GABRIEL COSTA LEAL DA CUNHA

Uma solução para a limitação do uso de licenças flutuantes baseada na configuração da plataforma computacional LSF

Campina Grande - PB, Brasil

Fevereiro de 2010

GABRIEL COSTA LEAL DA CUNHA

Uma solução para a limitação do uso de licenças flutuantes baseada na configuração da plataforma computacional LSF

Relatório de Estágio apresentado ao Curso de Engenharia Elétrica da Universidade Federal de Campina Grande, em cumprimento parcial as exigências para obtenção do Grau de Engenheiro Eletricista.

Orientador:
Prof. Glauco Fontgalland

Centro de Engenharia Elétrica e Informática
Unidade Acadêmica de Engenharia Elétrica
Universidade Federal de Campina Grande

Campina Grande - PB, Brasil

Fevereiro de 2010

Sumário

Lista de Figuras

Agradecimentos

Em primeiro lugar, meus sinceros agradecimentos aos meus pais e família por todo o apoio e incentivo recebidos durante o curso de graduação em Engenharia Elétrica.

Gostaria de agradecer a direção do centro de Pesquisa e Desenvolvimento da Infineon em Sophia Antipolis por ter me oferecido a oportunidade de realizar este trabalho com toda a estrutura necessária, assim como ao programa de cooperação entre universidades do Brasil e da França, BRAFITEC, que me proporcionou a realização desta temporada de estudos na França.

Agradecimentos especiais ao grupo GLM (Stéphane Muller, Mustapha Aqachmar, Sylvain Kahn, Navdeep Nandagopal e Marion Sauze) por terem me ajudado direta ou indiretamente neste trabalho, em particular Stephane Muller, que me propôs este estágio e foi meu orientador na Infineon. Gostaria também de agradecer a Pierre-Yves Thillier, Jean-Yves Larguier, Richard Skelton e Julien Lesaux pela cooperação.

Finalmente, meus sinceros agradecimentos aos professores Glauco Fontgalland e Raimundo Carlos Silvério Freire, os coordenadores do convênio BRAFITEC na Unidade Acadêmica de Engenharia Elétrica da UFCG, por saber a importância pessoal e profissional de uma experiência de estudos em outro país e de colocá-la em prática através deste convênio. Agradeço também ao professor Tan Phu Vuong por me ter tão bem acolhido junto a ESISAR e ao professor Smail Tedjini como meu orientador de estágio na ESISAR.

Introdução

A companhia Infineon Technologies é composta de diversos grupos, dentre os quais GLM, Global License Management, dentro do qual eu fui integrado durante o desenvolvimento deste trabalho. O grupo GLM é responsável por prover licenças EDA a toda comunidade de engenheiros da companhia Infineon, de forma que eles possam simular, testar e validar os chips que são desenvolvidos. A maioria das licenças usadas na Infineon são licenças flutuantes e seu custo é baseado na sua utilização, como será descrito no capítulo 3.

O título deste trabalho se refere ao estudo e implementação de um sistema para redução de custos com uso de licenças, em cada um dos centros de Pesquisa e Desenvolvimento da Infineon. Este sistema é baseado na configuração da Plataforma LSF e é acompanhado por uma interface gráfica, desenvolvida usando a linguagem Perl [1], a ser usada no monitoramento do uso das licenças.

Os primeiros passos deste trabalho foram dados no intuito de se adquirir familiaridade com o ambiente de trabalho da Infineon, assim como com todas as ferramentas que seriam posteriormente usadas, por exemplo, FlexLM para gerenciamento de licenças, linguagem Perl e a Plataforma LSF. Em seguida, foram feitos testes para validação do sistema proposto, que foi parcialmente implementado no sítio Infineon de Sophia Antipolis até o fim do presente trabalho.

Abstract

This report firstly describes the use of the software FlexLM for license management as well as it presents a license classification based on two different criteria: FlexLM criteria and Cost Calculation criteria.

By knowing that most of the licenses used by Infineon Design Community are floating licenses, which are paid based on their use, the goal of this work is to propose and put in place a system based on LSF configuration setup to limit the use of such kind of licenses. This will be done at a local approach in a way the license managers placed at Sophia Antipolis will have full control on the limits imposed by the system.

A set of tests will be planed and done in order to validate the system before it can be put in place definitely.

By the end of the report we will present a Graphic Interface, written using Perl [2] scripting, which has been developed to allow the license managers checking the license status at real time.

Keywords: Floating Licenses, FlexLM, LSF and Perl Scripting.

Glossário

Daemon: Disk and Execution Monitor. Este termo foi supostamente inventado em 1963 por pesquisadores do MIT. Trata-se de um processo que é responsável por monitorar a execução de uma determinada tarefa.

DSP: Digital Signal Processor. Um DSP é um microprocessador específico para processamento de sinais, possuindo uma grande capacidade de manipulação matemática.

EDA: Electronic Design Automation. Por exemplo, ferramentas usadas na concepção de chips são conhecidas por ferramentas EDA.

EDGE: Enhanced Data rates for GSM Evolution. Evolução da tecnologia GPRS que possibilita taxas de transmissão ainda mais altas.

ESISAR: Ecole Supérieure d'Ingénieurs en Systèmes Industriels Avancés Rhône-Alpes. Escola de Engenharia da qual fui aluno durante esta temporada na França.

FlexLM: Flex License Manager. *Software* comercial responsável pelo gerenciamento de licenças.

GLM: Global License Management. Um dos grupos nos quais se divide a companhia Infineon no sítio de Sophia Antipolis.

GPRS: Baseado na tecnologia GSM, **General Packet Radio Service** é uma tecnologia de comunicação sem fio que possibilita taxas de transmissão que vão de 56 até 114kbps.

GSM: Global System for Mobile communication. Sistema de telefonia digital móvel.

LSF: Load Sharing Facilities é um *software* comercial usado pra gerenciar recursos computacionais.

RF: Radio-Frequência.

UMTS: Universal Mobile Telecommunication System. Tecnologia empregada

em sistemas de comunicações móveis da terceira geração. Alcança taxas de transmissão de até 2 Mbps.

TCP/IP: Transmission Control Protocol/Internet Protocol. Protocolo de comunicação usado pela internet.

1 *Infineon Technologies*

1.1 Infineon em âmbito mundial

A companhia Infineon Technologies foi fundada em Abril de 1999 quando a companhia Siemens decidiu separar suas operações na área de semicondutores e formar uma nova companhia. A Infineon atualmente ocupa o 11° lugar no ranking das companhias de semicondutores líderes em vendas, como é ilustrado na tabela da figura 1.

1H08 Top 20 Semiconductor Sales Leaders ($M)

1H08 Rank	2007 Rank	Company	Headquarters	1Q07 Tot Semi	2Q07 Tot Semi	1H07 Tot Semi	1Q08 Tot Semi	2Q08 Tot Semi	1H08 Tot Semi	1H08/1H07 % Change
1	1	Intel	U.S.	8,072	7,916	15,988	8,841	8,655	17,496	9%
2	2	Samsung	South Korea	4,697	4,552	9,249	5,569	5,618	11,187	21%
3	3	TI	U.S.	3,115	3,257	6,372	3,191	3,175	6,366	0%
4	4	Toshiba	Japan	3,249	2,510	5,759	3,055	2,789	5,844	1%
5	6	TSMC*	Taiwan	1,922	2,258	4,180	2,767	2,894	5,661	35%
6	5	ST**	Europe	1,953	2,086	4,039	2,179	2,391	4,570	13%
7	8	Renesas	Japan	1,949	1,984	3,933	2,142	2,195	4,337	10%
8	7	Hynix	South Korea	2,569	1,998	4,567	1,675	1,824	3,499	-23%
9	9	Sony	Japan	1,716	1,573	3,289	1,946	1,484	3,430	4%
10	14	Qualcomm***	U.S.	1,259	1,367	2,626	1,620	1,762	3,382	29%
11	12	Infineon	Europe	1,282	1,361	2,643	1,566	1,580	3,146	19%
12	15	NEC	Japan	1,346	1,366	2,712	1,475	1,523	2,998	11%
13	13	Micron	U.S.	1,370	1,328	2,698	1,391	1,529	2,920	8%
14	10	NXP	Europe	1,390	1,461	2,851	1,443	1,445	2,888	1%
15	11	AMD	U.S.	1,233	1,378	2,611	1,505	1,349	2,854	9%
16	16	Freescale	U.S.	1,295	1,310	2,605	1,304	1,389	2,693	3%
17	17	Fujitsu	Japan	1,047	1,061	2,108	1,226	1,108	2,334	11%
18	21	Panasonic	Japan	870	905	1,775	1,058	1,170	2,228	26%
19	19	Nvidia***	U.S.	847	896	1,743	1,158	985	2,143	23%
20	23	Broadcom***	U.S.	897	892	1,789	986	1,153	2,139	20%
—	—	Total Top 20		42,078	41,459	83,537	46,097	46,018	92,115	10%

*Foundry **Not incl. flash in 1H07 & 1H08 ***Fabless

Figura 1: Top 20 das companhias de semicondutores líderes em vendas.

Em Maio de 2006, a Infineon decidiu fundar uma nova companhia para assumir suas atividades na área de Memórias. Esta companhia recebeu o nome de Qimonda. Excluindo Qimonda, a Infineon é composta de aproximadamente 30.000 empregados que projetam, produzem e vendem semicondutores e sistemas para os setores: automotivo e industrial, assim como soluções em comunicações.

DSP's, Micro-controladores, Produtos RF, Circuitos Integrados e Dispositivos Semicondutores de Potência fazem parte dos produtos desenvolvidos pela Infineon. A figura 2

ilustra como os sítios da Infineon estão dispostos geograficamente em todo o mundo.

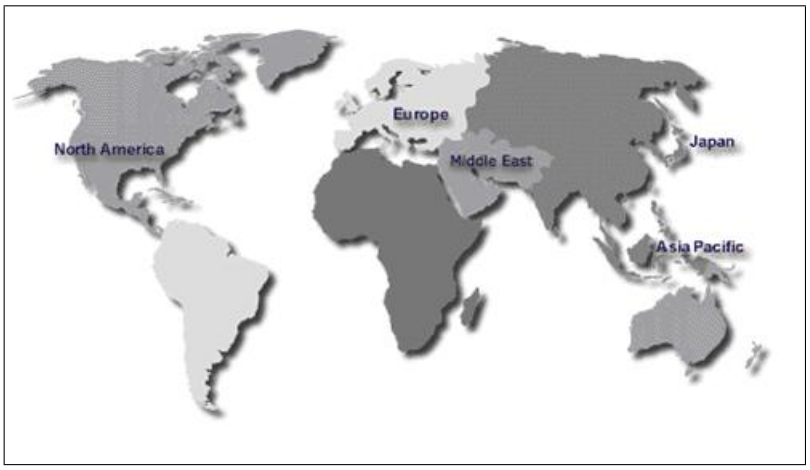

Figura 2: Localização geográfica dos sítios da Infineon.

Na França, a Infineon mantém dois sítios localizados em Saint-Denis e Sophia Antipolis. O primeiro sítio é encarregado de Marketing e Vendas enquanto o segundo é um centro de Pesquisa e Desenvolvimento.

1.2 Infineon em Sophia Antipolis

Figura 3: Entrada da Infineon em Sophia-Antipolis.

Localizado no pólo de alta tecnologia de Sophia Antipolis, este centro de Pesquisa

e Desenvolvimento da Infineon é composto de engenheiros, que se dividem conforme os seguintes grupos de atividade:

Soluções Móveis: A missão do Grupo de Soluções Móveis é definir, projetar e trazer para a produção em massa soluções para o mercado de telefonia móvel (GSM/GPRS/EDGE/UMTS).

Desenvolvimento de *Software*: O Grupo de Desenvolvimento de Software desenvolve soluções que ajudam na automatização das etapas do fluxo de concepção de circuitos integrados. Em outras palavras, eles desenvolvem soluções que facilitam a relação entre os engenheiros e as ferramentas de concepção.

Desenvolvimento de Bibliotecas para Projeto de Memórias: Este grupo usa de sua experiência em concepção e simulação de circuitos analógicos, modelagem digital, testes e concepção de máscara para desenvolver bibliotecas a serem usadas no projeto de memórias SRAM de silício de simples e duplo acesso, alta freqüência, baixa potência e alta densidade.

Gerenciamento Global de Licenças: O Grupo GLM Group tem como missão prover licenças EDA a toda comunidade de engenheiros da Infineon em todo o mundo. Este estágio foi desenvolvido como parte das atividades deste grupo.

2 Especificações

A Infineon Technologies possui cerca de 20 Centros de Pesquisa e Desenvolvimento espalhados por todo o mundo. Os engenheiros que trabalham nestes centros usam ferramentas EDA para similar/testar/validar o seu trabalho. Ferramentas EDA, por sua vez, necessitam licenças EDA para poder funcionar. Atualmente, a maioria destas licenças EDA está instalada em Munique, podendo ser usadas a partir de diferentes sítios da Infineon. No entanto, não há nenhuma forma de limitação do número de licenças que podem ser usadas ao mesmo tempo por cada sítio.

O objetivo deste estágio é então propor um sistema de limitação do uso de licenças, de forma que cada sítio Infineon possa limitar o número de licenças que podem ser usadas por seus engenheiros. Este sistema pode ser feito via configuração da plataforma computacional LSF.

Durante o desenvolvimento deste trabalho nós vamos usar LSF versão 6.2 e FlexLM versão 9.2 em um sistema operacional UNIX.

Ao fim do trabalho, as soluções esperadas poderão ser adotadas nos outros sítios Infineon. Estas soluções são:

- Controle de uso de licenças em nível local via configuração LSF
- Desenvolvimento de uma interface gráfica, usando scripts Perl, de forma a permitir que os gerentes de licença possam controlar, de forma fácil, os limites de uso de licenças

3 *Gerenciamento de licenças usando FlexLM e licenças EDA*

O grupo GLM usa o software FlexLM [3] para gerenciamento de licenças EDA. FlexLM classifica as licenças como: servidas/contadas e não-servidas/não-contadas.

3.1 Licenças servidas/contadas

Licenças que necessitam de um servidor, servidor de licenças, para serem instaladas e monitoradas. Licenças Servidas/Contadas podem ser classificadas como:

Licenças flutuantes: São licenças que podem ser usadas por qualquer das máquinas na rede, até que se atinja o limite de licenças instaladas no servidor. Como ilustra a figura 4.

