumer became extremely literate about Megahertz and motherboards. And now, as we've entered the new millennium, digital signal processing (DSP) has become the technology of focus with consensus expectations of exponential growth. "Everybody knows that DSP is the technology driver for the semiconductor industry," says Will Strauss, an analyst with Forward Concepts Co., Tempe, AZ.

Why? Because semiconductors, DRAMs and microprocessors have enabled an insatiable appetite for communications in virtually every thing we do. Pervasive computing with an always-on Internet infrastructure that allows immediate access to the information that rules our business world. Voice activated controls enabled by speech synthesis to provide all members of society an equal access to the devices that can improve their quality of life. Wireless communications that keep us in touch no matter where or when we need it. A consumer industry dominated by a plethora of entertainment devices that deliver audio and video to a youthful generation with unprecedented expendable income. Guidance and navigation systems that make our transportation systems safer and more cost effective. And as the world struggles to deal with the terrorism that threatens the safety of our global societies, DSP is key to the effectiveness of the military's command, control and communications systems.

Until now, most DSP designs have been implemented in general-purpose digital signal processors from semiconductor vendors such as TI, Analog Devices and Motorola. These general-purpose processors provided an ideal implementation vehicle for DSP product development teams. General-purpose processors are relatively cheap, are supported by high quality and inexpensive programming tools, facilitate rapid implementation of DSP algorithms, and provide the flexibility development teams prefer for reprogramming during prototyping and debug.

The Need for Speed

The performance requirements of today's electronic systems now exceed the capabilities of general-purpose DSP processors. Simply put, general-purpose DSP processors are running out of gas. Figure 1 shows the projected DSP algorithm performance requirements driven by the broadband networking market versus the available performance capacity of general-purpose DSP processors over the next 3 years.

The gap between the performance capacity of general-purpose DSP processors and the requirements of new broadband communication technologies widens exponentially over the next several years. The only alternative to general-purpose DSP processors previously available to DSP developers was to cast their DSP algorithms into an ASIC for hardware acceleration. An ASIC solution has proven to be less than desirable though. DSP algorithms implemented in ASICs sacrifice the flexibility of reprogrammability, command a hefty non-recurring engineering (NRE) fee, require a lengthy manufacturing cycle for initial prototypes, and mandate the purchase of expensive IC design tools.

With the introduction of advanced FPGA architectures such as the Xilinx Virtex-II and the Altera Stratix-II devices, a new hardware alternative is available for DSP designers combining all the benefits of general-purpose DSP processors with the performance advantages of ASICs. These new FPGA architectures were created to optimize DSP implementations and provide the necessary computing resources to meet the performance needs of today's electronic systems. An advantage of FPGAs is they allow the DSP designer to "fit the architecture to the algorithm" – that is, the designer can implement as many parallel resources inside the FPGA as necessary to realize the performance required of the system. In general-purpose processors, the resources are fixed as each processor contains a finite number of basic computing functions such as multiply accumulators (MAC). Thus, in a general-purpose DSP processor, the designer must "fit the algorithm to the architecture" and the required performance is not obtainable as in an FPGA.

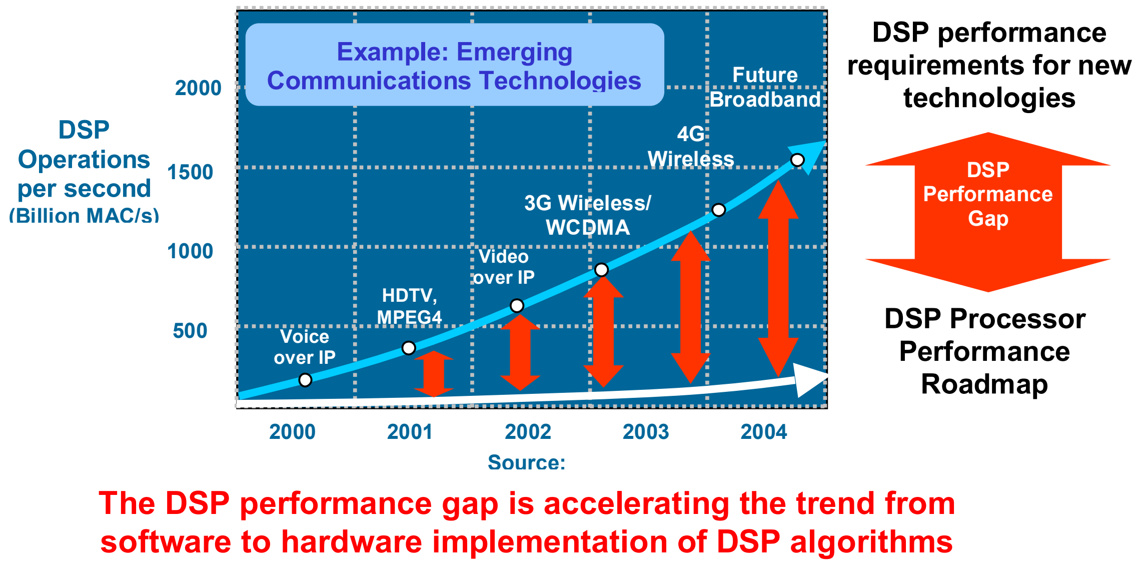


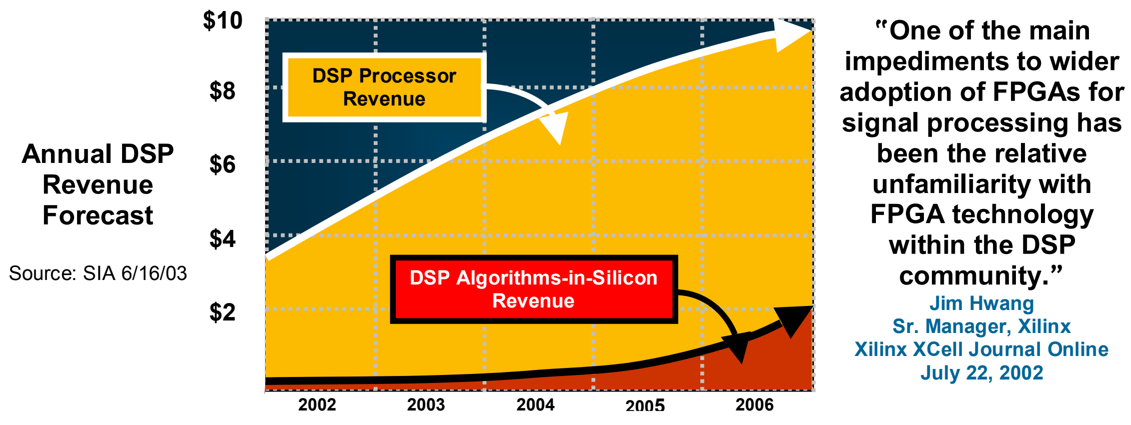
Figure 1 - DSP Performance Requirements vs. General-Purpose DSP Processors (source: Xilinx)

Semiconductor Industry Bright Spot

Figure 2 shows the estimated annual revenue for the total DSP market, and for the algorithm-in-silicon market subsection that includes FPGAs, structured ASICs and ASICs. The DSP algorithm-in-silicon market will grow at a compound annual growth rate (CAGR) of greater than 42% and represents one of the largest growth segments for all semiconductors in the next 3 years. The challenges that now face the DSP design community are the same ones that ASIC designers battled in the 1990s. What changes do DSP development teams need to make to their design methodologies to target FPGAs instead of general-purpose DSP processors? How do DSP design teams develop the required new design skills efficiently? How can companies afford to completely retool their design flows? How can design teams simultaneously champion a new way of implementing DSP algorithms without jeopardizing current product development schedules? And perhaps most importantly, what can managers do to minimize the risk of making decisions that could produce catastrophic results?

Change needs to be managed carefully.
Rapid growth in every new technology sector has required the introduction of a new design methodology to dramatically increase engineering productivity. Semiconductors flourished with the introduction of IC CAD systems by GE Calma in the late 1970s. DRAM technology exploded in the 1980s when process and transistor-level simulation tools let device physicists accurately model the behavior and manufacturing of the basic memory cell. And microprocessors, ASICs and FPGAs hallmarked the 1990s as logic designers put aside their proven schematic capture tools and took a leap of faith to top-down design flows based on logic synthesis and the Verilog and VHDL design languages.

AccelChip believes the future of digital signal processing is also dependent on the adoption of a new design methodology that will allow companies to meet the DSP market's demands in a timely and cost-effective manner. Like the ASIC and FPGA generation before, the answer for the DSP revolution is a true top-down design process.



"One of the main impediments to wider adoption of FPGAs for signal processing has been the relative unfamiliarity with FPGA technology within the DSP community."
Jim Hwang
Sr. Manager, Xilinx
Xilinx XCell Journal Online
July 22, 2002

Figure 2 - DSP Forecasted Worldwide Chip Shipments ($ millions) (source: Forward Concepts)

The Way We Were

DSP design has traditionally been divided into two types of activities – systems/algorithm development and hardware/software implementation. These tasks have been accomplished by two very disparate groups of engineers that often have little connection or interaction. In larger companies, it is quite common for these two groups to be physically separated with the interface between the two groups consisting of a detailed written specification that includes block diagrams, mathematical formulas and waveforms showing the expected output based upon the systems inputs.

Algorithm developers create, analyze and refine the required DSP algorithms using mathematical analysis tools at the behavioral level without regard to the underlying system architecture or hardware/software implementation details. The system designer is concerned with defining the functionality and architecture of the design to adhere to the product specification and interface standards. Systems designers and algorithm developers interact efficiently with each other because they work in a common design environment based on a high-level programming language. The majority of DSP system designers and algorithm developers use the MATLAB language from The MathWorks. MATLAB has proven to provide the most productive and accurate DSP development environment because of its intuitive user interface, built-in math and graphics functions, and strong programming language. "Each time we attempted to code routines in C, we encountered subtle numerical problems and were not able to match the numerical performance of the MATLAB code," said Jack Staub of Hughes Aircraft. MATLAB also is used extensively in universities and is often the first design language engineering students are taught, which minimizes the learning curve for companies who are continually adding new engineers to their staff.