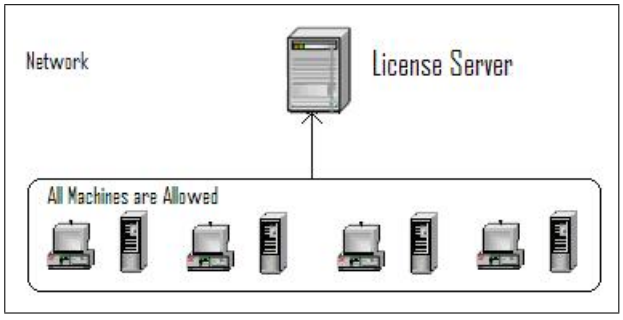

Figura 4: Acesso as licenças flutuantes.

Licenças contadas de nó-bloqueado: São licenças que podem ser usadas unicamente por uma ou um conjunto de máquinas na rede, até que se atinja o limite de licenças instaladas no servidor. Como ilustrado na figura 5.

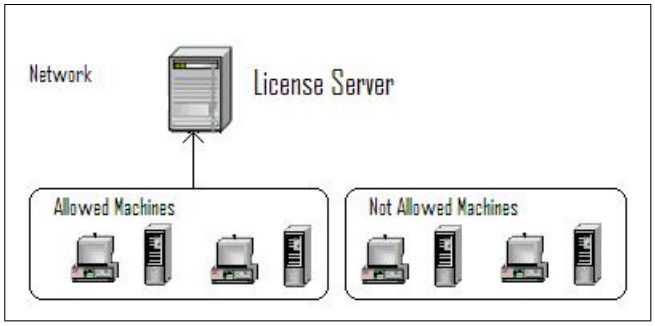

Figura 5: Acesso as licenças contadas de nó-bloqueado.

3.1.1 Componentes FlexLM para licenças servidas/contadas

Licenças servidas/contadas necessitam dos quatro componentes FlexLM seguintes para serem gerenciadas:

- *FlexLM Licensed Application*
- *License Manager Daemon (lmgrd)*
- *Vendor Daemon*
- *License File*

Além destes quatro componentes, existem outros três componentes que podem ser usados opcionalmente pelos gerentes de licença, para facilitar o seu trabalho:

- *Debug Log-File*
- *Report Log-File*
- *End-User Administration Options File*

A figura 6 ilustra todos estes componentes e a maneira como eles estão relacionados uns aos outros.

O componente *FlexLM-Licensed Application* é ligado a um programa chamado *FlexLM Client Library*, que provém a comunicação com o servidor de licença via comunicação TCP/IP. O servidor de licença é composto pelos componentes *License Manager Daemon (lmgrd)* e o *Vendor Daemon*. Para cada vendedor de ferramentas EDA, um *Vendor Daemon* é necessário, por exemplo, a Infineon usa ferramentas da Mentor Graphics, Cadence, Magma, etc. Então, é preciso um *Vendor Daemon* para cada um destes vendedores.

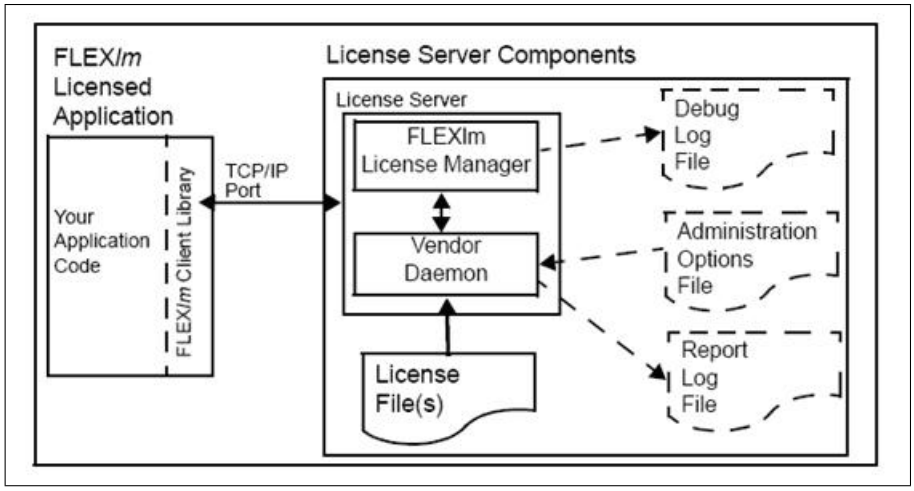

Figura 6: Componentes FlexLM para licenças servidas/contadas.

O processo de pedido de licença é iniciado quando a aplicação, componente *FlexLM-Licensed Application*, é lançada, o primeiro passo é estabelecer a comunicação entre a aplicação e o componente *License Manager Daemon (lmgrd).* lmgrd por sua vez, reconhece a que vendedor a aplicação pertence e em seguida transfere a comunicação para o *Vendor Daemon* daquele vendedor que, em seguida, baseado no *License File*, no *End-User Administrator Options File* e na quantidade atual de licenças em uso, permite ou nega o pedido de licença feito pela aplicação.

O *License File* armazena dados sobre as licenças. Por exemplo, o número de licenças instaladas, o endereço do servidor de licença, a data de expiração destas licenças, se a licença é flutuante ou de nó-bloqueado, etc. Este arquivo é criado pelo vendedor que fornece as licenças e editado pelos gerentes de licença.

O *Debug Log-File* é atualizado pelo servidor de licença e contém mensagens de erro relacionadas aos componentes *License Manager Daemon* e *Vendor Daemon.* Este arquivo é bastante útil aos gerentes de licença em caso de depuração de problemas nos servidores de licença. O *End-User Administrator Options File* é usado para restringir o uso de licenças por um ou um grupo específico de usuários assim como o número máximo de licenças disponíveis no servidor de licença. O *Report Log-File*, se usado, é criado pelo *Vendor Daemon* e contém informação sobre o uso das licenças, basicamente, ele informa quem usa as licenças e onde, em que máquinas, elas estão sendo usadas.

3.2 Licenças não-servidas/não-contadas

Licenças que não necessitam de um servidor de licença para serem instaladas. Estas licenças são normalmente instaladas na própria máquina que usa a licença ou em um diretório comum ao conjunto de máquinas que a utiliza. Licenças não-servidas/não-contadas possuem apenas uma classificação, como segue:

Licenças não-contadas de nó-bloqueado: Licenças que podem ser usadas ilimitadamente por uma ou um grupo de máquinas, como ilustra a figura 7.

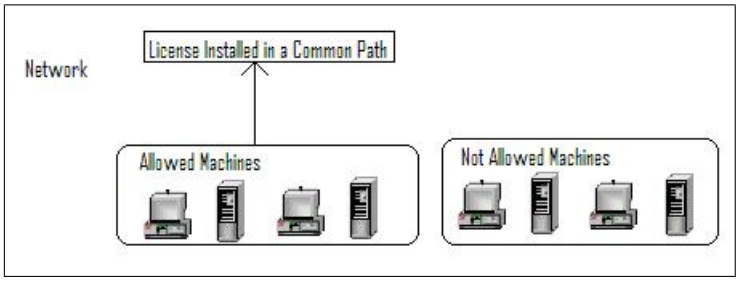

Figura 7: Acesso as licenças não-contadas de nó-bloqueado.

3.2.1 Componentes FlexLM para licenças não-servidas/não-contadas

Licenças não-servidas/não-contadas requerem apenas os seguintes componentes FlexLM para serem gerenciadas:

- *FlexLM-Licensed Application*
- *License File*

A figura 8 ilustra estes componentes.

Haja visto que as licenças não-servidas/não-contadas não requerem um servidor de licença, o processo de pedido de licença é muito mais simples do que para as licenças contadas/servidas. Primeiramente, o componente *FlexLM-Licensed Application* é iniciado, então, as rotinas que compõem o programa *FlexLM Client Library* procuram pelo *License File* e, baseado em seu conteúdo, o uso da licença é permitido ou negado.

Para licenças não-servidas/não-contadas, uma vez que não há um servidor de licença, o *License File* também se torna um pouco mais simples. Este arquivo contém informação acerca da licença em si, por exemplo, a data de expiração, que máquinas são permitidas de usar aquela licença, etc.

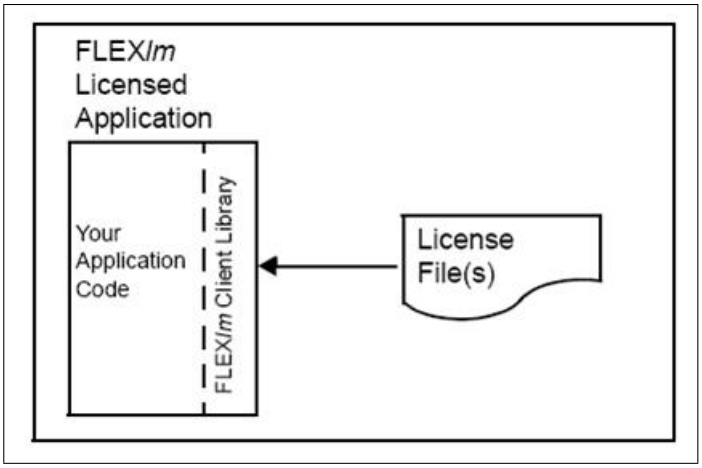

Figura 8: Componentes FlexLM para licenças não-servidas/não-contadas.

3.3 lmstat

FlexLM oferece um comando chamado lmstat, o qual é muito útil aos gerentes de licença quando é necessário checar o atual uso de uma determinada licença fornecida por um vendedor específico. A figura 9 ilustra como o comando lmstat pode ser usado em um ambiente Unix.

```
vlbica13{costagab} lmstat -a -c mentor_8.2_lic -f qhsimvh
```

Figura 9: Exemplo de uso do comando lmstat para a licença qhsimvh.

O comando mostrado na figura 9 é relativo à licença qhsimvh, fornecida pelo vendedor Mentor Graphics. A resposta a esse comando nos informa quem está usando as licenças qhsimvh e de onde estão sendo usadas, a versão destas licenças, em que servidor as licenças estão instaladas e o momento em que as mesmas começaram a ser usadas. Como pode ser visto na figura 10.

3.4 Classificação de licenças por cálculo de custo

O contrato assinado com os vendedores, quando da compra das licenças, define como o cálculo do custo das licenças será efetuado. Dessa forma, podemos classificar as licenças segundo diferentes modelos, que seguem:

Licenças *Pay Per User* (Licenças PPU): O custo destas licenças é baseado em

```
lmstat - Copyright (c) 1989-2005 Macrovision Europe Ltd. and/or Macrovision Corporation. All Rights Reserved.
Flexible License Manager status on Mon 8/18/2008 11:00

License server status: 3225@ulicserv2.muc.infineon.com,3225@ulicserv3.muc.infineon.com,3225@ulicserv1.muc.infineon.com
    License file(s) on ulicserv2.muc.infineon.com: /local/lic/lmadmin/mentor_8.2/license/mentor_8.2_lic:

ulicserv2.muc.infineon.com: license server UP (MASTER) v10.8
ulicserv3.muc.infineon.com: license server UP v10.8
ulicserv1.muc.infineon.com: license server UP v10.8

Vendor daemon status (on ulicserv2.muc.infineon.com):

     mgcld: UP v10.8

Users of qhsimvh:  (Total of 1063 licenses issued;  Total of 219 licenses in use)

  "qhsimvh" v2008.090, vendor: mgcld
  floating license

    Kut vihlc171 vihw4072:16642_2 (v2006.05) (ulicserv2.muc.infineon.com/3225 100391), start Thu 8/14 17:16
    Kut vihlc054 vihw4072:130014_2 (v2006.05) (ulicserv2.muc.infineon.com/3225 66610), start Mon 8/18 10:47
    alingry wien37 vlbica13:3129724_2 (v2006.05) (ulicserv2.muc.infineon.com/3225 132833), start Thu 8/14 10:34
    altieri mucsxbora07 rom40:585615_2 (v2008.02) (ulicserv2.muc.infineon.com/3225 133845), start Mon 8/18 9:29
    arey oslo08 rom40:6719715_2 (v2006.05) (ulicserv2.muc.infineon.com/3225 49456), start Mon 8/18 9:23
    arey oslo06 rom40:6727795_2 (v2006.05) (ulicserv2.muc.infineon.com/3225 42047), start Mon 8/18 10:08
    asa vihlc254 viws87:1131478_2 (v2008.02) (ulicserv2.muc.infineon.com/3225 118234), start Thu 8/7 19:43
    asa vihlc251 viws87:1121643_2 (v2008.02) (ulicserv2.muc.infineon.com/3225 84557), start Mon 8/4 17:30
    auerbern avalon9.lnz.infineon.com localhost:1710547_3 (v2006.05) (ulicserv2.muc.infineon.com/3225 83374), start Thu
 8/14 15:26
```

Figura 10: Resposta ao comando lmstat para a licença qhsimvh.

seu uso. Os vendedores fornecem aos gerentes de licença um número de licenças bem acima do necessário, no entanto, apenas as licenças usadas serão pagas.

Licenças Baseadas no Tempo (Licenças TBL): Licenças que são adquiridas por um período de tempo, que varia normalmente de 3 meses a 3 anos. As licenças serão todas pagas, mesmo senão usadas, durante o período de tempo para a qual foram adquiridas.

Licenças Perpétuas: Licenças que são adquiridas por um longo período de tempo, em geral, 99 anos ou um período mais curto. A manutenção destas licenças também deve ser adquirida. Devido à flexibilidade de contrato, as licenças PPU e TBL são preferíveis as licenças perpétuas.

A maioria das licenças usadas pela comunidade de engenheiros da Infineon é classificada como flutuante e PPU. Dessa forma, a partir de agora vamos nos focar unicamente neste tipo de licença.

4 Plataforma computacional LSF

O projeto de circuitos integrados exige recursos computacionais, em outras palavras, máquinas de alto desempenho. Estes recursos precisam ser gerenciados de alguma forma, no intuito de se evitar que algumas máquinas sejam sobrecarregadas, ao passo que outras estejam sendo subutilizadas. A Infineon usa a plataforma computacional LSF para o gerenciamento de recursos computacionais.

4.1 Conceitos básicos da plataforma LSF

Antes de abordar o funcionamento da plataforma LSF, é preciso que os seguintes conceitos se tornem bastante claros.

- ***Job:*** Qualquer programa de computador/simulação ou comando submetidos à plataforma LSF são referenciados como *jobs*, por exemplo: verificação funcional, solução de problemas complexos, etc.
- ***Host:*** Qualquer máquina que faça parte da rede.
- ***Server host:*** São *hosts* que podem não somente submeter *jobs* à plataforma LSF, mas também recebê-los e executá-los.
- ***Client host:*** São *hosts* que podem apenas submeter *jobs* à plataforma LSF.
- ***Cluster:*** Grupo de *hosts* que, normalmente possuem recursos computacionais desiguais.
- ***Queue:*** Container para o qual os *jobs* são enviados. Em seguida, eles são ordenados, programados e despachados para os *hosts* nos quais serão executados.