Hardware/software design teams take the specifications created by the systems engineers and algorithm developers and are tasked to create a physical implementation of the DSP design. Typically, the specification is divided into small modules and distributed to individual members of the implementation team who first must gain an understanding of the functionality of their module. If the target of the DSP algorithm is an FPGA, structured ASIC, ASIC or SOC, the first task is to create a register transfer level (RTL) model in a hardware description language (HDL) such as Verilog or VHDL. The implementation engineer is required to know communications theory and signal processing to be able to interpret the written specification provided by the systems engineer. The process of creating an RTL model and a simulation testbench usually takes about 1 to 2 months because of the need to verify the manually created RTL file exactly matches the MATLAB model.

Once the RTL model and simulation environment is created, the implementation engineer interacts with the systems engineers and algorithm developers to analyze the performance, area and functionality of the hardware realization of the DSP system. It is quite common that the original algorithms and system architecture need to be modified because the systems engineers had no visibility into the physical design domain during their algorithm development. In other words, the systems engineers rely on experience, intuition and luck when they develop the system specification and as can be expected, often a good deal of refinement is required. The iteration process continues – refine the algorithms and system architecture, update the written specification, modify the RTL models and testbenches, and resimulate – until the DSP system requirements are met by the hardware realization. The implementation team then executes a standard FPGA/ASIC top-down design flow using logic synthesis to map the RTL model into a gate-level netlist and physical design tools to place and route the netlist in a given FPGA or ASIC device.

Figure 3 shows the basic DSP algorithm-in-silicon design process consisting of separate design domains for algorithm development and hardware implementation. As discussed above, the lack of a link between the two design domains delays the process of design exploration until after the lengthy process of manually creating RTL models based on a written specification. Perhaps a larger concern with this design process is that the physical design of the DSP algorithms is based upon interpretation by the hardware engineers of the written specification. The lack of DSP expertise of the hardware engineers, coupled with the risk of an incomplete or ambiguous specification, often leads to catastrophic results due to misinterpretation of the required functionality. With the increase in DSP complexity, the chance of an error occurring in the manual generation of the RTL models is now the rule, no longer the exception. Thousands of man-hours are wasted doing exhaustive logic simulations to ensure the correct interpretations were made during the RTL model generation. Many times the errors are not caught in simulation because the same misinterpretations are written into the simulation testbenches, thus the hardware design errors are not found until the design reaches a prototype stage.

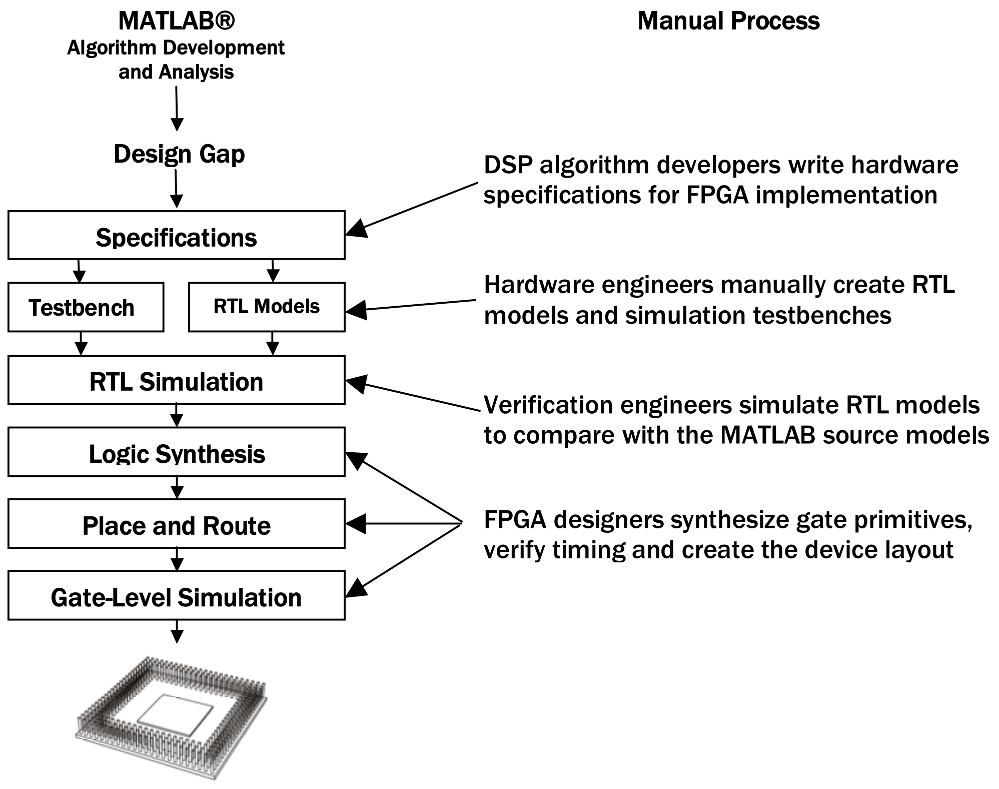

MATLAB®
Algorithm Development and Analysis
Manual Process
Design Gap
DSP algorithm developers write hardware specifications for FPGA implementation
Specifications
Testbench
RTL Models
Hardware engineers manually create RTL models and simulation testbenches
RTL Simulation
Verification engineers simulate RTL models to compare with the MATLAB source models
Logic Synthesis
Place and Route
FPGA designers synthesize gate primitives, verify timing and create the device layout
Gate-Level Simulation

Figure 3 - Traditional DSP Design Environment

One of the significant benefits that ASIC/FPGA designers realized when they moved to a true top-down design methodology was improved design data management. When ASICs and FPGAs were designed bottom-up as DSP designs are done today, many errors were introduced due to the lack of a single, golden source for the design data. In DSP design today, it is incumbent on the disparate design teams to keep their MATLAB models synchronized with the manually created RTL models and testbenches. As mentioned, these two groups have very little interaction and are usually geographically dispersed so the task of managing the design data becomes daunting. CoWare proposes a solution to the model synchronization problem in their Signal Processing Worksystem (SPW) tool suite by introducing the concept of "Simulation-Aided Design Methodology for Transition from Specification to Implementation". In this methodology, CoWare proposes that DSP design teams create an executable specification using their Hardware Design System (HDS) with a library of DSP hardware models instead of a programming language that will allow simulation to replace interpretation of the DSP specification and algorithms. While this methodology has merits in eliminating the misinterpretations hardware engineers make in developing RTL models, it falls well short of ensuring design data synchronization. It is still the responsibility of the two disparate design organizations to manually edit their models with every modification to the executable specification, and the likelihood of errors occurring is quite high especially given the increased complexity and time to market pressures of today's projects.

A True Top-Down DSP Design Methodology

AccelChip is the only company that enables a true top-down DSP design methodology for DSP algorithm developers. The AccelChip DSP synthesis tool directly reads in MATLAB models and automatically outputs synthesizable RTL models and simulation testbenches in VHDL or Verilog. By linking the two design domains of DSP, AccelChip provides DSP design teams a significant reduction in design labor and time, elimination of misinterpretations and costly design rework, automatic verification of the hardware implementation, and the ability of systems engineers and algorithm developers to perform architectural exploration in the early phases of their development cycle. Figure 4 shows a true top-down DSP design flow for FPGA implementations using AccelChip.

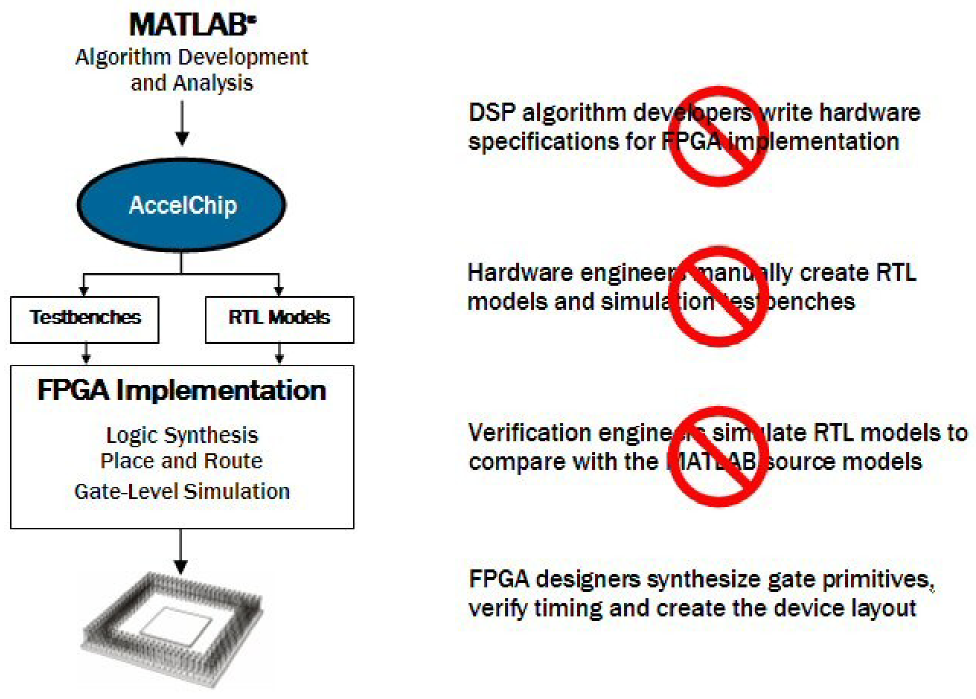

MATLAB
Algorithm Development and Analysis
AccelChip
Testbenches
RTL Models
FPGA Implementation
Logic Synthesis
Place and Route
Gate-Level Simulation
DSP algorithm developers write hardware specifications for FPGA implementation
Hardware engineers manually create RTL models and simulation testbenches
Verification engineers simulate RTL models to compare with the MATLAB source models
FPGA designers synthesize gate primitives, verify timing and create the device layout

Figure 4: True Top-Down DSP Design Process

AccelChip eliminates the need for hardware engineers to manually create RTL models and testbenches, shaving months off the development cycle and reducing the number of engineers required to produce a hardware implementation. The RTL models that are created by AccelChip are "architecture aware" of the target FPGA device and thus are not "generic RTL" models. AccelChip uses a Resource Description Language (RDL) to provide awareness of the resources available inside each type of FPGA. When the RTL model is created, the AccelChip high-level synthesis tool will create an optimal implementation for logic synthesis, ensuring the resultant gate-level netlist takes full advantage of the FPGA device. For example, DSP algorithms implemented in one supplier's FPGA will be significantly different in performance and area than in another's because the architecture, logical resources and routing methodologies are different for the devices. By being *architecture aware*, AccelChip produces the best physical implementation for the FPGA device the DSP design team is targeting.

By providing an easy to use, automated path directly from MATLAB to silicon implementation, AccelChip enables DSP systems engineers and algorithm developers to refine their algorithms early in the development cycle based on design exploration. Algorithm developers can quickly convert their MATLAB designs into a gate-level netlist for a target FPGA device to assess the tradeoffs in performance, area, cost and power. Having the feedback from the physical implementation of the algorithms early and often during the development cycle means fewer iterations are made later in the design process, again saving significant time and labor.

In Summary

Digital Signal Processing is one of today's most important areas of technology development, driven by the needs of the communications, consumer, defense and transportation industries. The performance requirements of DSP algorithms are outstripping the capabilities of general-purpose DSP processors, causing DSP implementation teams to seek hardware solutions. FPGAs provide the ideal platform for DSP implementation, combining the reprogrammability, architectural flexibility, and system-level integration of general-purpose processors with the performance offered by customizable hardware.

With the introduction of the AccelChip high-level synthesis tool, DSP design teams now have a tool that enables the deployment of a true top-down DSP design process. AccelChip directly links the most commonly used DSP design langugage – MATLAB – with proven HDL-based FPGA design flows, resulting in increased productivity, faster time-to-market, and improved product functionality through design space exploration early and often during the development process. AccelChip seamlessly plugs into the designer's existing DSP environment to ensure minimal risk for management in transitioning to a true top-down DSP design methodology. AccelChip does not require the implementation of a new design language, and by providing an intuitive user interface it requires only a minimal learning curve for DSP designers. The AccelChip DSP synthesis solution allows DSP design teams to realize the full potential of their DSP algorithms in silicon.

Upcoming Advanced & Educational Trainings

上海浦东新区后生集成电路培训中心高级培训
Migrating from ASIC/SoC to Structured Design (7/10 ~ 7/11)

全国第一家以集成电路命名的上海浦东新区集成电路培训中心，诚邀您参加此高级研修班，亲身感受业界领先的技术，分享我们的过去和现在的实战经验，共同探讨对将来的展望。培训具体内容附后，也可详见相关网页（http://www.hwswworld.com）。服务您是我们的荣幸！您的成功就是我们的骄傲！

我们的培训客户遍布全国，包括世界五百强在华企业，台资企业，国内软件/集成电路百强及各著名高校等。另外除每月的公开课外，我们也可以为您提供内训. 我们的内训客户包括北京大唐微电子、上海朗讯及苏州摩托罗拉等。具体合作流程也可详见相关网页。

时间：7/10 ~ 7/11
地点:上海浦东新区郭守敬路 498 号浦东软件园 22217-22219 座

费用：包括授课, 工作西餐, 英文教材及英文讲义。

报名电话(Tel): 021-5080-2811, 5080-2813, 5131-4066, 5131-4177 传真 (Fax)：021-5131-4176

E-mail: training@hwswworld.com

开户银行：工行科苑支行 收款人：上海后生微系统有限公司 帐号：1001194909026168183

报名回执 （请传真回执及汇款回单以确认报名，一经确认，恕不退款，强化班取消除外）

单位/人数	电子邮件
姓名	付款方式 （电汇或贷记凭证）
电话/传真	是否需要预定浦东软件园招待所（150 元三星级标间，费用自理）？

具体报名流程：

1．填写上述报名回执表格传真给我们；

2．按填写的付款方式将报名费用转至我们的帐上；

3．将付款付款凭证的复印件传真给我们；

4．我们发报名确认传真给您；

5．发票在培训的第一天上午给

Program Agenda (Subject to Change without Notice)

Program Agenda (Subject to Change without Notice)

1. Structuctured ASIC Introduction
1.1 Why SoC Cannot Solve Problems that both ASIC and FPGA Cannot Solve: cost, TTM, Capacity, Performance, Power & IP Reuse?
1.2 The Gap between FPGA & ASIC
1.3 Structured ASIC's for Mid-Volume Designs
1.4 The Similarity & Differnece bween Structured ASIC Design & FPGA Design

2. Structured ASIC Properties
2.1 Low NRE cost
2.2 High performance
2.3 Low power consumption
2.4 Less Complex Manufacture Process
2.5 Shorter TTM

3. Structured ASIC Architecture
3.1 Two Main Hierarchical Levels
3.1.1 Structured Elements
3.1.2 Array of Structured Elements
3.2 The Granularity of Architecture: Fine Granularity Vs Medium Granularity

4. Structured ASIC Design Flow
4.1 RTL Design
4.2 Logical synthesis: Logic Optimization & Technology Mapping
4.3 DFT
4.4 CTS
4.5 Physical Synthesis: Placement & Routing
4.6 ASIC Flow Vs Structured ASIC Flow
4.7 FPGA Flow Vs Structured ASIC Flow
4.8 The Impact of Architecture Quality on Design Flow
4.9 The Impact of Library Quality on Design Flow
4.10 Supports from EDA Tool Vendors
4.11 Supports from FAB

5. Issues on Migrating from ASIC Design Structured ASIC Design & a Case Study

6. Issues on Migrating from SoC Design Structured ASIC Design & Case Studies

上海后生微系统有限公司

Advanced J2EE for Web Services (7/13 ~ 7/14)

上海后生微系统有限公司，诚邀您参加此高级研修班，亲身感受业界领先的理念，分享我们的过去和现在的实战经验，共同探讨对将来的展望。培训具体内容附后，也可详见相关网页（http://www.hwswworld.com ）。服务您是我们的荣幸！您的成功就是我们的骄傲！我们的培训客户遍布全国，包括世界五百强在华企业，台资企业，国内软件/集成电路百强及各著名高校等。我们的代表客户及合作伙伴详见网页。除每月的公开课外，我们也可以为您提供内训. 我们的内训客户包括沈阳东软、广西平方软件、上海朗讯等。具体合作流程也可详见相关网页。另外我们新近创办了英文高级技术月刊 System Design Frontier，并推出了在线商店 System Design Mall（其中有不少英文原版书, 另外你还可以在线报名），欢迎在线订阅、在线购买和在线报名。

时间: 7/13 ~ 7/14

地点:上海浦东新区郭守敬路 498 号浦东软件园 22217-22219 座

费用：包括授课, 工作西餐, 英文教材及英文讲义。

报名电话(Tel): 021-5080-2811, 5080-2813, 5131-4066, 5131-4177

传真 (Fax)：021-5131-4176　E-mail: training@hwswworld.com

开户银行：工行科苑支行 收款人：上海后生微系统有限公司 帐号：1001194909016268183

报名回执 （请传真回执及汇款回单以确认报名，一经确认，恕不退款，强化班取消除外）

单位/人数	电子邮件
姓名	付款方式 （电汇或贷记凭证）
电话/传真	是否需要预定浦东软件园招待所（150 元三星级标间，费用自理）？

具体报名流程：

1．填写上述报名回执表格传真给我们；

2．按填写的付款方式将报名费用转至我们的帐上；

3．将付款付款凭证的复印件传真给我们；

4．我们发报名确认传真给您；

5．发票在培训的第一天上午给

Program Agenda (Subject to Change without Notice)

1. J2EE Best Practices
1.1 J2EE Life Cycle Overview
1.1.1 Design for Change With Dynamic Domain Model
1.1.2 Use a Standard Modeling Language
1.1.3 Recycle You Resources
1.2 J2EE Best Practice in Development Phase
1.2.1 Use Proven Design Patterns
1.2.2 Automate the Build Process
1.2.3 Integra of Ten
1.2.4 Optimize Communication Costs
1.3 J2EE Best Practice in Test Phase
1.3.1 Build Test Casesfirst
1.3.2 Create a Testing Framework
1.3.3 Automate Testing
1.4 J2EE Best Practice in Deployment Phase
1.4.1 Use J2EE Standard Packing Specification
1.4.2 Use Tools to Help in Deployment
1.4.3 Back Up Your Production Data and Enviponment
1.5 J2EE Best Practice in Optimization Phase
1.5.1 Build a Performance Plan
1.5.2 Manage Menory and Plug Leaks
1.5.3 Focus on Priorities
1.6 J2EE Best Practices When Choosing Environments
1.6.1 Do Not Estrict Deployment Options at Design Time
1.6.2 Create a Responsive Development Enviponment

2. Introducing Web Services
2.1 Challenges Presented by Existing Systems
2.2 Web Enabled Components
2.3 Evolution of Distributed Computing
2.4 Problems with Traditional Distributed Computing
2.5 Service-Oriented Architecture and XML Web Services
2.6 The Web Technology Stack (XML, HTTP, SOAP, WSDL, UDDI, etc.)
2.7 The Web Services Marketplace
2.8 Common Web Services Scenarios
2.9 Live Web Services Demonstration

3. Web Services Architecture and Standards
3.1 Introduction to Service-Oriented Architecture
3.2 XML Web Services and Service-Oriented Architecture In a Nutshell
3.3 How to blend Object-Oriented with Service-Oriented Architectures
3.4 Key XML and Web Services Technologies In Depth
3.5 J2EE Web Services Vs .NET Web Services
4. XML Web Services Technical Primer
4.1 HTTP and SMTP Basics
4.2 XML Technologies in a Nutshell: XML, XSL/XSLT, XML Schema, XML Namespaces
4.3 Serializing XML over HTTP
4.4 SOAP, WSDL, and UDDI Fundamentals
5. Digging In Deeper
5.1 The Simple Object Access Protocol