4.2 Como a plataforma LSF funciona

A plataforma LSF é basicamente composta de arquivos de configuração e rotinas de *software* chamadas de *daemons* [4]. Estas rotinas são responsáveis por monitorar os *jobs* a partir do momento em que são submetidos à plataforma LSF até o momento em que são finalizados. Já os arquivos de configuração definem os *clusters*, *queues*, definem limites de uso de recursos, criam e mapeiam recursos externos, entre outras ações.

O ciclo do *job*, que compreende desde a submissão do mesmo até a sua execução, pode ser acompanhado na figura 11 e acontece da seguinte maneira: Inicialmente, o *job* é submetido por qualquer dos *hosts* da rede, esse *host* é conhecido como *submission host* e pode ser tanto um *host* do tipo *server host* quanto do tipo *client host*. Em seguida, o *job* é recebido pelo *master host*, que é o *server host* mais importante de todo o sistema, O *master host* coloca o *job* na *queue* especificada pelo usuário, caso nenhuma *queue* seja especificada, o *job* é colocado na *queue default* [5].

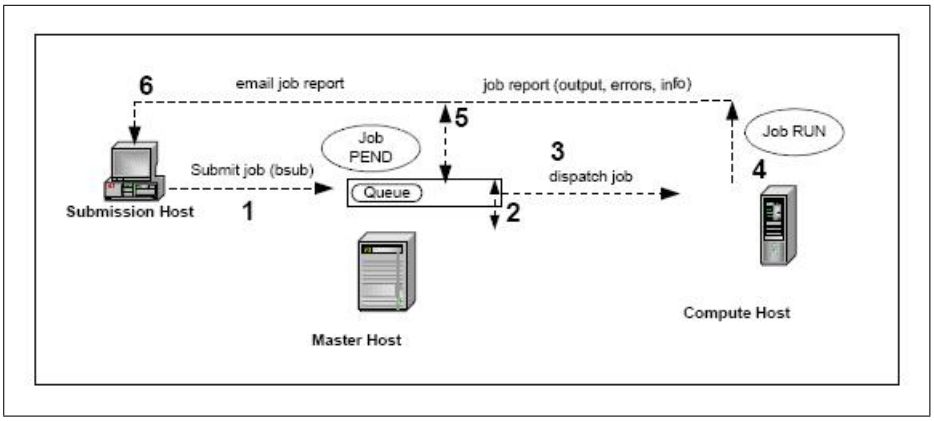

Figura 11: Ciclo do job na plataforma LSF.

Uma vez que o *job* é recebido pela *queue*, os *daemons* verificam a situação de carga de todos os *hosts* do *cluster* e repassa essa informação ao *master host*. Esta situação de carga diz respeito a quão carregados estão os *hosts*, por exemplo, quanto de memória *RAM* cada *host* tem disponível, quanto de espaço de armazenamento em disco, entre outras informações. Caso algum erro seja encontrado no *job*, o mesmo é imediatamente cancelado, como mostrado no passo 5, da figura 11. Caso contrário, o *master host* se baseia na situação de carga de cada *host*, nos limites de uso de recursos definidos nos arquivos de configuração e nos recursos requeridos pelo usuário, para checar se há algum *host* disponível no *cluster* para executar aquele *job*. Caso algum *host* esteja disponível, o *job* é despachado e começa a ser executado imediatamente. Caso contrário, o *job* é

mantido na *queue* até que o *master host* encontre um *host* disponível.

Após ser iniciada a execução do *job*, se algum erro for encontrado, o *job* é imediatamente finalizado, caso contrário ele é executado com sucesso. A plataforma LSF pode ser configurada de forma a enviar uma mensagem de *e-mail* ao usuário que submeteu o *job*, assim que o mesmo seja finalizado, reportando erros e resultados, como mostra o passo 6 da figura 11.

4.3 Como submeter *jobs* na plataforma LSF

Os *jobs* são submetidos na plataforma LSF usando um comando bastante simples, chamado bsub [6]. Como pode ser visto na figura 12.

Figura 12: Exemplo de como submeter um *job* na plataforma LSF.

No exemplo da figura 12, o *job* submetido é o simulador ModelSim da Mentor Graphics. Trata-se de um *job* bastante simples que apenas abre o simulador para que possa ser usado por algum engenheiro, como é visto na figura 13.

4.4 Recursos definidos na plataforma LSF

Todas as decisões tomadas pela plataforma LSF na hora de despachar os *jobs* são, basicamente, baseadas na disponibilidade de recursos. Alguns destes recursos são comuns a todos os *hosts* do *cluster*, por exemplo, memória *RAM*, espaço em disco, memória *flash*, entre outros recursos. Estes recursos são automaticamente reconhecidos pela plataforma LSF e são conhecidos como *built-in resources*. No entanto, qualquer outro recurso externo, por exemplo, licenças instaladas em servidores de licença, podem ser adicionados a plataforma LSF. Estes recursos são conhecidos como *site-defined resources* [7].

4.4.1 Recursos do tipo *site-defined*

Recursos do tipo *site-defined* são classificados como:

Recursos estáticos: O que caracteriza este tipo de recurso, é que a sua quantidade é definida nos arquivos de configuração de LSF e não muda no decorrer do tempo.

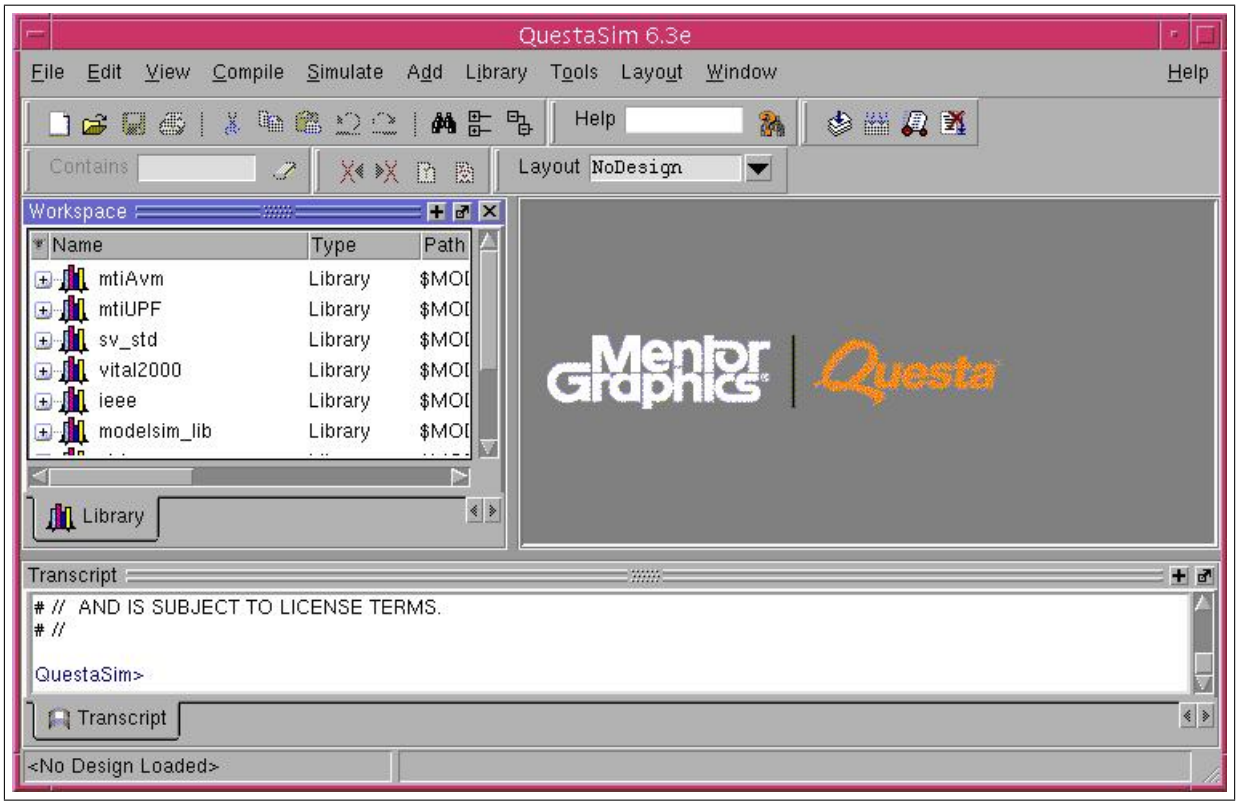

Figura 13: Interface do simulador Modelsim da Mentor Graphics.

Recursos dinâmicos: Diferentemente dos recursos estáticos, para os recursos dinâmicos, a sua quantidade muda no decorrer do tempo. Esta quantidade é atualizada através de um *script* chamado elim, que é fornecido por LSF e pode ser editado pelos administradores da plataforma LSF.

Como já foi visto neste capítulo, a configuração da plataforma LSF é realizada através de arquivos de configuração. Para adicionar recursos do tipo *site-defined* é necessário que se altere dois desses arquivos de configuração, que são: lsf.shared e lsf.cluster.<clustername>. O primeiro arquivo é usado para criar os recursos, ao passo que o segundo é usado para mapeá-los entre os *hosts* do *cluster*.

Imagine agora que você possua licenças chamadas qhsimvh e verilog instaladas em um servidor de licenças, e que você queira representá-las como recursos na plataforma LSF. A seguir, temos dois exemplos ilustrados que mostram como adicionar tanto recursos estáticos como dinâmicos do tipo *site-defined* para representar estas licenças.

Exemplo Ilustrado: Representação de licenças tanto como recursos estáticos quanto como recursos dinâmicos do tipo *site-defined*

O primeiro passo é abrir o arquivo lsf.shared e procurar pela seção chamada *Resource*.

Nesta seção é possível alterar uma série de parâmetros, que irão definir as características do recurso a ser criado. Os parâmetros mais importantes são:

RESOURCENAME: Este parâmetro define o nome do recurso, em nosso exemplo nós iremos usar o nome das licenças que estamos representando, no caso, verilog e qhsimvh.

TYPE: Representa o tipo dos recursos criados. Os valores aceitos pelo parâmetro *TYPE* são *Boolean*, *Numeric* e *String*. Para representar licenças vamos usar o tipo *Numeric*, uma vez que a quantidade de licenças será sempre um valor numérico.

INTERVAL: Este parâmetro representa o período de tempo no qual a quantidade de recursos disponíveis será atualizada pelo *script* elim. Como dito anteriormente, o *script* elim é usado somente para recursos dinâmicos, então, para recursos estáticos não há necessidade de definir o parâmetro *INTERVAL*, é preciso somente abrir e fechar parênteses, como mostra a figura 14.

DESCRIPTION: O parâmetro *DESCRIPTION* é usado para descrever o recurso criado. Para as licenças verilog, por exemplo, nós o definimos como verilog *licenses* para descrever os recursos que representam tais licenças, como é ilustrado na figura 14.

Assim, podemos criar tanto recursos estáticos quanto dinâmicos como ilustram as figuras 14 e 15, respectivamente.

```
Begin Resource
RESOURCENAME      TYPE     INTERVAL       INCREASING      DESCRIPTION
verilog           Numeric    ()             N             (verilog licenses)
qhsimvh           Numeric    ()             N             (qhsimvh licenses)
End Resource
```

Figura 14: Criação de recursos estáticos do tipo *site-defined*.

```
Begin Resource
RESOURCENAME      TYPE     INTERVAL       INCREASING      DESCRIPTION
verilog           Numeric    30             N             (verilog licenses)
qhsimvh           Numeric    30             N             (qhsimvh licenses)
End Resource
```

Figura 15: Criação de recursos dinâmicos do tipo *site-defined*.

Observe que, comparando as figuras 14 e 15, o único parâmetro que muda de uma para a outra é o parâmetro *INTERVAL*. Para recursos dinâmicos, a quantidade destes recursos é atualizada pelo *script* elim a cada 30 segundos, para os recursos estáticos, a

quantidade de recursos é definida no arquivo de configuração lsf.cluster.<clustername> como veremos.

Uma vez que os recursos forem criados, é preciso mapeá-los entre os *hosts* do *cluster*, o que significa dizer quais *hosts* dentro do *cluster* serão autorizados a usufruir de tais recursos. Inicialmente, abrimos o arquivo lsf.cluster.<clustername> e buscamos pela seção *ResourceMap*. Assim como a seção *Resource* do arquivo lsf.shared, a seção *ResourceMap* também possui alguns parâmetros a serem definidos, que são:

RESOURCENAME: Identifica o nome dos recursos a serem mapeados. Estes recursos devem ter sido previamente criados no arquivo de configuração lsf.shared.

LOCATION: Caso os recursos a serem mapeados sejam estáticos, o parâmetro *LOCATION* define tanto a quantidade de cada recurso como a lista de *hosts* autorizados a usá-los. No entanto, para recursos dinâmicos, este parâmetro define apenas a lista de *hosts* autorizados, pois, como já vimos, sua quantidade é atualizada pelo *script* elim.

Pode-se então definir 10 amostras de cada recurso estático e, autorizar todos os *hosts* do *cluster* a usá-las, da forma como segue, mostrada na figura 16.

```
Begin ResourceMap
RESOURCENAME            LOCATION
verilog                 (10@[all])
qhsimvh                 (10@[all])
End ResourceMap
```

Figura 16: Mapeamento de recursos estáticos do tipo *site-defined*.

Para os recursos dinâmicos, é necessário editar o *script* elim para atualizar a quantidade de recursos disponíveis. Neste exemplo, estamos usando recursos para representar licenças que estão instaladas em um servidor de licenças. Dessa forma, o *script* elim pode usar o comando lmstat (ver seção 3.3 do capítulo 3) para consultar a disponibilidade de tais licenças e, em seguida, atualizar a quantidade de recursos a partir da quantidade de licenças disponíveis. Como ilustra a figura 17.

Agora, só precisamos definir no arquivo lsf.cluster.<clustername>, quais os *hosts* que serão autorizados a utilizar destes recursos. No exemplo mostrado, os recursos dinâmicos criados podem ser usados por todos os *hosts* do *cluster*, como ilustra a figura 18.