5.2 SOAP Architecture
5.3 Applying SOAP
5.4 The Web Services Description Language
5.5 The WSDL Description Model
5.6 Utilizing WSDL
5.7 The Universal Description, Discovery and Integration project
5.8 UDDI Information Model
5.9 The UDDI RoadMap
6. Pulling it All Together: SOAP, WSDL, UDDI
6.1 Defining and Implementing the Service Interface
6.2 Generating and Publishing the Service Description
6.3 Discovering the Service Description
6.4 Defining the Client
6.5 Invoking the Service
7. Java Web Services Developer's Pack (JWSDP) Introduction
7.1 JWSDP Component Overview
7.2 JWSDP API Overview
7.3 JWSDP Development Process
7.4 Setting up the JWSDP
8. JAX-RPC Development
8.1 Java API for XML-based RPC (JAX-RPC)
8.2 JAX-RPC Architecture
8.3 The SAAJ API
8.4 JAX-RPC Data Types
8.5 JAX-RPC Development and Deployment Process
8.6 The xrpcc Tool
8.7 Dynamic JAX-RPC Development
9. JAXR Development
9.1 Java API for XML Registries (JAXR)
9.2 JAXR Architecture
9.3 JAXR API
9.4 JAXR Tools
10. JAXM Development
10.1 Java API for XML Messaging (JAXM)
10.2 JAXM vs JAX-RPC
10.3 JAXM Architecture
10.4 JAXM and the SAAJ API
10.5 JAXM Messaging Models
10.6 JAXM Development and Deployment Process
10.7 Advanced JAXM Topics
11. JAXP Development
11.1 Java API for XML Processing (JAXP)
11.2 JAXP and Web Services
12. Advanced Web Services Topics
12.1 Web Services Security
12.2 Managing Web Services Transactions
12.3 Stateful Web Services
12.4 EJB, JMS, and Web Services
12.5 J2EE Web Services Tools Overview
13. Where is Web Services Going?
13.1 The Death of the Browser?
13.2 Services on Demand
13.3 Grid Computing

上海浦东新区后生集成电路培训中心 教育培训

集成电路正向设计出国与就业暑期指导班招生简章 (7/30~8/2)

出国和就业——谁动了你的奶酪?

简介:

我国集成电路产业经过30多年的发展，初步形成了由7个芯片生产骨干企业，十几个封装厂，近百家设计公司（中心），以及若干个关键专用材料和设备制造厂构成的产业群体，但与国际先进国家相比，总体水平还比较落后。究其原因，专业人才的严重缺乏成为阻碍我国集成电路产业发展的重要瓶颈之一。集成电路产业对国民经济和其他产业的巨大影响力有目共睹，而如何提高中国自身集成电路设计能力已关乎到国家的核心竞争力

特色:
- 不仅在技术上引你入门，更给予你选择正确人生道路的启迪
- 以主流厂商 FPGA器件为IC设计的切入点，深入介绍RTL级正向设计及综合技术
- 以业界最流行的硬件描述语言Verilog为正向设计语言，贴近工业实际
- 不是来自学院派的陈旧理论，而是源自工业界的实战演练
- 小班授课，限额10名，对考核合格者颁发上海浦东新区后生集成电路培训中心结业证书
- 设有丰富实验环节（50%为实验时间），并设有专门讨论时端
- 采用中英文授课及讨论，使用英文教程及英文讲义
- 对优秀学员提供可能的推荐就业机会及提供可能的实习机会
- 免费参加为期一天的全额奖学金留学美加五一长假指导班

招生对象：所有有志于集成电路设计者，具有电子工程或计算机专业背景更佳。

收费标准：费用包括讲课，实验，现场讨论, 工作午餐，英文教材及英文讲义，全日制在校生凭有效身份证及学生证享受半价优惠。只针对个人，不接受公司报名。

报名联系办法:
电话（周一至周五）：021-5080-2811，5080-2813, 5131-4066, 5131-4177
传真：021-5131-4176 E-mail: training@hwswworld.com

上课时间及地点:
7/30，7/31，8/1，8/2（上午九点至下午四点），合计 24 学时。在浦东软件园 22217-22219 座

开户银行：工行科苑支行 收款人：上海浦东新区后生集成电路培训中心 帐号：1001194909026460463

报名回执 （请传真回执及汇款回单以确认报名，一经确认，恕不退款，强化班取消除外）

单位/人数	电子邮件
姓名	付款方式 （电汇或贷记凭证）
电话/传真	是否需要预定浦东软件园招待所（150 元三星级标间，费用自理）或科技部上海韩国语培训中心宿舍（50 元 5 人间，费用自理）？

具体报名流程：

1．填写上述报名回执表格传真给我们，也可登陆我们的 System Design Mall 网站在线报名；

2．按填写的付款方式将报名费用转至我们的帐上；

3． 将付款付款凭证的复印件传真给我们；

4． 我们发报名确认传真给您；

5．发票在培训的第一天上午给

Program Agenda (Subject to Change Without Notice)

1. Opening Remark
2. RTL Design under From Algorithm to System Context
2.1 Design Flows: FPGA, ASIC & COT
2.2 Design Views in Y Chart: Algorithmic, Structural & Physical
2.3 Migrating from ASIC Flow to COT Flow: What Is It & How It Can Be Done?
2.3 Computation Models: Control-flow, Data-flow & Hybrid
2.4 Timing Models: Untimed Vs Timed (Cycle-based & Discrete Event)
2.5 Algorithm Design, Specification & Optimization in HLL: C, C++, Java & MatLab
2.6 Behavioral/ Transaction-level Design, Specification & Optimization in System-level Design Language: System C & System Verilog In Depth
2.7 HW/SW Co-design: Partitioning, Co-verification & Co-validation
2.8 EDA Flow Setup, Validation & Turning with Scripting Languages: Tcl & Perl In Depth
2.9 Verification & Validation for RTL Design – Solutions to Verification Challenges
2.10 Bridging Logic Design to Physical Design – Solutions to Timing Closure Challenges
2.11 On RTL Design Frontier: High Speed, Low Power Design 2.12 IP-based SoC Design Reuse: Never Reinvent Your Wheel
3. Verilog at RTL – Part I
3.1 Digital Design Strategies & Techniques
3.2 Verilog Syntax Basics
3.3 Hierarchical Modeling Concepts in Verilog
3.4 Behavioral Subset of Verilog
3.5 The Parameterized Design – The 1st Step for Design Reuse
3.6 Verilog Coding Styles
4. FPGA Design: The State-of-the-Art
4.1 FPGA, ASIC & SoC
4.2 From Algorithm to System & m Architecture to Layout
4.3 Design Flows: FPGA Flow, ASIC Flow & COT Flow

4.4 Migrating FPGA Designs to ASIC/SoC Designs
4.5 From PAL, PLD to FPGA
4.6 Field Programmable Elements: Logic, Register, Interconnect & I/O
4.7 FPGA Architecture Variations: Table-Look-Up Vs. MUX-based
4.8 FPGA Architectures at a Glance: Altera, Xilinx & Actel
4.9 FPGA Applications in a Nut Shell: Prototyping, Emulation & Production Usage
4.10 From Field Programmable (Configurable) to Reconfigurable: Hope or Hyper?
5. Verilog at RTL– Part II
5.1 Structural Subset of Verilog
5.2 Useful Modeling Techniques in Verilog
5.3 Timing & Delays in Verilog
5.4 Verilog PLI
5.5 Verilog Test Fixtures
6. The Essence of RTL Design
6.1 Temporal Domain & Spatial Domain of HW
6.2 Scheduling, Resource Allocation and Binding
6.3 Controller & Data path
6.4 Controller: Hardwired Vs. Programmable
6.5 FSM: Mealy Machine Vs. Moore Machine
6.6 Data path: Regular Vs. Irregular
6.7 Build High Performance Data path
6.8 Memory Architecture In Depth: On-chip Flash, ROM, DRAM, SDRAM & CAM
6.9 Bus Architecture In Depth: High Performance Vs Low Cost
7. RTL Design Using ASM
7.1 On the ASM Chart
7.2 ASM Notations
7.3 The Theory behind ASM
7.4 Scheduling, Resource Allocation & Binding
7.5 Advanced ASM Techniques
7.6 The Trade-off between Mealy Machine & Moore Machine
7.7 The Trade-off between Data path & Controller
7.8 RTL design in the SoC Context - When You Have to Deal with 3rd-Party IP)
7.9 Mini-Case Study: Designing an SRA (Square Root Approximation) Block
8. Open Discussion

Lab 1 Design an N Bit ALU in Behavioral View

1. Hard-wired Controller based RTL Design with ASM for Speed & Cost
1.1 The Cycle-based Delay Model
1.2 Multi-cycle, Chaining and Pipelining
1.3 Latency Vs. Throughput
1.4 Pipelining, Currency & Super scalar
1.5 Architecture-level Trade-off between Speed & Cost
2. RTL Design In Depth
2.1 Data Representation: Issues on Floating to Fixed-Point Optimization
2.2 Combinatorial Building Blocks: MUX, Decoders, Encoders & Priority encoders
2.3 Clocked Building Blocks: Edge-based Design Vs Latch-based Design

2.4 Interconnection Structures: Bus-based Vs MUX-based
2.5 Arithmetic Building Blocks: Custom Vs Synthesis-based
2.6 Data path Functions: Organization and types of comparators, counters and ALUs
2.7 Finite State Machines: State Encoding &State Minimization
2.8 Memory structures: Basics, synchronous and dual port RAM, LIFO and FIFO structures
2.9 Partitioning Issues
3. Verilog at RTL – Part III
3.1 Verilog Hierarchy
3.2 Built-In Logic Primitives
3.3 User-Defined Primitives (UDP)
3.4 Library Parameterized Modules (LPM)
3.5 Latches and Flip-flops
3.6 Blocking and Non-blocking Assignments
3.7 Miscellaneous Verilog Modeling Tricks
4. Verilog at RTL – Part IV
4.1 Simulation View Vs Synthesis View
4.2 Synthesizable Subset of Verilog
4.3 Synthesizable Subset of Verilog
4.4 Verilog Synthesis Styles
4.5 RTL Coding Styles for Verilog
4.6 Design for Performance/Cost Trade-off
4.7 Design for Design Reuse
5. Real Life Digital Design Strategies and Techniques
5.1 Synchronous Logic Rules
5.2 Clock Strategies
5.3 Design for Test Issues
5.4 Area/Delay Optimization
6. Case Study: Design a Pipelining FFT (Fast Fourier Transform) Block with ASM
7. Open Discussion

Lab2 Design a N Bit Shift-Subtract-based Division Machine based on N Bit ALU

1. Issues in Real Life Digital Design
1.1Verilog Hierarchy Revisited
1.2 Tri-state Signals and Buses
1.3 Reset, Preset, Tri-state and Bi-directional Signals
1.4 Priority Encoders
1.5 Area/Speed Optimization in Synthesis
1.6 Trade-off between Operating Speed and Latency
1.7 Delays in FPGA Elements
1.8 Design Partitioning
1.9 Scalable and Parameterized Design
2. Programmable Controller based RTL Design with ASM
2.1 MCU, CPU & DSP
2.2 CISC, RISC, Super Scalar & VLIW
2.3 The Impact of ISA on Programmable Micro architecture
2.4 The Demands of Programmable Micro architecture on Compiler

2.5 Controller Design for Programmable Micro architecture
2.5.1 The Design of Custom ISA
2.5.2 The Design of Sequencer
2.5.3 On Decoding Logic
2.5.4 The Design of Custom Data path
2.5.5 Memory Architecture Issues
2.5.6 Bus Architecture Issues
3. Building Blocks in Real Life Digital Design
3.1 Decoders and Encoders
3.2 Registers and Latches
3.3 Adders and Sub tractors
3.4 Multipliers and Dividers
3.5 Counters and Simple Arithmetic Functions
3.6 Finite State Machines Revisited
3.7 Register Files
3.8 ROM, RAM
FIFO and UART
4. Verilog at RTL – Part V
4.1 Simulation View Vs Synthesis View – How to Deal with Mismatches?
4.2 Synthesizable Subset of Verilog
4.3 RTL Coding Styles for Verilog: Synthesizability & Reusability 4.4 Design for Performance/Cost Trade-off: Partitioning & Timing Budgeting Verilog Hierarchy Revisited
4.5 Specific Design Issues
4.5.1 On Dealing with Long Wires
4.5.2 Gated Clock Vs Clock Enable
4.5.3 On Glitch Elimination Vs Redundancy Removal
4.5.4 Asynchronous Design and Metastability
4.5.4.1 Asynchronous Design Basics
4.5.4.2 Metastability and Mean Time Between Failure (MTBF) Calculation
4.5.4.3 On Handling Asynchronicity
4.5.4.4 On Multi-clock domain interfacing
4.6 Trade-off between Operating Speed and Latency
4.7 Scalable and Parameterized Design – The 1st Step for IP-based SoPC (System-on-a-Programmable Chip) Design
5. Migrating from FPGA Design to ASIC Design
5.1 On Design-For-Testability (DFT)
5.2 On Clock-Tree-Synthesis (CTS)
5.3 Timing Closure Issues
5.4 Interaction with the Fab
6. Case Study: Design CISC (PDP-8 subset) & RISC (ARM subset) CPU Using ASM
6. Libraries, Reusable Modules and IP
6.1 The Importance of Library Elements
6.2 Design Reuse and Design for Reuse
6.3 The Reuse Methodology Manual
6.4 IP based FPGA Design – LPM & Beyond
7. Open Discussion

Lab 3 Design a N Bit Division Machine with read FIFO IP & Write FIFO IP (LPM) in Non-Pipelined Mode

上海浦东新区后生集成电路培训中心教育培训

基于 ARM 的嵌入式软件设计出国与就业 暑期提高班 (7/9~7/12)

出国族 - TOEFL 和 GRE 以后的下一步是什么？ 都市小资 – IT 领域的最热门工作在哪里？

简介：

“４－５年后，嵌入式智能电脑将是ＰＣ和因特网后的最伟大的发明。”网络之父、搜狐最早投资人、麻省理工学院教授尼葛洛庞帝下了这样的预言。近年来，随着软硬件资源的成熟与完善和Internet的普及，我们已经进入了网络时代，后PC嵌入式时代。嵌入式系统的应用得到了迅猛的发展，其应用领域涉及通信，自动化，信息家电，军事等各个方面。据ＩＤＣ发布的统计表明，未来5年间，信息家电市场将增长５－１０倍。由此可见嵌入式软件应用开发的巨大潜力与商机。同时，市场对嵌入式软件开发人员的需求也日趋上升，与现有的嵌入式开发人数产生了强烈供需矛盾。未来几年必有更多的硬件设计人员及应用软件开发人员进入嵌入式应用开发领域

特色：

- 不仅在技术上引你入门，更给予你选择正确人生道路的启迪
- 以主流厂商 CPU为嵌入式软件设计的切入点，深入介绍嵌入式软件设计技术
- 以业界最流行的C语言为正向设计语言，贴近工业实际
- 不是来自学院派的陈旧理论，而是源自工业界的实战演练
- 小班授课，限额10名，由上海浦东新区后生集成电路培训中心颁发结业证书
- 设有丰富实验环节（50%为实验时间），并设有专门讨论时端
- 采用中英文授课及讨论，使用英文教程及英文讲义
- 对优秀学员提供可能的推荐就业机会及提供可能的实习机会
- 免费参加全额奖学金留学美加五一长假指导班

招生对象：所有有志于嵌入式软件设计者，具有电子工程或计算机专业背景更佳。

收费标准：

费用包括讲课，实验，现场讨论, 工作午餐，英文教材及英文讲义，全日制在校生凭有效身份证及学生证享受半价优惠。只针对个人，不接受公司报名。

报名联系办法：

电话（周一至周五）：021-5080-2811，5080-2813, 5131-4066, 5131-4177
传真：021-5131-4176　E-mail: training@hwswworld.com

上课时间及地点：

(7/9~7/12)（上午九点至下午四点），合计 24 学时。在浦东软件园 22217-22219 座

开户银行：工行科苑支行 收款人：上海浦东新区后生集成电路培训中心 帐号：1001194909026460463

报名回执 （请传真回执及汇款回单以确认报名，一经确认，恕不退款，研修班取消除外）

单位	电子邮件
姓名	付款方式 （电汇或贷记凭证）
电话/传真	是否需要预定浦东软件园招待所（150 元三星级标间，费用自理）或科技部上海韩国语培训中心宿舍（50 元 5 人间，费用自理）？

具体报名流程：

1．填写上述报名回执表格传真给我们，也可登陆我们的 System Design Mall 网站在线报名；

2．按填写的付款方式将报名费用转至我们的帐上

3．将付款付款凭证的复印件、身份证复印件、学生证复印件（如适用）传真给我们；

4．我们发报名确认传真；

Program Agenda (Subject to Change Without Notice)

Agenda
0. Opening Remark **1.Introduction to Embedded SW Design** 1.1 What Is an Embedded System? 1.2 The Build Process 1.3 Compiling 1.4 Linking 1.5 Locating 1.6 Local Debugging – Debug at Host 1.7 Downloading 1.8 Remote Debugging - Debug at Target 1.9 Emulators, Simulators and Other Tools **2. Hardware Basics** 2.1 Dedicated HW Vs Programmable HW 2.2 Computation Models & Communication Schemes 2.3 MCU, CPU & DSP 2.4 On-chip & Off-chip Peripherals 2.5 Initialize the Hardware: Things to Be Done before Entering main() Function 2.6 Memory Classification: ROM, RAM & Flash; On-chip Memory Vs Off-chip Memory 2.7 On Flash Programming **3. Peripherals** 4.1 Control and Status Registers 4.2 The Device Driver Philosophy 4.3 A Simple Timer Driver 4.4 PLL, GPIO, UART, RS232, SPI & I^2C In Brief **4. Introduction to ARM Development Environment – Hardware & Software** 1.1 ARM IDE: Compiler, Debugger, Linker and Emulator – the PC Host Side 1.2 On JTAG : JTAG Configuration & Troubleshooting 1.3 ARM7TDMI(-S) based Board – the Target Side **2. Open Discussion**
Lab 1 Make ARM Board Communicate with Host through Its UART Interface

1. The ARM HW Architecture - ARM7TDMI Processor Core
1.1 Overview of ARM, registers, modes, exception handling, instruction sets, supporting technologies
1.2ARM7TDMI pipelines, data paths and instruction decoding.
2. The ARM SW Architecture - ARM and Thumb Instruction Sets
2.1Overview of the ARM and Thumb Instruction Sets. Includes practical work.
2.2 Mixing ARM and Thumb code in the same application
3.ARM7TDMI Bus Interface
3.1Memory Interface, clocking and interrupts on the ARM7TDMI
3.2The ARM on-chip bus architecture: AHB, ASB and APB
4. ARM7TDMI based SoC LPC2106's On-chip Peripherals In Depth
4.1 UART
4.2 GPIO
4.3 I2C
4.4 SPI
5. Open Discussion

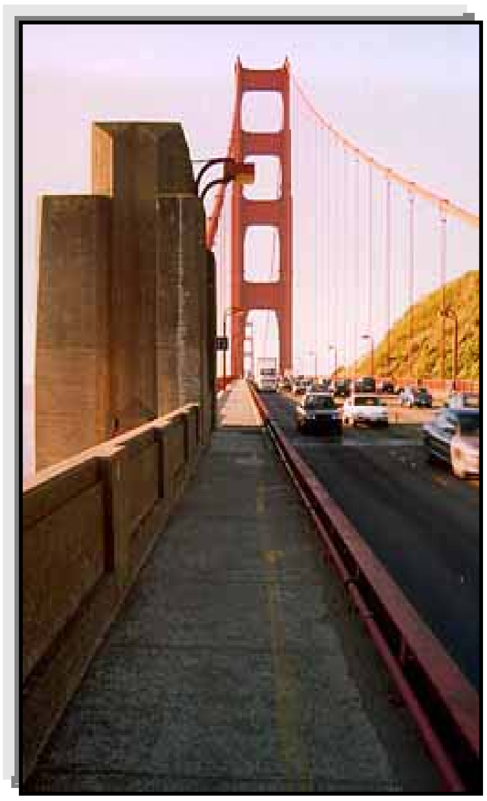
1. Memory Mapping in ARM7TDMI based SoC LPC2106
1.1 On-chip Flash & On-chip RAM
1.2 Memory Mapping Mechanisms
1.2.1 Boot Loader based Mapping in System Mode
1.2.2 Flash based Mapping in User Mode
1.2.3 RAM based Mapping in User Mode
2. Real-time Operating Systems (RTOS) in Embedded SW Design
2.1 Why RTOS ?
2.2 RTOS Characteristics
2.3 RTOS u-COSII In Depth
2.4 Task Scheduling in u-COSII
2.5 Memory Management in u-COSII
2.6 Interrupt Handling in u-COSII
2.7 Key u-COSII System Calls
2.8 Interrupts with or without RTOS's Supervision
3. Optimizing Your Code
3.1 The Importance of Coding Style
3.2 Coding from HW Point of View
3.3 Code Profiling for Performance
3.4 Increasing Code Efficiency
3.5 Decreasing Code Size
3.6 Reducing Memory Usage
3.7 Reduce Power Consumption
4. Advanced Embedded Software Development In Practice
4.1 Running code from ROM
4.2 Reset handlers
4.3 Locating code and data in memory (scatter loading)
4.4 Library retargeting.
4.5 Debug Solutions
4.5.1 On-chip debug logic
4.5.2 Debug Mode Vs Release Mode
4.5.3 Breakpoints and Watchpoints
4.5.4 Debug Communication Channel
4.5.5 Debug Target Code at Host through JTAG
4.6 System Design Considerations
5. Open Discussion