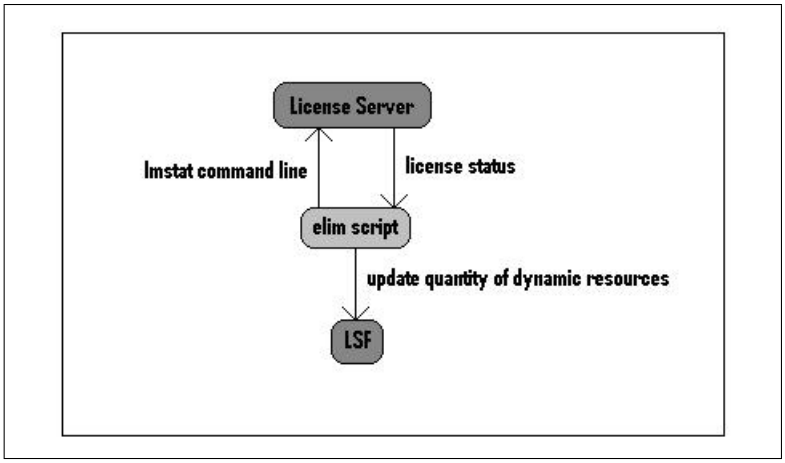

Figura 17: Atualização da quantidade de recursos dinâmicos.

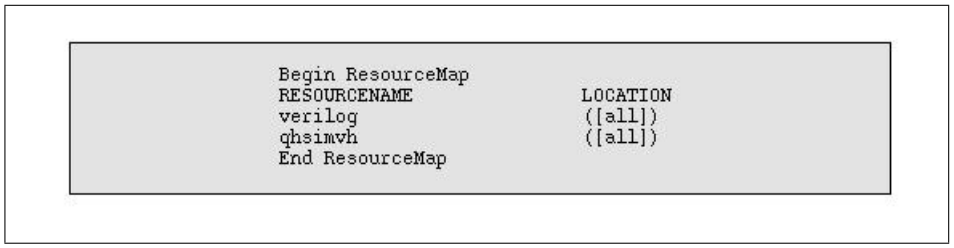

Figura 18: Mapeamento de recursos dinâmicos do tipo *site-defined*.

4.4.2 Requerimento de recursos na submissão de *jobs*

Qualquer recurso definido na plataforma LSF pode ser requerido pelos usuários quando da submissão de jobs [8]. Para tanto, é necessário adicionar no comando bsub, a opção rusage. Por exemplo, para requerer uma amostra do recurso verilog, que acabamos de adicionar a plataforma LSF no exemplo anterior, basta usar o comando explicitado na figura 19.

```
vlbica13{costagab} bsub -R "rusage[verilog=1]" my_job
```

Figura 19: Submissão de *job* requerendo por recurso verilog.

No caso em que queiramos requerer por ambos os recursos, verilog e qhsimvh, devemos usar o comando ilustrado na figura 20, por exemplo.

```
vlbica13{costagab} bsub -R "rusage[verilog=1:qhsimvh=1]" my_job
```

Figura 20: Submissão de *job* requerendo por recursos verilog e qhsimvh.

5 Limitação do uso de licenças

5.1 Objetivo da limitação do uso de licenças

A maioria das licenças adquiridas pela Infineon é instalada em servidores de licença e, podem ser acessadas paralelamente por qualquer *host* de qualquer *cluster* a partir dos diferentes centros de pesquisa e desenvolvimento da Infineon, como observado na figura 21. Estas licenças são classificadas por FlexLM como sendo licenças flutuantes (Ver capítulo 3) ou [3].

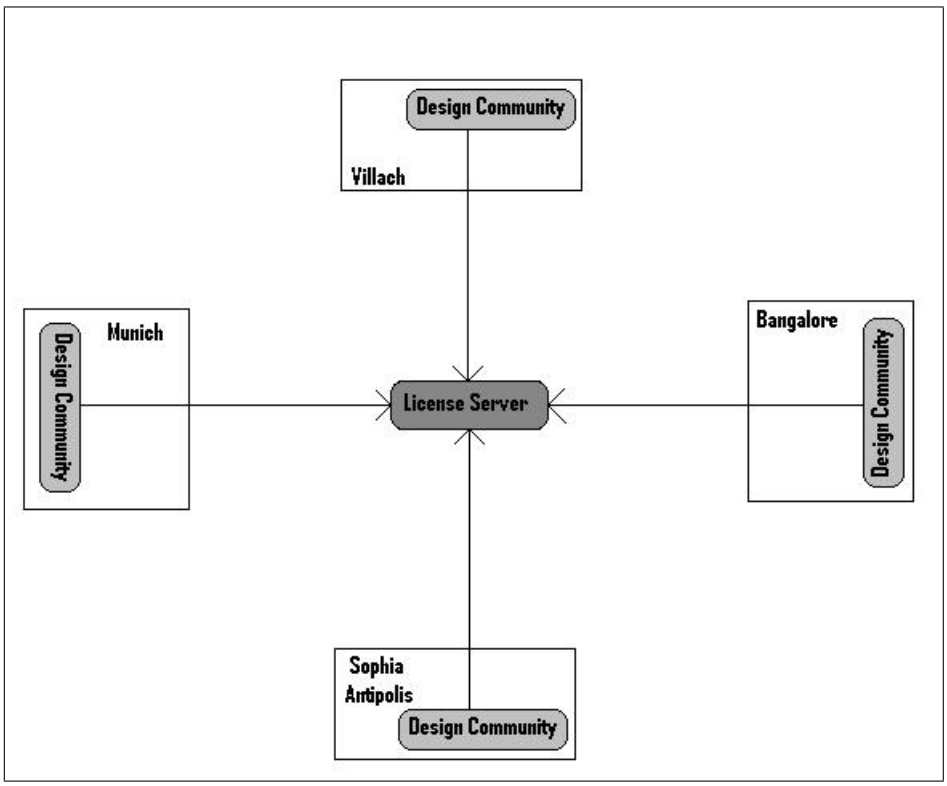

Figura 21: Acesso as licenças flutuantes desde diferentes centros de P&D.

O pico de uso de licenças significa o número máximo de licenças usadas, em paralelo, por um ou por diferentes usuários. Este pico, observado no período de um dia é chamado de pico diário de uso de licenças. A figura 22 ilustra alguns destes picos, para um dada

licença, observados num período total de 3 meses.

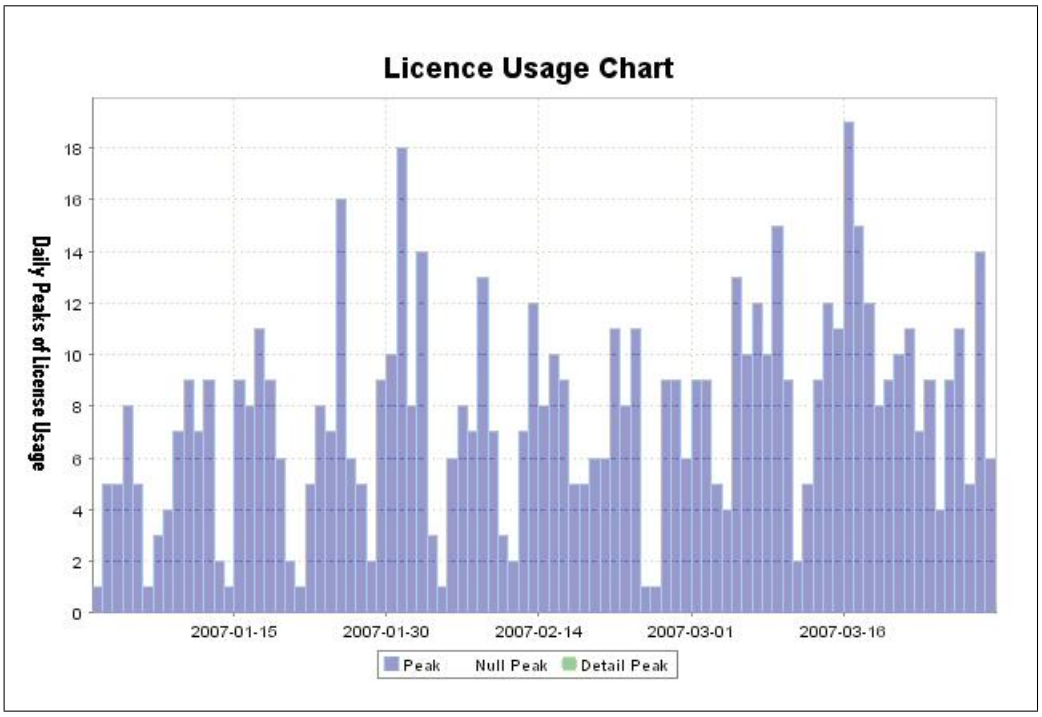

Figura 22: Picos diários de uso de licenças num período de 3 meses.

O cálculo do custo da maioria das licenças usadas pela Infineon é baseado nestes picos diários. Então, altos picos diários de uso significam altos custos.

O objetivo da limitação do uso de licenças é reduzir custos através da redução destes picos diários. A idéia é liberar apenas uma quantidade limitada de licenças a serem usadas em paralelo, mesmo quando há muito mais licenças disponíveis nos servidores de licença do que as licenças liberadas para uso.

O gráfico ilustrado na figura 23 representa como deveria ser o gráfico da figura 22 caso houvesse um sistema de limitação do uso de licenças, neste caso, por exemplo, limitando o uso o pico diário em 12 licenças. O uso de licenças representado em vermelho representa o uso de licenças de um dia que teria de ser repassado ao dia subseqüente.

5.2 Limitação do uso de licenças em diferentes níveis

A limitação do uso de licenças pode ser aplicada em diferentes níveis, que são: global e local [9].

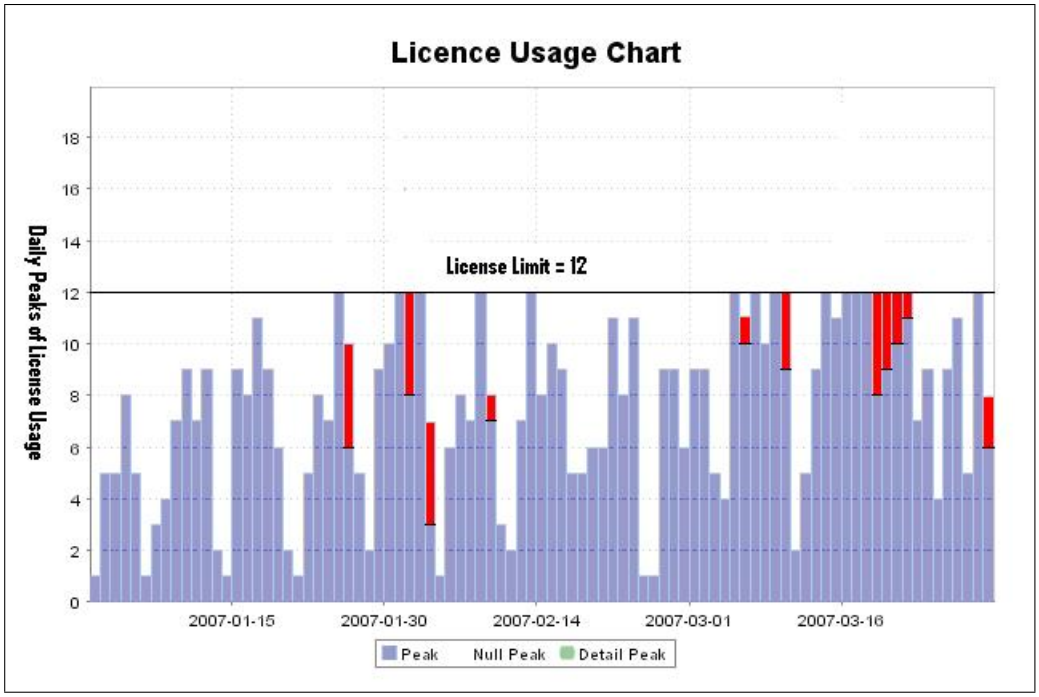

Figura 23: Picos diários de uso de licenças limitados a 12 licenças.

5.2.1 Limitação do uso de licenças à nível global

Quando os limites aplicados são os mesmos para todos os centros de P&D, diz-se da limitação e nível global. Por exemplo, se tivermos 1000 licenças do tipo qhsimvh (usadas pelo simulador Modelsim) instaladas no servidor de licença e o seu uso é limitado a 200 licenças para todo o conjunto de centros de P&D, este tipo de limitação é a nível global. Esse tipo de limitação pode ser mais bem entendido com ajuda da figura 24.

No escopo deste trabalho não iremos nos focar neste tipo de limitação de uso de licenças. No entanto, nos anexos deste relatório é descrito como a limitação a nível global pode ser implementada.

5.2.2 Limitação do uso de licenças a nível local

Como pode ser visto na figura 25, a limitação do uso de licenças a nível local permite que cada centro de P&D da Infineon controle seus próprios limites de uso de licenças. Por exemplo, tomando as mesmas 1000 licenças qhsimvh do exemplo anterior, poderíamos limitá-las da seguinte forma: 40 licenças disponíveis para o centro de Sophia-Antipolis, para o de Munich, 50 para o de Bangalore e 60 para o de Villach. Neste caso, teríamos uma limitação a nível local.

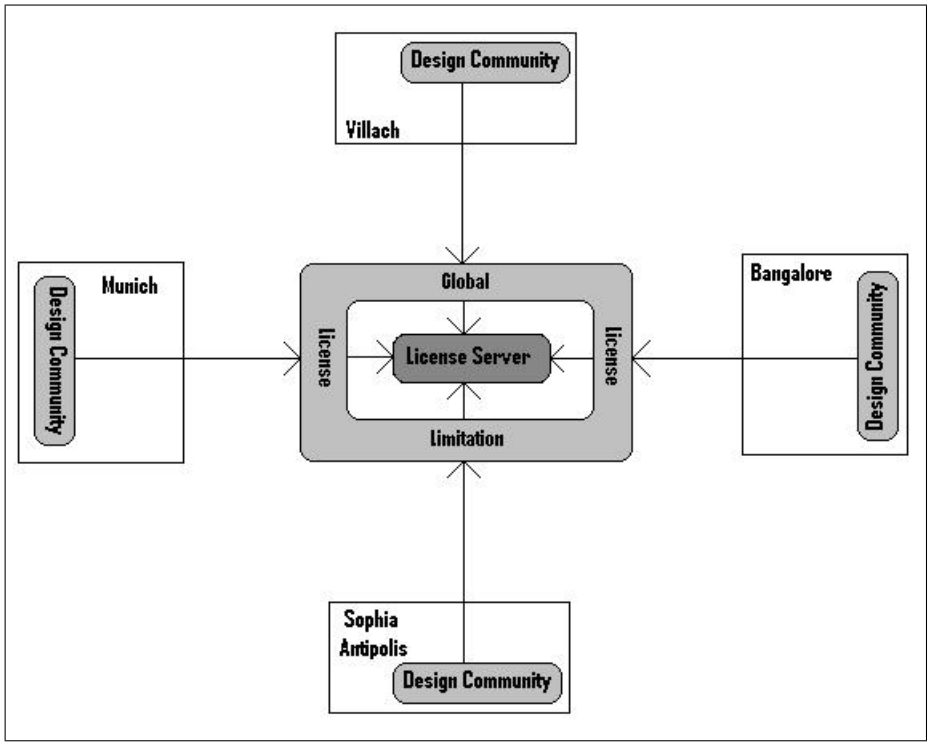

Figura 24: Limitação do uso de licenças a nível global.