Lab2 Design a Binary Calculator SW Using SPI & GPIO on ARM Board
Floating Lab: Porting RTOS uCosII to ARM & Multitask Programming under uCosII

上海浦东新区后生集成电路培训中心

Linux 平台下软件设计出国与就业指导班

(7/16 ~ 7/19)



背景：
中国加入 WTO 后，知识产权保护将逐步规范，使得更多企业转向自由开放的、成本较低的 Linux 操作平台。据权威组织 IDC 统计，去年，Linux 在服务器市场上的占有率超过 27%，其增长率超过 Windows 操作系统 4 个百分点，同时中国人才市场也亟需 Linux 方面的专业人才。目前从事 Linux 应用开发的专业人员全国仅有 3 万多 人，国家正大力扶持 Linux 产业，预计到 2008 年对于 Linux 软件工程师的需求将达到 120 万人。

简介：
- 以 Linux 为平台，详细阐述系统编程所需的操作系统内核知识
- 以 Linux 为平台，深入介绍相关 CASE（计算机辅助软件工程）工具
- 不是来自学院派的陈旧理论，而是源自工业界的实战演练
- 小班授课，限额 10 名，考核合格者颁发上海浦东新区后生集成电路培训中心结业证书
- 设有丰富实验环节（50%为实验时间），并设有专门讨论时端
- 采用中英文授课及讨论，使用英文教程及英文讲义
- 对优秀学员提供可能的推荐就业机会及提供可能的实习机会
- 我们的教育培训只针对个人，不接受公司报名

招生对象：所有有志出国或高薪就业者

收费标准：
费用包括讲课，实验，现场讨论,工作午餐，英文教材及英文讲义，全日制在校生凭有效身份证及学生证享受半价优惠。只针对个人，不接受公司报名。

报名联系办法：
电话（周一至周五）：021-5080-2811，5080-2813， 5131-4066， 5131-4177
传真：021-5131-4176　　E-mail: training@hwswworld.com

上课时间及地点：
7/16 ~ 7/19（上午九点至下午四点），合计 24 学时，软件园 22217-22219 座

开户银行：工行科苑支行收款人：上海浦东新区后生集成电路培训中心 帐号：1001194909026460463

报名回执 （请传真回执及汇款回单以确认报名，一经确认，恕不退款，指导班取消除外）

单位 电子邮件

姓名 付款方式 （电汇或贷记凭证）

电话/传真 是否需要预定浦东软件园招待所（150 元三星级标间，费用自理）或科技部上海韩国语培训中心宿舍（50 元 5 人间，费用自理）

具体报名流程：

1. 填写上述报名回执表格传真给我们，也可登陆我们的 System Design Mall 网站在线报名；
2. 按填写的付款方式将报名费用转至我们的帐上
3. 将付款付款凭证的复印件、身份证复印件、学生证复印件（如适用）传真给我们；
4. 我们发报名确认传真；

Program Agenda (Subject to Change Without Notice)

1. Opening Remark
2. Linux Kernel in Depth
2.1 Kernel Architecture
2.1.1 Kernel Layout and Configuration
2.1.2 Kernel Style and General Considerations
2.1.3 Kernel Modules
2.1.4 Kernel Initialization
2.1.5 Kernel Debugging Techniques
2.2 Synchronization Methods
2.3 Processes
2.3.1 Process Limits and Capabilities
2.3.2 Process Scheduling
2.4 Interrupts and Exceptions
2.5 System Calls In Depth
2.6 Signals
2.7 Memory Management
2.8 File Systems
2.9 The Configurable Nature of Linux
2.10 The Real-Time Extension of Linux – Does It Work?
3. Open Discussion

4 Key GNU SW under Linux In Depth
41 The Free SW Foundation (FSF) & GPL License
42 The history of GNU & GNU SW
43 C Shell Vs B Shell & Their scripting
4.4 The Make Utility & Configuration Utility
4.5 Retargetable GNU Compiler gcc In Depth & Demo
4.6 Retargetable GNU Debugger gdb In Depth & Demo
4.7 GNU Editor vi & emacs In Depth & Demo
4.8 Memory Leakage Detection under Linux & Corresponding Tool Demo
4.9 GNU Performance Profiler gprof In Depth & Demo
4.10 Linux Security Issues
4.11 Linux Specific Porting Issues
4.12 From Client/Server Computing to Multi-Tier Computing: Introduction to J2EE

5. Open Discussion

6. SW Engineering Practice – The State-of-the-Art In Depth
6.1 Feature-driven Requirement Elicitation, Analysis, Design, Test & Maintenance
6.2 Test First Programming
6.3 Distributed Development & Parallel Development
6.4 Daily Build/Release & Milestone-based Synchronize & Stabilize
6.5 Trade-off between Running Time Complexity & Memory Complexity
6.6 Ascending from Client/Server Computing to Truly Distributed Computing
6.7 On-demand Computing, Grid Computing & Self-healing Computing

7. The Impact of SW Architectural on Linux
7.1 Architecture – The Bridge between SW Requirement Analysis & SW Design
7.2 The 4+1 View of SW Architecture
7.3 SW Architectural Structures & Variations: UI-centric Architecture, Data-driven Architecture, Control-intensive Architecture etc.
7.4 The Seamless Mapping among SW Architecture, Team Architecture & Project Architecture – How Linux Was Developed as Open Source SW?
7.5 The Open System SW Architecture - Linux SW Architecture In Depth
8. Open Discussion

9. The Importance of SW Process for Large-scale SW Development under Linux
9.1 The Rectangle of SW Project Success: People, Process, Technology & Product
9.2 Light-weight SW Process at In Depth
9.2.1 Extreme Programming (XP)
9.2.2 SRUM
9.2.3 Feature-Driven Development (FDD)
9.3 Distributed Development & Parallel Development through Advanced Configuration Management & Test 9utomation – ClearCase, ClearQuest, Robert, Perl Scripting & LSF Revisited
9.4 The Importance of In-house Libraries & Other SW Modules Designed for Reuse
9.5 The Importance of Custom Memory Manager: How It Works & How It Is Designed?
9.6 The Daily Build/Test/Release & Mile-stone based Synchronize & Stabilize Practice
9.7 Why We Need to Differentiate between Point Tool based SW Development Vs Framework based SW development, and How We Do It Differently?
9.8 The Importance of Top Performers Composed Tiger Team under the Leadship of Superstar

10. Configuration Management under Linux
10.1The Zero-Defect SW Development Practice
10.2 The Principle of Change Management
10.3 Version Control In Depth: Backward Merging Vs Forward Merging among Others
10.4 Client/Server Mode CVS Demo
10.4 Change Request (CR) Tracking in Depth & Corresponding Linux Tool Demo
10.5 On Configuration Management Automation

11. Test Automation under Linux
11.1 Data-intensive Testing Vs Scenario-based Testing
11.2 Five Generations of Test Automation SW: Record & Replay, Record & Replay plus Scripting, Data-driven Scripting, Framework-centric Data-driven Scripting, Framework-centric Data-driven Scripting with Separated Host/Target
11.3 The Golden Result Concept & Regression Test In Depth – Make Your SW Better Of Today
11.4 Quality-of-Result (QoR) Test In Depth – Go Beyond Functional Correctness
11.5 Perl Scripting for Regression & QoR Test Automation
11.6 Record & Replay for GUI Test Automation
11.7 In-house Test Automation Platform SW Demo
12. Open Discussion

Advanced IC Design Training Courses

IC1	Advanced RTL Design with Hard-wired/Programmable Controller & Data path
IC2	Advanced Logic Synthesis for Rapid Timing Closure & Signal Integrity
IC3	Advanced Verilog for RTL Design
IC4	Migrating from ASIC Flow to COT Flow for IP-based SoC Design
IC5	Testing Techniques for SoC/ASIC Design - ATPG, DFT & BIST
IC6	Data path Design & Optimization for High Performance ASIC/SoC
IC7	PLL & DLL for Synchronous Design
IC8	Embedded SW Design & Optimization for IP-based SoC
IC9	Embedded SW Test Automation from Design Point of View
IC10	High Performance FPGA Design in Practice
IC11	Scan Insertion & Optimization for Testable Design
IC12	Clock Tree Synthesis & Optimization for High Performance Design
IC13	Technical & Business Aspects of Project Management for ASIC/SoC
IC14	IP based SoC Integration in Practice
IC15	Low Power Design Techniques in IP-based SoC
IC16	Very High Speed Design Techniques for Low Power ASIC/SoC Design
IC17	Advanced Physical Synthesis - Theory & Practice
IC18	Advanced Static Timing Analysis & Optimization for ASIC/SoC Design
IC19	Advanced Functional Verification Techniques for ASIC/SoC Design
IC20	From Verilog/VHDL to System C/System Verilog - Ascending from RTL Design to System-level Design
IC21	Timing-driven (Traditional) Logic Synthesis: Theory & Practice
IC22	Very/Ultra Deep Sub-Micron (VDSM) Issues in SoC Design
IC23	Advanced Synthesizable VHDL for RTL Design
IC24	C/C++ based HW Modeling & Verification at System-Level
IC25	CMOS Analog Design
IC26	Key Issues in Mixed-Signal Design