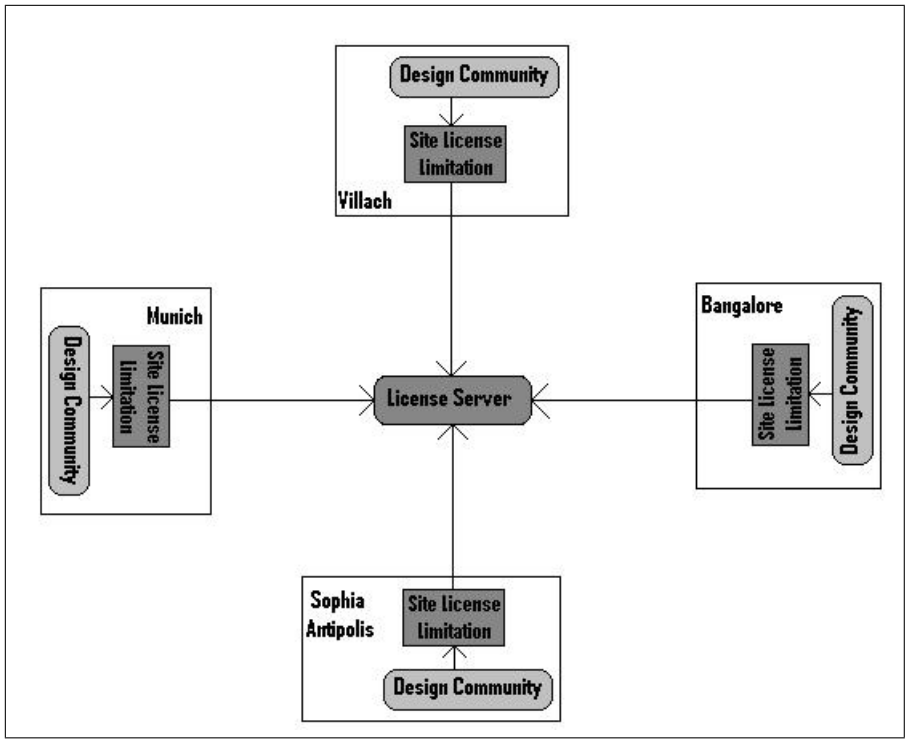

Figura 25: Limitação do uso de licenças a nível local.

Este é o tipo de limitação que focamos durante este estágio e que será apresentado neste relatório. O capítulo 6 traz a solução proposta e todos os passos necessários para a sua implementação.

6 *Solução proposta e implementação*

6.1 Conceito de *license-token*

Apresentamos no capítulo 4, os recursos LSF, em particular, os recursos LSF do tipo *site-defined*. Também foi visto como adicionar recursos como estes à plataforma LSF. Veja mais em [4].

As licenças instaladas nos servidores de licença podem ser então representadas por recursos do tipo *site-defined*, adicionados a plataforma LSF (Veja seção 4.4.1 do capítulo 4). Estes recursos serão chamados a partir de agora de *license-tokens*. Por exemplo, se adicionarmos um recurso chamado qhsimvh a plataforma LSF com uma quantidade de 30, para representar 30 licenças qhsimvh que estão instaladas no servidor de licença, estamos criando 30 *license-tokens* para a licença qhsimvh.

Observe que o número de *license-tokens* criados não precisa ser o mesmo número de licenças instaladas no servidor. Por exemplo, mesmo se tivermos 500 licenças verilog instaladas, podemos criar apenas 30 *license-tokens* para representar apenas 30 destas 500 licenças instaladas.

Os *license-tokens* são criados localmente, uma vez que cada centro de P&D da Infineon possui seu próprio *cluster* LSF. O que significa que podemos criar, por exemplo, 30 *license-tokens* para a licença qhsimvh no centro de Sophia-Antipolis, 60 no de Munich, 40 no de Villach, 50 no de Bangalore e assim por diante, como ilustrado na figura 26.

6.2 Solução proposta

A solução proposta consiste em criar *license-tokens* para as licenças que desejamos limitar. A partir dai, os *jobs* devem ser submetidos requerendo os *license-tokens* referentes

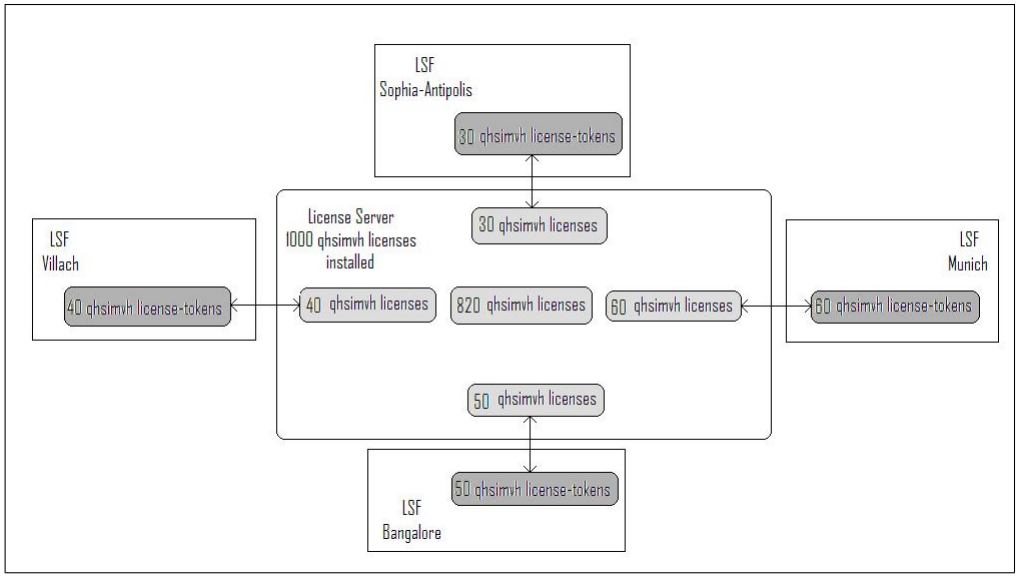

Figura 26: *License-tokens* de licenças qhsimvh em diferentes centros.

as licenças que serão usadas na execução do *job*.

Quando LSF recebe o *job*, ele é inicialmente colocado em estado de *PEND* enquanto o *master host* checa a disponibilidade dos *license-tokens* requeridos e também se todos os outros limites/regras definidos nos arquivos de configuração de LSF estão sendo obedecidos. Caso haja disponibilidade, o *job* é despachado imediatamente para o *host* escolhido mudando seu estado de *PEND* para *RUN*, caso contrário, o *job* mantém seu estado de *PEND* até que haja disponibilidade dos *license-tokens* requeridos. Veja mais no capítulo 4 ou em [4].

A parte final da solução proposta é realizada pelo *software* gerenciador de licenças, FlexLM, e consiste da comunicação entre o *host* escolhido para executar o *job* e o servidor de licença, no intuito de obter a*s* licença*s* que serão usadas por esse *job*, como descrito no capítulo 3 e mais detalhadamente em [3]. A figura 27 ajuda no entendimento de como funciona a solução proposta.

Vamos agora imaginar que criamos 10 *license-tokens* para representar 10 licenças specman e que, em seguida, 11 *jobs* são submetidos, os quais requerem, cada um, 1 *license-token* da licença specman. Nesta situação, o sistema funcionaria como ilustrado na figura 28.

Podemos ver que quando o décimo primeiro *job* é submetido, não existe mais nenhum *license-token* do tipo specman disponível. Dessa forma, este *job* fica no estado de *PEND* até que um dos dez primeiros *jobs* seja finalizado e libere um *license-token* specman. Na

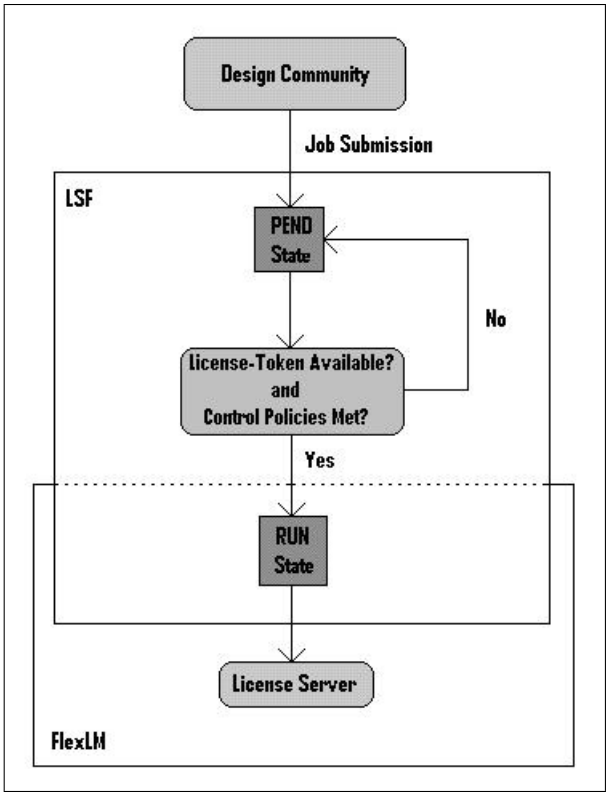

Figura 27: Solução proposta para a limitação do uso de licenças a nível local.

figura 28, vemos que o *job* 1 é o primeiro a ser finalizado, dessa forma, no instante T12, o *job 11*, que até então estava em estado de *PEND*, é despachado e executado no *host* escolhido pela plataforma LSF.

6.3 Testes usando o *cluster* vlbglm

No centro de P&D de Sophia-Antipolis, existem 2 *clusters* diferentes definidos nos arquivos de configuração de LSF, estes receberam os nomes de vlb e vlbglm. O primeiro deles é o mais importante, pois é ele que contém os *hosts* de mais alto desempenho, usados pela comunidade de engenheiros projetistas. Dessa maneira, qualquer modificação na configuração do *cluster* vlb deve ser feita de forma bastante cuidadosa no intuito de não prejudicar *jobs* importantes. O segundo *cluster*, vlbglm, é usado exclusivamente pelo grupo GLM. Assim, foi decidido começar os testes pelo *cluster* vlbglm, por ser um *cluster* que recebe *jobs* menos importantes do que aqueles recebidos pelo *cluster* vlb [10].

Teste 1: Requerimento de *license-token* de forma manual

Inicialmente criamos 2 *license-tokens* para licenças qhsimvh e mais 2 para licenças

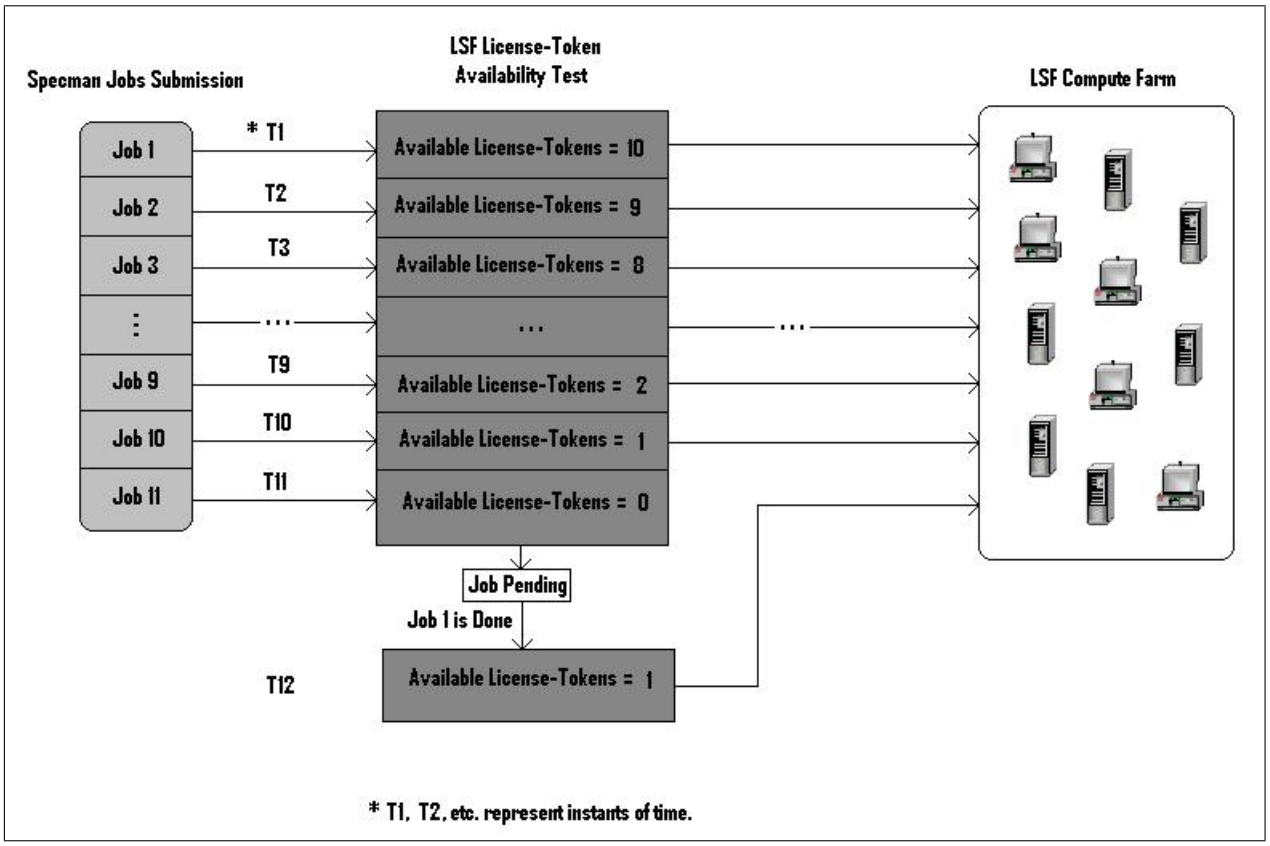

Figura 28: Exemplo de funcionamento da solução proposta.

msimhdlmix. Definimos estes *license-tokens* como qhsimvh_token e msimhdlmix_token, respectivamente. Decidimos criar esses recursos como sendo dinâmicos do tipo *site-defined* para evitar usar o *script* elim, pois este *script* precisaria se comunicar freqüentemente com os servidores de licença, usando o comando lmstat, podendo sobrecarregá-los, tornando-os mais instáveis, o que não é desejável de forma alguma.

Como já discutimos na seção 4.4.1 do capítulo 4, os *license-tokens* podem ser criados editando os arquivos de configuração lsf.shared e lsf.cluster.<clustername>. Como usamos o *cluster* vlbglm em nossos testes, editamos então os arquivos lsf.shared e lsf.cluster.vlbglm. As figuras 29 e 30 ilustram as modificações necessárias feitas nesses dois arquivos.

lsf.shared

```
Begin Resource
RESOURCENAME            TYPE     INTERVAL  INCREASING      DESCRIPTION
qhsimvh_token           Numeric    ()          N           (tokens for feature qhsimvh - mentor_8.2)
msimhdlmix_token        Numeric    ()          N           (tokens for feature msimhdlmix - mentor_8.2)
End Resource
```

Figura 29: Criação dos *license-tokens* qhsimvh_token e msimhdlmix_token.

lsf.cluster.vlbglm

```
Begin ResourceMap
RESOURCENAME                        LOCATION
qhsimvh_token                       (2@[all])
msimhdlmix_token                    (2@[all])
End ResourceMap
```

Figura 30: Mapeamento dos *license-tokens* qhsimvh_token e msimhdlmix_token.

As licenças qhsimvh e msimhdlmix são usadas pelo simulador Modelsim, então, nosso teste consistiu em submeter 3 *jobs* modelsim para LSF, usando o comando bsub e requerendo os *license-tokens* de ambas as licenças, ou seja, qhsimvh_token e msimhdlmix_token para cada *job*.

Como criamos apenas 2 *license-tokens* para cada licença, o terceiro *job* deve supostamente ficar em estado de *PEND*, até que um dos dois primeiros *jobs* seja finalizado. Em seguida, o terceiro *job* se inicia automaticamente [11].

Quando os 3 *jobs* são submetidos, a gente pode observar na figura 31 que os dois primeiros *jobs* reservam os 2 *license-tokens*, qhsimvh_token e msimhdlmix_token, disponíveis para as licenças qhsimvh e msimhdlmix, respectivamente.

Figura 31: *License-tokens* reservados por *jobs* Modelsim.

A figura 32 mostra a lista de *jobs*, na qual podemos observar que os dois primeiros *jobs* estão em estado de *RUN* ao passo que o terceiro permanece em estado de *PEND*.

Figura 32: Terceiro *job* Modelsim em estado de *PEND*.

Quando o primeiro *job* é finalizado, o terceiro passa do estado de *PEND* para o de *RUN*, como podemos evidenciar na figura 33.

Figura 33: Terceiro *job* se inicia quando um dos dois primeiros é finalizado.

O teste 1 é então validado. No entanto, a maioria dos engenheiros projetistas da Infineon não usa o comando bsub para submeter seus *jobs*, assim como fizemos em nosso teste. Eles, ao contrário, usam o fluxo de concepção da Infineon para submeter seus *jobs*, que é abordado na próxima seção.

6.3.1 Fluxo de concepção da Infineon

O fluxo de concepção desenvolvido e usado pelos engenheiros da Infineon é chamado *Inway Design Flow*. O objetivo deste fluxo de concepção é facilitar a vida dos engenheiros

em seus projetos. O fluxo de concepção *Inway* nos permite criar projetos e subprojetos e ainda, configurar os subprojetos a depender das ferramentas EDA a serem usadas pelo projeto. A versão atual do fluxo *Inway* é 5.2.2 [12].

O fluxo *Inway* usa subfluxos para ligar os engenheiros projetistas às ferramentas EDA de forma mais simples, através de simples comandos previamente definidos. Por exemplo, para usar a ferramenta Modelsim, inicialmente precisamos adicionar o subfluxo modelsim ao subprojeto, no qual se trabalha. Em seguida, para submeter *jobs* modelsim para a plataforma LSF usamos apenas o comando iwmsim ao invés de usarmos o comando bsub seguido de vários parâmetros.

O fluxo de concepção Inway é composto de vários *scripts* e arquivos de configuração. Para nós, o mais importante deles é um arquivo de configuração chamado ExecHandler.xml. Este arquivo nos permite relacionar o comando, correspondente a cada subfluxo, a um perfil de requerimento de recursos. Por exemplo, para o comando iwmsim, existirá um perfil de requerimento de recursos definido em ExecHandler.xml de forma que cada *job* submetido usando este comando usará este perfil previamente especificado [13].

O nosso interesse em usar o arquivo ExecHandler.xml é em configurá-lo, de maneira a adicionar ao perfil de requerimento de recursos, o pedido de *license-tokens*. Dessa forma, os usuários não necessitarão fazê-lo de forma manual. A seguir, temos uma breve explicação sobre o arquivo ExecHandler.xml e suas diversas seções.

6.3.1.1 Arquivo ExecHandler.xml

O arquivo ExecHandler.xml pode ser criado tanto a nível local, nível de sítio, quanto a nível de projeto. No primeiro caso, as configurações em ExecHandler.xml serão aplicadas para todos os *jobs* submetidos usando o fluxo de concepção *Inway*. Para o segundo caso, as configurações são aplicadas apenas aos *jobs* submetidos dentro do escopo daquele projeto.

O arquivo ExecHandler.xml é composto de 3 seções, que são: site_def, job_def e os_def [14].

Seção site_def: Esta seção é usada para declarar variáveis globais. Os valores destas variáveis serão definidos nas outras duas seções do arquivo ExecHandler.xml. Na figura 34 podemos ver, por exemplo, a definição das seguintes variáveis: mem (Memória *RAM*), swp (Memória virtual), tmp2 (Memória livre em disco), tipo e versão dos sistemas operacionais, dentre outras.

Pode-se também declarar variáveis customizadas nesta seção, para serem usadas na

```
<definition name="LS_RES_RAM" filter="LSF">mem >= %d</definition>
<definition name="LS_RES_SWAP" filter="LSF">swp>=%d</definition>
<definition name="LS_RES_TMPSPACE" filter="LSF">tmp2>=%d</definition>
<definition name="LS_MODULE_LSF">lsf</definition>
<definition name="LS_KILL" filter="LSF">bkill</definition>
<definition name="LS_CMD_INTERACTIVE" filter="LSF">bsub -Is</definition>
<definition name="LS_CMD_INTERACTIVE_EXEC" filter="LSF">bsub -Is</definition>
<definition name="LS_CMD_DISPATCH" filter="LSF">bsub</definition>
<definition name="LS_CMD_SHELL" filter="LSF">bsub -Is</definition>
<definition name="LS_CMD_CUSTOM" filter="LSF">bsub -I</definition>
<definition name="OS_TYPE">type == %s</definition>
<definition name="OS_VERSION">osrel==%s</definition>

<definition name="LIC_titanak_select">titanak>=%d</definition>
<definition name="LIC_titanak_rusage">titanak=%d</definition>
```

Figura 34: Parte da seção site_def do arquivo ExecHandler.xml.

seção job_def. Na parte inferior da figura 34, pode-se ver que criamos a variável titanak, que será usada na seção job_def para requerer por *license-tokens* da licença titanak.

Seção job_def: Esta seção cria perfis de requerimento de recursos LSF e os associa aos *jobs*. Na figura 35, podemos ver o perfil criado para os *jobs* titan, que usam licenças titanak. Esse perfil é identificado por um *job ID*, neste caso, TITAN.

```
<entry name="TITAN">
  <option name="LS_SYSTEM">LSF</option>
  <option name="QUEUE">background</option>
  <option name="LICENSE">
    <member>LIC_titanak=1</member>
  </option>
  <option name="RUN_OS">
    <member>LINUX30</member>
    <member>SOLARIS</member>
  </option>
</entry>
```

Figura 35: Parte da seção job_def do arquivo ExecHandler.xml.

Como se pode ver na figura 35, os *jobs* titan requerem pela *queue background*, um *license-token* titanak, e uma máquina que possua seja o sistema operacional Linux 3.0 ou Solaris.

Seção os_def: A última seção é usada para declarar todos os sistemas operacionais disponíveis e suas versões. A figura 36 ilustra uma parte desta seção, onde é declarado o sistema operacional Solaris nas versões 5.8 e 5.9.

Teste 2: Requerimento de *license-token* definido no arquivo ExecHandler.xml

Inicialmente, contatamos o pessoal do grupo de tecnologia da informação da Infineon, para lhes pedir que nos criassem um projeto de testes, pois para usar o fluxo de concepção

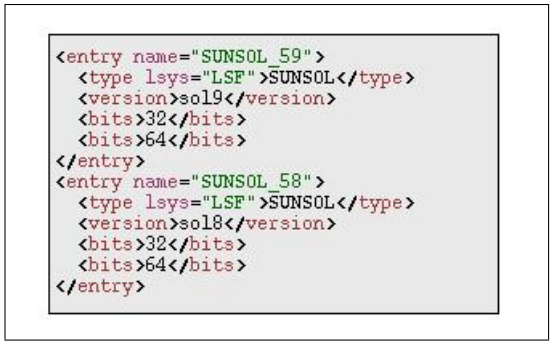

Figura 36: Parte da seção os_def do arquivo ExecHandler.xml.

Inway é necessário estar conectado em um projeto. Uma vez que este projeto foi criado, nós o configuramos de forma a usar o subfluxo iwmsim para submeter nossos *jobs* de teste, e criamos também um arquivo ExecHandler.xml a nível de nosso projeto.

No teste 2, usamos os mesmos *license-tokens* anteriormente criados para o teste 1, qhsimvh_token e msimhdlmix_token. Como já esperávamos, para modificar os perfis de requerimento de recursos de forma a adicionar o requerimento de *license-token*, precisamos modificar as seções site_def e job_def do arquivo ExecHandler.xml. Inicialmente declaramos a variável ModelsimAny, da forma como ilustra a figura 37.

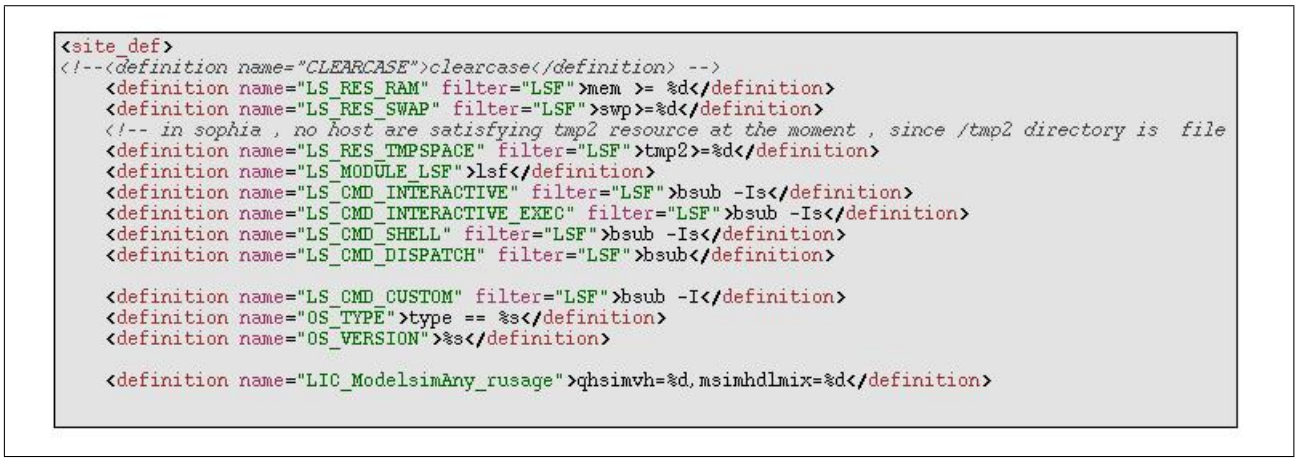

Figura 37: Declaração da variável ModelsimAny na seção site_def.

Em seguida, modificamos os *job ID's* referentes aos *jobs* modelsim como é ilustrado na figura 38.

O teste consiste em submeter 3 *jobs* modelsim, o mesmo feito no teste 1. No entanto, desta vez, os *jobs* serão submetidos usando o fluxo de concepção *Inway*, e o requerimento de *license-tokens* é feito automaticamente, como definido no arquivo ExecHandler.xml.

```
<entry name="MODELSIM_SOLARIS">
  <option name="LS_SYSTEM">LSF</option>
  <option name="QUEUE">batch</option>
  <option name="RUN_OS">
    <member>SUNSOL</member>
  </option>
  <option name="LICENSE">
    <member>LIC_ModelsimAny=1</member>
  </option>
</entry>

<entry name="MODELSIM_BATCH">
  <option name="LS_SYSTEM">LSF</option>
  <option name="QUEUE">batch</option>
  <option name="LICENSE">
    <member>LIC_ModelsimAny=1</member>
  </option>
</entry>
```

Figura 38: Adição de pedido dos *license-tokens* qhsimv_test e msimhdlmix_test.

Figura 39: *License-tokens* reservados por *jobs* Modelsim.

A figura 40 ilustra a lista de *jobs*, na qual observamos que os dois primeiros *jobs* estão em estado de *RUN*, enquanto que o terceiro é mantido em estado de *PEND*.

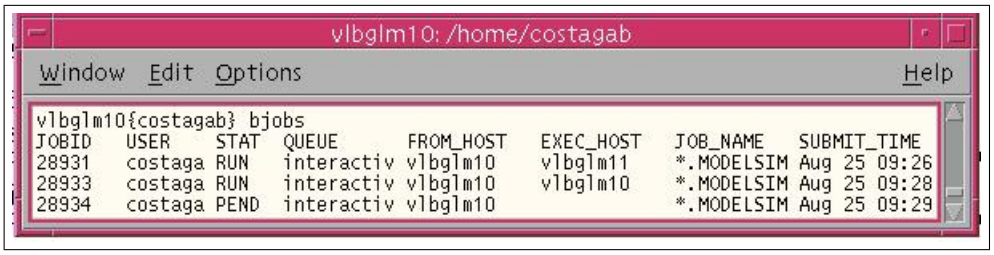

Figura 40: Terceiro *job* Modelsim em estado de *PEND*.

Quando o primeiro *job* é finalizado, o terceiro *job* passa do estado de *PEND* para o de *RUN* e começa a ser executado, como é evidenciado na figura 41.

O próximo passo do nosso trabalho foi então definir, de forma conjunta com o grupo GLM, quais seriam as licenças a serem limitadas. A partir daí, poderíamos então expandir o teste 2 para as licenças escolhidas. Após uma reunião, me foi passado uma lista com as licenças mais caras. Essa lista é explicitada na figura 42.