Advanced Software Engineering Training Courses

SW1	SW Agile Development Process & Its Enabling Technology
SW2	Pattern & Component based Next Generation RAD Technology - From Client/Server Computing to J2EE based Distributed Computing
SW3	Advanced Algorithms & Data Structures in SW Engineering
SW4	Leapfrog to CMM III through SW Process Automation
SW5	Advanced Symposium on Component-based OOA/OOD Using UML
SW6	Pattern & Component based Next Generation RAD Technology ¨C Advanced J2EE In Depth
SW7	Advanced Symposium on Pattern-based OOA/OOD Using UML
SW8	Large-scale SW Development under Linux Platform
SW9	OO Based SW Development & Project Management with Pattern & Component Technology
SW10	Project Management for Enterprise-Scale SW
SW11	SQA & SCM and Their Automation for Enterprise Scale SW
SW12	SQA & SCM Automation & Their Relationships with R & D
SW13	RAD with Pattern & Component based OOA & OOD
SW14	Enterprise Scale Modern Requirement Management & Its Automation
SW15	SW Architecture Design for Enterprise-Scale Applications
SW16	Goal-oriented, Architecture-centric Project Management for Enterprise-Scale SW
SW17	Technical Issues & Business Issues in SW Outsourcing Project
SW18	Testing Automation for Embedded SW
SW19	Testing Automation of Web-based, DB-backed Multi-Tier SW

Industrial Job Openings

Job Openings at IDT China

Title: Design Engineer

Job Description:

This person will be responsible for designing integrated circuits. Specific tasks will include schematic capture, layout, LVS, and DRC verification, Verilog logic simulation and Hspice circuit simulation. Writing circuit descriptions, RTL level verilog, Synopsys synthesis, parasitic extraction, back-annotation and static timing analysis may also be required.

Your resume should be sent to hr@nw.idt.com , IDT China has more job openings on design, testing and marketing, visit http://china.idt.com for more detailed info.

Job Openings at GSMC

The following positions are open at Grace. If you are interested and satisfy the prerequisites, we encourage you to apply. Please send relevant information (your resume¨|, the position you are interested in, etc) to our HR department (contact information below). Applications submitted on-line will be preferentially processed. Note that there may be other positions posted on the Chinese version of this page that are not posted here. These typically require native or close-to-native Mandarin speaking ability.

Title: Process Integration Engineer
Job Responsibilities:
1. Process flow handles & maintains to improve production yield
2. Quickly trouble shooting and co-work with module to make sure process flow smooth.
3. To improve Wafer Acceptable Test CP/CPK

Requirements:
1. Master degree or above
2. Major in electronic science, physics, chemistry, material Science or related majors
3. Fluency in written and spoken English, CET6

Title: Equipment Engineer PH/ETCH/TF/DIFF/CMP
Job Responsibilities:
1. Execute preventive maintenance
2. Execute machine troubleshooting
3. Assist to execute tool and parts continue improvement program

Job Requirements:
1. Bachelor Degree
2. Major in microelectronic, Mechanism, electrical, automation or related majors.
3. Fluency in written and spoken English, CET6
4. Male is preferred, can accept shift work

Title: Yield Enhancement Engineer
Job Responsibilities:
1. Focusing on execution of improvement
2. Maintaining production environment
3. New system new technology installation
4. Key trouble-shooter in big event

Job Requirements:
1. Master degree
2. Major in Electronic science, Physics, chemistry, material Science or related majors
3. Fluency in written and spoken English, CET6
4. Could work in night shift and 24 hours on call

Title: Process engineer (Diffusion/CMP/Photo/Etch/Thin film)
Job Responsibilities:
1. Improvement and maintenance of process related to diffusion, CMP chemical mechanical polish, Photo, Etch, Thin film
2. Evaluation and automation of process related to diffusion, CMP chemical mechanical & polish, Photo Etch, and Thin film

Job Requirements:
1. Master Degree
2. Microelectronics, material, chemical, chemical engineering, physical, or related majors

3. Good Oral and written English, CET6

E-mail any applications or inquiries to recruiting@gsmcthw.com
If you prefer, you can also contact us via snail mail at:
Human Resources Department
Grace Semiconductor Manufacturing Corporation
818 Guo Shou Jing Road,
Zhangjiang High-Tech Park
Shanghai, 201203,
People's Republic of China

Job Openings at Bearing Point, Inc

Bearing Point, Inc, is one of the world's largest consulting firms with approximately 16,000 professionals serving more than 2,100 clients, including Global 2000 companies and major government organizations. Our professionals deliver strategies, processes and technologies to enable our clients to transform or improve their businesses. Our mission is to be the world's most influential business systems integrator, providing competitive advantage for our clients, growth opportunities for our people and long-term value for our shareholders.

Bearing Point has built a Global Development Center with 4,500sm space on 4 floors located in Zhangjiang Hi-Tech Park, Shanghai Pudong New Area. The Bearing Point China Global Development Center will provide world-class software development and application package implementation for SAP, Siebel, Oracle and People Soft. Bearing Point GDC has already been certified at SEI CMM Level 3 in Oct.2003 and is striving for achieving CMM Level 4 by end of 2004 and Level 5 by July 2005.

We are looking for talents for both Bearing Point Management Consulting and Bearing Point Global Development Center.

Available Positions:
ERP - People soft Developer/Senior Developer
ERP - Project Manager
ERP - SAP Basis
ERP - Siebel Developer
ERP - Siebel Technical Lead/Lead Developer
ERP Developer

Recruiter：
Service Lead / Engineer - People Soft Application Management
Service Lead / Engineer - Siebel Application Management

Data Process Service (Specialist or Project Manager)
Managed Service-Service manager
Managed Service-Solution Architect
Managed Service - Network LAN specialist
Managed Service - Network WAN specialist
Managed Service - Network administrator
Managed Service- Network routing specialist

Accountant:
Core/EAI Project Manager
Data Warehouse & BI Developer/Senior Developer/Technical Lead
EAI Developer/ Senior Developer/ Tech lead
Hyper ion Technical Lead/Senior Developer/Developer
IT Support Engineer
J.D. Edwards Analyst
J2EE Developer/Senior Developer
Microsoft.NET Developer/Senior Developer/Technical Lead
Office Administrative Assistant
Oracle Developer (Japanese Speaking)

Receptionist:
Recruiter (DL)
SCM Engineer
SQA Engineer
Software Test lead / Project Manager
Software Testing Engineer
Software Testing Senior Engineers
Unix System Administrator
Administrative Assistant / Executive Assistant
Business Development Manager - Japanese Speaking
Consultant / Sr. Consultant ¨C Web Designing
Corporate Counsel
Executive Assistant (Japanese Speaker)
IT Architect Credit Application - Consultant / Sr. Consultant / Manager
M&A Corporate Development Manager
Manager - ERP Project Management
Manager in Insurance Accounting
Manager, E-Learning
Oracle MFG Sr. Consultant
Oracle FIN Sr. Consultant
People Soft Consultant / Sr. Consultant
Property or Life Insurance IT Architect - Manager / Sr. Consultant

Receptionist:
Senior Consultant / Mgr. - Six Sigma (6 Sigma)

Senior Consultant/Manager. (Technical Related)

Motivated candidates please send your English and Chinese resume to:
Bearing Point Global Development Center
Bldg. 16, Pudong SW Park, 498 Guoshoujing Rd,
Shanghai 201203, PRC
http://www.bearingpoint.com
E-mail address: PDG.Recruitment@bearingpoint.com

~~~~~~

Job Openings at SMIC

Title: ASIC Backend Designer:

Candidates will join our design reference team, placing and routing ASIC chips using automatic P&R tools, such as Synopses Apollo/Astro, Cadence First Encounter/Plato NanoRoute, or SE/PKS.

Job Requirements;

-2+ years direct experience in physical design.
-Must be script oriented.
-Min. BS, prefer MS or PhD.
E-mail: HLi@smics.com

Title: Principle Engineer/Manager

-Wafer level reliability development
-Reliability test key design
-Reliability modeling & simulation
-Co work with TD, process integration, Fab modules for process reliability improvement & optimization
-Familiar with reliability knowledge to well communicate with customer & related units
-Better to have statistical background
-Need to have Fab experience for build-in reliability
-Be familiar with Excel, PowerPoint, Word, Windows 2000, Matlab

Title: Technician Assistant / Senior Technician Assistant

Job Responsibility:

Electrical technician for system operation, repair and maintenance
Principal Accountabilities.
-To be very familiar with utility layout and function (FAB1, 2,3B,CUB, PS, CWWT, BH).
-To do system operation for GIS, LV/HV switchgear, Generator, UPS, SCADA
-To do electrical shift job independently. To maintain electrical system under the instruction of engineer.
-To do easy emergency troubleshooting or repair for electrical system.
~~~~~~

-To do small installation project.
-To supervise the site electrical erection engineering.

Job Requirements:

Education or Working Experience
-Graduated from vocational school or above
-At least five years of duty experience in 10KV power station

Professional Skill:

-Have LV and HV Electrical Operation Certificate.
-To be electrical shift duty and secure the whole electrical system operation and emergent operation.
-Be able to do electrical system operation for LV & HV switchgear, PA, Generators, UPS.
-Be able to do easy system maintenance for MCC, LV switchgear, PA.
-Be able to do small electrical installation work.
-Be able to repair lighting and MCC circuits.

General Skill:

-Be able to do easy report
-Be able to use windows and office software
-Have clear mind and easy to communicate with.

Title: SAP Technical Leader

Job Responsibility:

Mangage Technical Development section as well as direct and manage ongoing infrastructure and technical development activities.

Principal Accountabilities:

-Mangage and direct daily activity to ensure SAP infrastructure availability.
-Detailed technical infrastructure planning, resource allocation, evaluation, and technical performance analysis.
-Coordinates and communicates within the team structure.
-Provides overall BASIS expertise in SAP.
-Give direction for SAP transport management and infrastructure landscape.
-Ensure future compatibility of hardware and infrastructure design with forth coming business requirements and projects.
-Ensure system availability and performance.
-Support the SAP implementation.
-Oversee all custom ABAP development, repairs, enhancements, as well as maintain and design interfaces to other systems via RFC's and Idoc's.