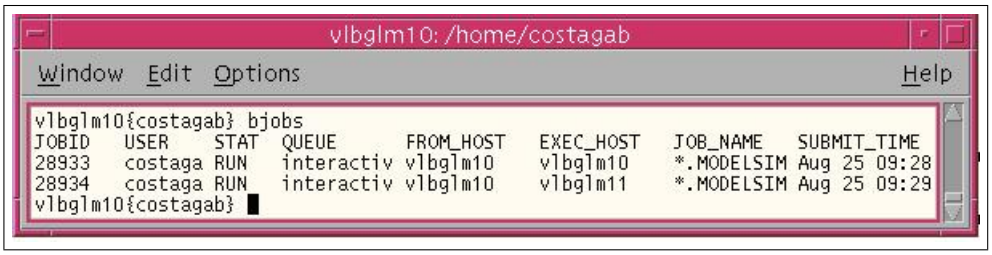

Figura 41: Terceiro *job* se inicia quando um dos dois primeiros é finalizado.

Nome da Licença
titanak
qhsimvh
Design-Compiler
PrimeTime
BLAST_VIEW
testkomp
calibrelvs
calibredrc
msimhdlmix
nanosim
Comet
CoCentric-SYS-Simulator
CoCentric-SYS-DesignCenter
TALUS_VORTEX
STAR-RC2
specman

Figura 42: Lista das licenças mais caras.

Tentamos repetir o teste 2 para todas as licenças listadas na figura 42. No entanto, algumas delas não puderam ser testadas, ou por conta do tempo que se mostrou insuficiente ao fim do estágio, ou pelo fato de não termos conseguido usar os subfluxos do fluxo de concepção *Inway* em conjunto com as ferramentas que utilizavam essas licenças para que pudéssemos testá-las. Dessa maneira, nós nos concentramos na lista de licenças presente na figura 43, que foram as licenças que conseguimos testar.

Nome das Licenças Testadas
titanak
qhsimvh
Design-Compiler
PrimeTime
BLAST_VIEW
testkomp
msimhdlmix
specman

Figura 43: Lista das licenças que foram testadas usando o teste 2.

6.4 Desenvolvimento da solução proposta em Sophia-Antipolis

Após realizar o teste 2, visto anteriormente, com a lista de licenças ilustrada na figura 43, nós estávamos prontos para aplicar a solução que propusemos em todo centro de P&D de Sophia-Antipolis. Decidimos iniciar pela licença specman devido ao número de usuários da mesma, apenas 3 em todo o centro de Sophia-Antipolis. Assim, em caso de problemas, os impactos seriam menos graves.

A primeira tarefa a ser realizada era pedir ao diretor do centro de P&D, o direito de fazer as modificações necessárias de forma a implantar a solução encontrada. Em seguida, deveríamos contatar todos os usuários de licenças specman para informá-los sobre as mudanças que seriam feitas e questioná-los sobre a quantidade de licenças specman que eles tinham necessidade, com esse dado era possível saber quantos *license-tokens* criar. Enfim, pudemos começar as modificações, que são descritas detalhadamente na próxima seção.

6.4.1 Limitação para licenças specman

Primeiramente, criamos os *license-tokens* para representar as licenças specman. Criamos 16 *license-tokens* como previamente havia sido concordado com os usuários specman. Diferentemente do que tínhamos em nossos primeiros testes, dessa vez, os *license-tokens* foram criados no *cluster* vlb, que é o *cluster* principal e que comporta as máquinas de alto desempenho.

A plataforma LSF foi então reiniciada para que as modificações nos seus arquivos de configuração pudessem surtir efeito. O arquivo ExecHandler.xml a nível de sítio, foi também modificado de forma que os *jobs* specman, submetidos usando o fluxo *Inway*, requeressem automaticamente pelo *license-token* specman, como descrito no teste 2.

Após a implantação de todas essas mudanças, nós monitoramos os *jobs* specman durante alguns dias, para estarmos certos que tudo estava correndo como esperado. Depois de comprovar que realmente estava indo como esperado, começamos a investigar a possibilidade de limitarmos o uso de licenças também por usuário. De maneira a evitar que um só usuário reserve todos os *license-tokens* disponíveis enquanto outros usuários também precisem usá-los. Para as licenças specman, havíamos criado 16 *license-tokens* e queríamos limitar para 6 o número de *license-tokens* por usuário usados ao mesmo tempo. A seção seguinte explica como o fizemos.

6.4.1.1 Limitação por usuário para licenças specman

Em [4], descobrimos que entre os arquivos de configuração LSF, existe um arquivo chamado lsb.resources, o qual podemos editar para limitar, por usuário, o uso dos recursos LSF. Pudemos até mesmo definir janelas de tempo, dentro das quais as limitações aplicadas eram variáveis, como veremos na figura 45.

Então, para os *license-tokens* specman, definimos as janelas de tempo com suas respectivas limitações, ilustradas na figura 44.

Janela de Tempo	Limitação
Segunda-feira a Sexta-feira (das 8 a.m as 8 p.m)	6 License-tokens por usuário
Todos os dias (das 8 p.m as 8 a.m)	16 License-tokens por usuário

Figura 44: Janelas de tempo e limitação por usuário para *license-tokens* specman.

Da forma ilustrada na figura 45, editando o arquivo lsb.resources.

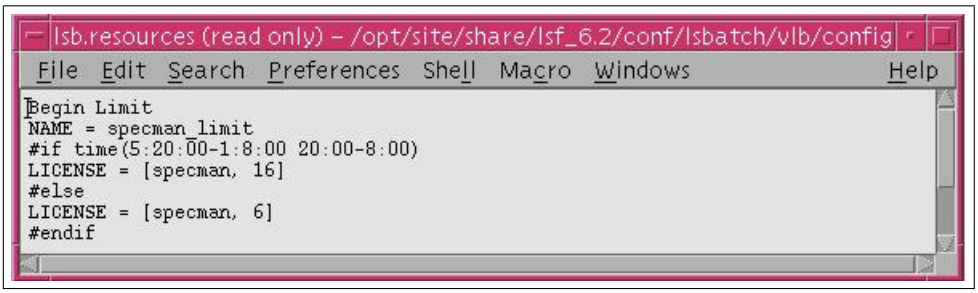

Figura 45: Limitação por usuário dos *license-tokens* specman.

A figura 46 ilustra a situação em que o usuário specman chamado sebai atinge o limite de 6 *license-tokens*, definido para o período diurno. Por isso, observamos alguns de seus *jobs* no estado de *PEND* [15].

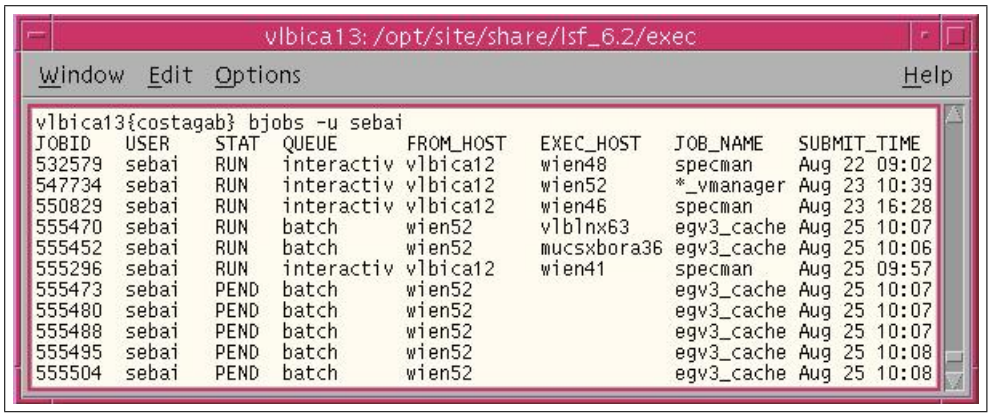

Figura 46: O usuário specman sebai atingindo o limite de 6 *license-tokens*.

6.4.2 Limitação para licenças qhsimvh, msimhdlmix e titanak

Depois de implantar a limitação para licenças specman, as próximas licenças da lista foram: qhsimvh, msimhdlmix e titanak. Então, de forma semelhante ao que fizemos anteriormente, discutimos com os usuários destas licenças para saber quantos *license-tokens* deveriam ser criados. Os números acordados foram: 110 *license-tokens* para licenças qhsimvh e msimhdlmix e 300 *license-tokens* para licenças titanak.

Em seguida, criamos todos esses *license-tokens* para essas três licenças e modificamos o arquivo ExecHandler.xml de forma que, os *jobs* relacionados a estas licenças requeressem os *license-tokens* correspondentes automaticamente no momento da sua submissão [16]. Para essas três licenças, devido ao alto número de *license-tokens* criados, vimos que não

havia necessidade de limitá-los por usuário, assim como fizemos com os *license-tokens* specman.

Uma vez que todas as modificações foram feitas, reiniciamos a plataforma LSF e monitoramos os *jobs* referentes as licenças qhsimvh, msimhdlmix e titanak por alguns dias, para garantir que tudo ia bem.

A partir daí, foi preciso desenvolver um ferramenta com interface gráfica, escrita usando *scripts Perl*, para auxiliar os gerentes de licença no monitoramento dos *license-tokens*, como será descrito no capítulo 7. Por isso, não tivemos tempo suficiente para implantar a limitação a todas as licenças listadas na figura 43.

7 *Monitoramento do uso de license-tokens*

Para facilitar o acompanhamento do uso dos *license-tokens*, a última parte deste trabalho foi dedicada ao desenvolvimento de uma aplicação simples, usando *scripts Perl*, que permite aos usuários e gerentes de licença checar, em tempo real, o uso atual de licenças e de *license-tokens*.

7.1 Adquirindo familiaridade com a linguagem *Perl*

Perl é uma linguagem de *scripts* bastante útil. Usando *Perl*, pode-se facilmente extrair informações de arquivos para criar relatórios usando estas informações, escrever *scripts* para executar tarefas de administração de sistemas, entre outras ações. Veja mais em [1].

Como primeira tarefa, desenvolvemos um *script* que fornece aos usuários o *status* do uso de licenças, o denominamos realtimeusage. Com este *script*, os usuários podem checar o *status* de uma determinada licença fornecida por um determinado vendedor, ou então, checar o *status* de todas as licenças, de um específico vendedor, usadas por um determinado usuário. O *script* realtimeusage usa o comando lmstat para obter todas as informações referentes ao *status* das licenças e em seguida reporta essa informação como ilustram as figuras 47 e 48.

7.2 licstat e licstat_gui

Depois de implantar a solução encontrada para a limitação do uso de licenças, desenvolvemos um *script*, em parceria com Jean-Yves Larguier da Infineon Technologies de Xian, que denominados licstat. Este *script* tem o objetivo de permitir aos usuários e gerentes de licença observar em tempo real o uso dos *license-tokens*. Para usarmos o *script* licstat, basta digitarmos licstat em um terminal UNIX, em seguida, temos como resposta

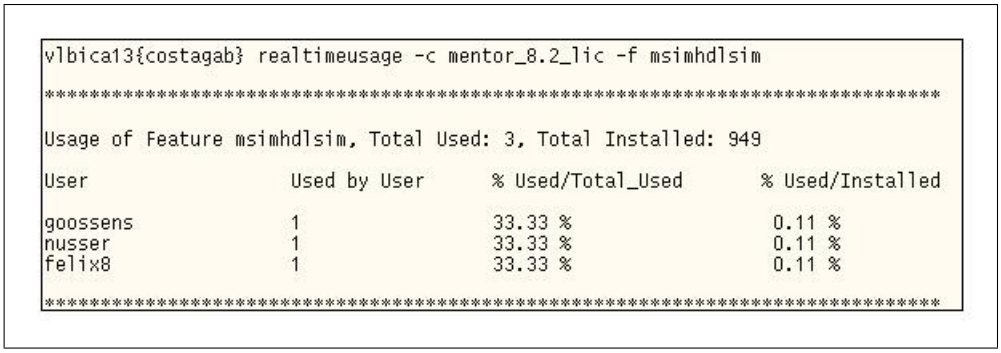

Figura 47: Checando o *status* da licença msimhdlsim usando realtimeusage.

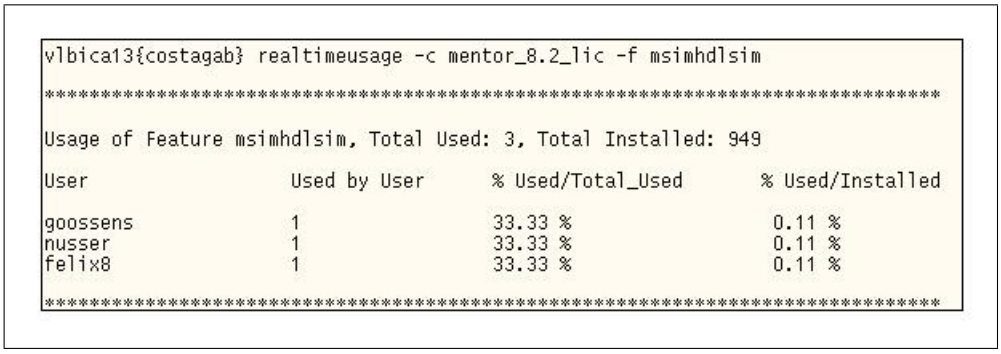

Figura 48: Checando o uso de licenças do usuário paganop usando realtimeusage.

uma tela como a que é ilustrada na figura 49.

Os usuários podem verificar apenas os *license-tokens* usados por si mesmos, usando o comando licstat -me, ao invés de licstat.

Na figura 50, temos um exemplo de uso do comando licstat -me. No caso, o usuário costagab checa seu uso pessoal de *license-tokens*. Como pode ser visto, ele estava naquele momento usando apenas 2 *license-tokens* specman.

A segunda aplicação, também escrita usando *Perl*, é chamada de licstat_gui e, nada mais é do que uma versão gráfica que incorpora realtimeusage e licstat em uma só ferramenta. A ferramenta licstat_gui foi desenvolvida para ser usada pelos gerentes de licença. A figura 51 ilustra a tela inicial de licstat_gui.

Na tela mostrada na figura 51, o usuário pode escolher entre checar o status das licenças, dos license-tokens, obter ajuda em como usar a ferramenta ou, simplesmente, sair da aplicação.