Education:

Bachelors in Computer Science or related field.
Experience: 4+years related work experience with 2+years of related SAP BASIS experience. Significant hands on technical knowledge of SAP.

Other Essential Attributes:

- 2+years experience programming in a structured language such as C/C++.
- 2+years experience with implement SAP.

- 2+years experience configuration experience in SAP.
http://www.smics.com/website/enVersion/Careers/index.jsp
E-mail: bj-hr_admin@smics.com

Title: Manufacturing Line Leader

Job Responsibility:

-Leading team members to perform and to complete daily mission
-To be the major trainers for trainees
-Initiate daily production demands clearly
-To be a disciplinarian
http://www.smics.com/website/enVersion/Careers/index.jsp

Title: Manufacturing Engineer/Line Supervisor

Job Responsibility:

-Coordinate internal & external resource to achieve daily production goal
-Raise productivity through continuous improvement methodology
-Shorten cycle time with systematic methodology
-Reduce manufacturing cost without affecting quality
Sex: Male
Language: Chinese/English is good
E-mail: bj-hr_admin@smics.com
http://www.smics.com

Title: Fab Material/Parts Forecast Analysis

Job Responsibility:

Electrical technician for system operation, repair and maintenance
Principal Accountabilities
-To be very familiar with utility layout and function (FAB1, 2,3B,CUB, PS, CWWT, BH).
-To do system operation for GIS, LV/HV switchgear, Generator, UPS, SCADATo do electrical shift job independently.
-To do electrical system maintenance under the instruction of engineer.
-To do easy emergent troubleshooting or repair for electrical system. To do small installation project.
-To supervise the site electrical erection engineering.

Job Requirements:

Education or Working Experience
-Graduated from vocational school or above
-At least five years of duty experience in 10KV power station

Professional Skill:

-Have LV and HV Electrical Operation Certificate.
-To be electrical shift duty and secure the whole electrical system operation and emergent operation.
-Be able to do electrical system operation for LV & HV switchgear, PA, Generators, and

UPS.
-Be able to do easy system maintenance for MCC, LV switchgear, PA.
-Be able to do small electrical installation work.
-Be able to repair lighting and MCC circuits.

General Skill:

-Be able to do easy report
-Be able to use windows and office software
-Have clear mind and easy to communicate with.

E-mail: Jim_Liu@smics.com
http://www.smics.com

Job Openings at Alphatec Electronics Corp. of Shanghai

Title: HR Manager

Job Qualification:

Male/Female, Bachelor's degree or higher in any related fields
3-5 years experiences in Personnel or HR Manager Level
At least 5 years experiences in Employee Relation or Organization development
Good people development and interpersonal skills, outstanding business awareness
Have lots of initiative, English communication ability, and be service minded

Title: Senior Training Officer

Job Qualification:

Male/Female, Bachelor's degree or higher in any related fields
At least 3 years experiences in training and development in manufacturing environment is preferable
Knowledge of Quality & Environment System Certification ie; ISO9001: 2000, ISO/TS16949, ISO14000 etc. will be advantage
Good command of both spoken and written English
Computer knowledge and skill in Microsoft Office

Title: Management Accountant

Job Qualification:

Male/Female, Bachelor's degree or higher in Accounting or Finance
At least 3 years relevant experiences in manufacturing environment
Knowledge of Costing is a must
Experience in an auditing firm is a plus
Self motivated with strong analytical skill
Good command of both spoken and written English

Title: Quality Engineering Manager

Job Qualification:

Male, Age above 30 years
Bachelor's degree or higher in electronics, electrical or mechanical engineering
At least 8 years in process engineering, QA/QC
Experiences in semiconductor manufacturing is a plus
Good command of both spoken and written English

Please submit your resume to HR&Facitlities Director
Prasithra@m-microtech.com or fax #038 845 572
17/2 Moo 18 Suwintawong Rd, T. Saladang, A.Bang-nam-priow, Chacherngsao 24000

Job Openings at Innosis, Suzhou

Title:Embedded SW Engineer

Possess profound knowledge or experience in one or more of the following areas:
One or more of the operating system kernels such as Linux/Unix, Windows, RTLinux, WindowsCE and uCOS.
OS porting on embedded platforms such as ARM core based chips.
Linux or Windows drivers such as USB, PCI, Storage devices, Display devices, IEEE1394, PCI, USB and 802.11b/a/g.
BIOS or boot loaders in embedded systems, ARM experience is preferred
GUI in embedded applications.
File system in embedded applications.
High frequency board design including schematic design, PCB layout and testing.
Implementation of network protocols such as TCP/IP, UDP, PPPoE, PPPoA, ARP, ICMP, DHCP, TFTP and RIP v1/2.
Implementation of MPEG2, MPGEG4 or AVS.
Personnel Specialist/Supervisor

Job Requirement:

1. Bachelor degree and above, major in HR management is preferred.
2. At least 3 years or 7 years of recruiting & HR experience in multi-national IT company.
3. Proven hands-on experience with recruiting process & interview skills.
4. Strong interpersonal communication skills.
5. Proficient in English reading, writing & speaking well. CET-6 is preferred.
6. Strong computer know-how, including Microsoft Office, MS Windows OS.

Job Responsibility:

1. Handle company-wide recruiting & employee relations' functions.
2.Conduct HR cross function projects, collaborating with Training, IT, and other related departments.

Title: IC Layout Technician

Good reading and writing in English
Good computer skill, especially in Office 2000, C Programmed
Major in Electronics, Computer and Automatic control

Title: IC Verification Managers (3 positions in total)

Job Responsibilities:

- Developing system level verification environment for Digital TV products.
- Developing detailed test specification.

Job Qualifications:

- College graduate, EE or related field.
- 3 to 5 years IC hardware design or verification experience.
- Good knowledge in PCI, I2C and USB bus architecture.
- Familiar with VCS, Vera, Verilog-XL, Debussy, Design Compiler, Altera Quartus and Xilinx tools.
- Proficient in Verilog, Perl, C/C++ & Assembly Language programming.
- Experience with test bench methodology, test plan & test case creation, simulation debugging, simulation modeling, random test generation, coverage analysis.
- Strong problem solving skills.
- Self-motivated team player.
- DTV knowledge is preferred.

Title: Digital ASIC Engineer

1. Two years experience with MSEE is a must;
2. Experienced in verilog RTL coding;
3. Good knowledge of synthesis, STA, etc.digital chip design flow;
4. Ability in architecture and specification define;
   C or C++ programming background is preferred.

Title: Software Application Engineer

Software Engineer experienced in Windows application;
Experience in good GUI development is a must;
Familiar with Graphics format like BMP, PCX, etc.;
Experience in Compiler Programming is a plus;
At least 2 year working experience.

Title: Hardware Designer: C System-level

Job Description:

Innosis is looking for engineers for the DTV product focusing on the DVB-T and DVB-C asic solutions. You will bring your knowledge of communication receiver knowledge in real-time system environment. You will contribute your background with hands-on simulation tools experience. You will join a team of extremely productive and dedicated engineers. - Define and trade off architectures for DVB-based point-to-multipoint

communication system development.
- Analyze system-level performance and detailed implementation issues.
- Develop C code in MATLAB, then C++ or C code for independent platform.
- Keep abreast of the current technologies and exhibit aptitude to learn state-of-the-art Technologies and prototype new technologies.

Job Requirements:

-MSEE and above with at least 3 years experience in communications engineering.
-Proven ability to code/simulate algorithms in MATLAB simulink environment.
-Hands-on knowledge of OFDM signal processing, and communication theory is required, as well as experience developing DSP algorithms and implementations.
-Experience in implementing timing recovery, carrier recovery, PLL, TDE/FDE, Reed-Solomon decoder, punctured convolution codes.
-Good understanding of DVB-T or/and DVB-C standards is a must.
-Good communication, organizational, presentation, and interpersonal skills required.

Title: Senior Executive Assistant

Job Requirements:

Minimum 5 years of secretarial experience at foreign enterprises
Minimum 2 years at administrative related supervisory experience
Superb interpersonal, interface & communication skills
Proficient in English both in writing and speaking
Hands-on PC skills, including Microsoft Office
Knowledge of general office procedures (e.g., filing, correspondence, scheduling)
Tact and good judgment in confidential situations and the ability to interact with senior management
Be highly energetic, self-motivated: Must perform under minimum supervision
Desire to develop management skills, a professional appearance and demeanor and a proven track record of career stability

Job responsibilities:

1. Supports President and oversees the general administrative responsibilities for the company.
2. Perform diversified secretarial duties and administrative functions requiring confidentiality, initiative and sound judgment for a manager or a team of principals.
3. Drafts and edits written correspondence, reports, presentations, and other documents on behalf of the President as requested.
4. Arranges complex and detailed President's daily itineraries, compiles documents for meetings, and accompanies supervisor when requested.
5. Takes and transcribes dictation, and composes and prepares confidential correspondence, reports, and other complex documents.
6. Maintains detailed records and a filing system for all documents, including meeting minutes, business cards, budget and financial reports, employment requisitions and personnel actions, etc.
7. Plans and executes company's quarterly internal customer surveys and host quarterly all-staff communication meetings.
8. Act as editor for company monthly newsletters.

Title: Senior Algorithm Implementation Engineer/Manager

Must have MSEE or equivalent (Ph.D preferred).
Must have extensive experience in C/C++ & Matlab.
Must have in-depth experience with Video/Imagine & Telecom technologies & algorithms.
Must have excellent working knowledge of TV Codec, Color Space Transfer and processes with hands-on R&D and implementation experience.
Experience with MPEG is a big plus.

Title: Verification Engineer

For digital circuit design, simulation/Verification, test pattern generation:
1.2-3 year IC design experience with MSEE is a must;
2.Knowledge in IC Design tool set, simulation, verification, synthesis;
3.Knowledge in digital video processing experience;
4.Knowledge in I.C. design back end experience preferred.

Title: Sales Engineer

1. BS in Electronics and Electrical Engineering;
2. Good Communication skills;
3. At least 3 years relevant working experiences in IC sales and Agent Management in prior;
4. Good at English speaking, listening, reading and writing.