***Status* das licenças:** Clicando sobre o botão *License Status* temos acesso à janela de *status* de licenças ilustrada na figura 52. Em seguida, escolhemos o nome do *license file*

```
vlbica13{costagab} licstat
*************************************************************************************************
TOKEN                                LIMIT       AVAILABLE       USED      USERS
specman                                16              12          4      sebai=2 costagab=2
qhsimvh                               110             107          3      collura=1 sebai=2
msimhdlmix                            110             107          3      collura=1 sebai=2
msimhdlsim                            500             500          0
fastscan                              500             500          0
testkomp                              500             499          1      nieuwenh=1
*************************************************************************************************
```

Figura 49: Usando o *script* licstat para checar o uso dos *license-tokens*.

```
vlbica13{costagab} licstat -me
*************************************************************************************************
TOKEN                                LIMIT       AVAILABLE       USED      USERS
specman                                16              12          4      sebai=2 costagab=2
*************************************************************************************************
```

Figura 50: Usando o *script* licstat para checar o uso pessoal de *license-tokens*.

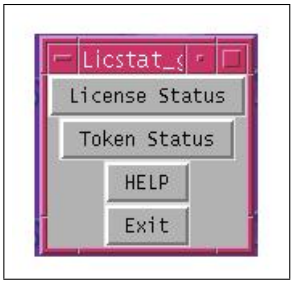

Figura 51: Tela inicial de licstat_gui.

correspondente a licença que queremos observar o *status*, juntamente com o nome desta licença. Como exemplo, usamos o *license file* do vendedor Mentor Graphics e o nome da licença qhsimvh. Os resultados são apresentados na figura 52. Os usuários locais, neste caso, os de Sophia-Antipolis, aparecem em cinza.

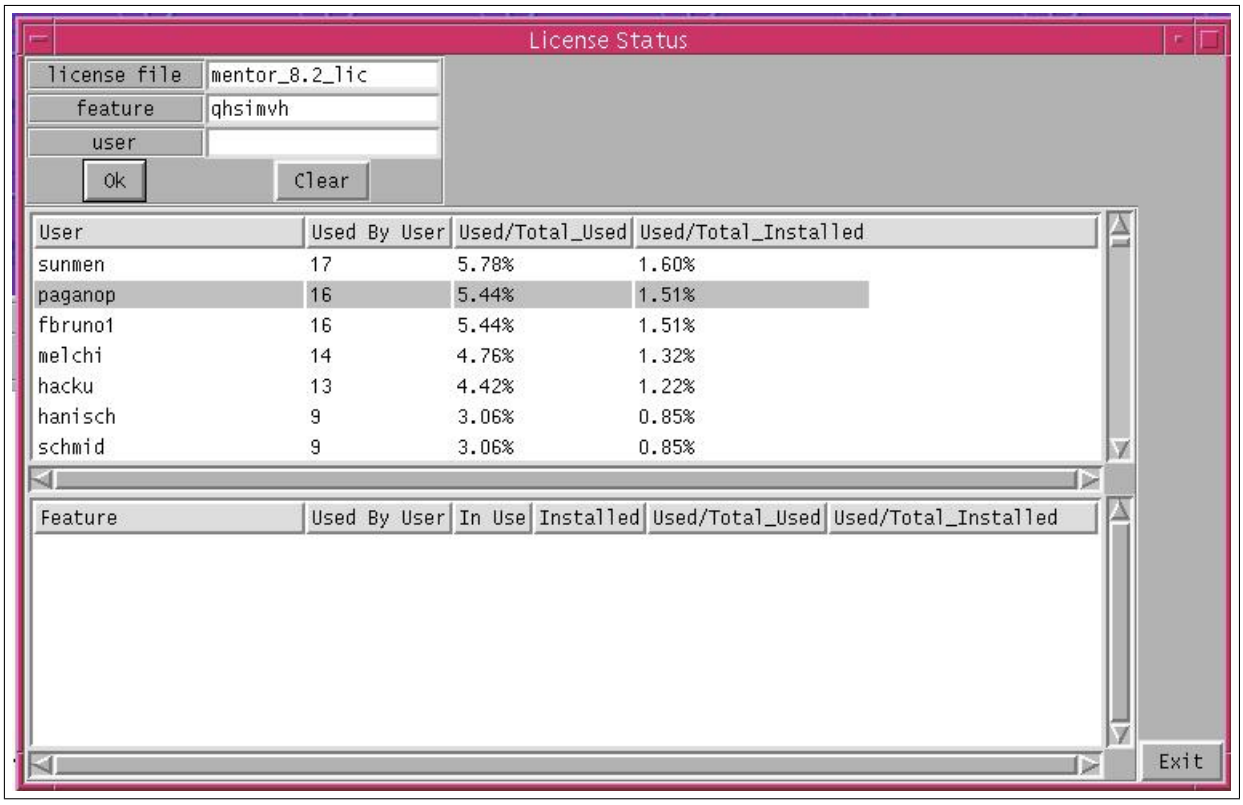

Figura 52: Usando licstat_gui para obter o *status* da licença qhsimvh.

Outra forma de se observar o *status* de licenças, é feita por usuário, de forma que escolhemos o *license file*, assim como visto anteriormente, porém, ao invés de informar o nome de uma licença, informamos o nome de um usuário. Dessa maneira, todas as licenças do vendedor informado, que estão sendo usadas por aquele usuário serão mostradas. Como mostra a figura 53.

***Status* dos *license-tokens*:** Clicando no botão *Token Status* na tela ilustrada na figura 51, podemos checar o *status* dos *license-tokens* para todo o centro de Sophia-Antipolis. São mostrados os nomes dos *license-tokens*, seus limites definidos, o *status* de uso, a disponibilidade, quem os usa e em que projetos eles estão sendo usados. Para atualizar os resultados, apenas clicamos sobre o botão *Update*. A figura 54 ilustra o uso da ferramenta licstat_gui para monitorar os *license-tokens*.

Ajuda: Uma pequena aba de ajuda contendo algumas instruções é também fornecida por licstat_gui. Para acessá-la basta clicar sobre o botão *HELP*. Veja esta aba de ajuda

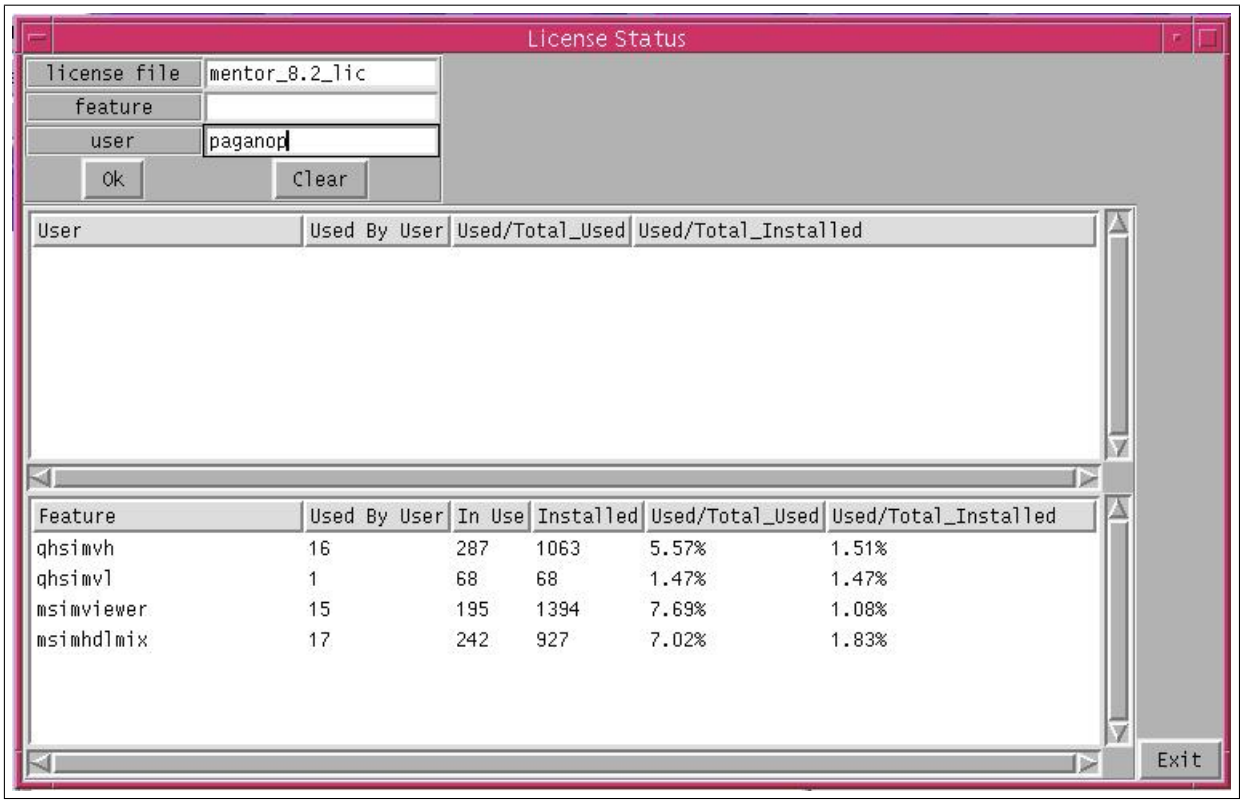

Figura 53: Usando licstat_gui para obter o *status* das licenças por usuário.

na figura 55.

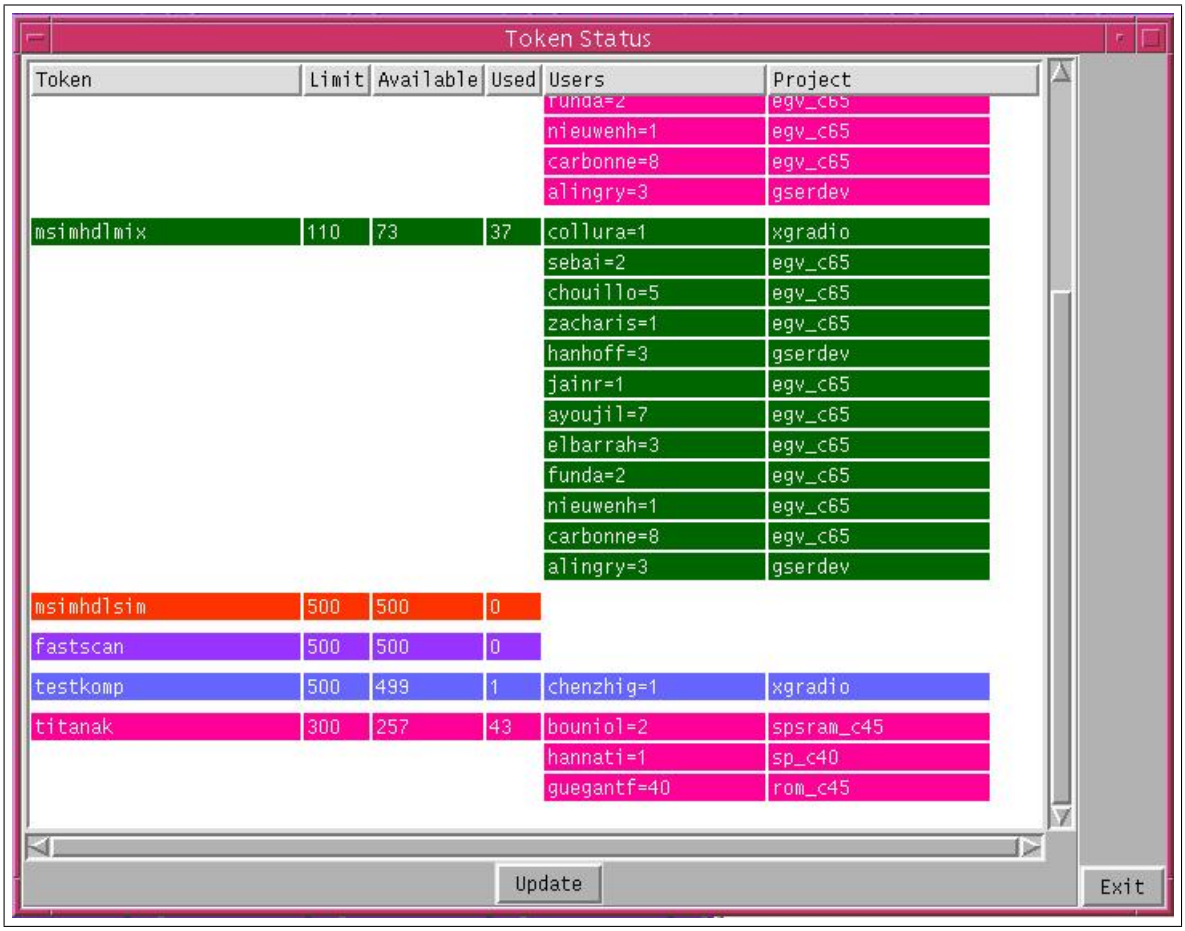

Figura 54: Usando licstat_gui para obter o *status* dos *license-tokens*.

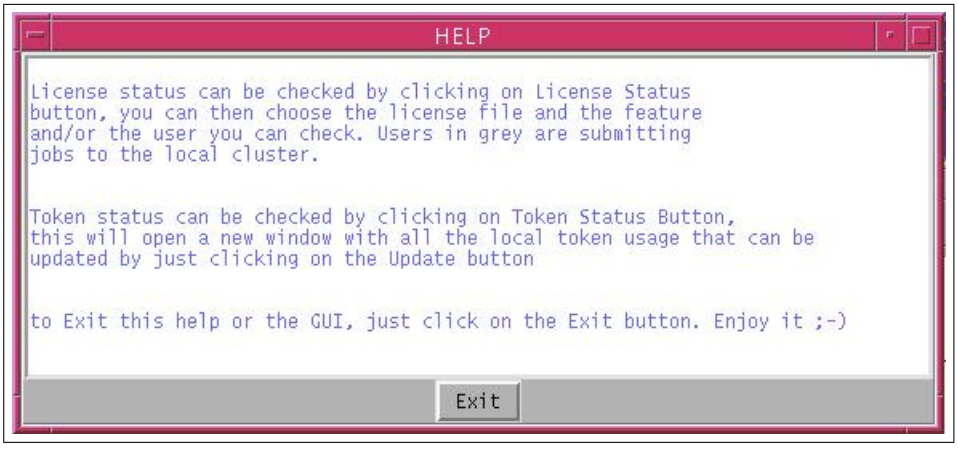

Figura 55: Usando a aba de ajuda da ferramenta licstat_gui.

Perspectivas para o projeto

Gostaria de relembrar que o principal objetivo de nosso trabalho era validar e implementar uma solução para limitar o uso de licenças flutuantes em Sophia-Antipolis. Esta solução seria então expandida para outros centros de P&D da empresa. Como um segundo objetivo, devíamos desenvolver uma ferramenta gráfica que permitisse aos gerentes de licença acompanhar os limites impostos sobre as licenças, em tempo real e de maneira fácil.

Como se pode ver neste relatório, nosso principal objetivo foi parcialmente atingido, já que a solução foi aplicada ao centro de Sophia-Antipolis, no entanto, não tivemos tempo de expandi-la para os outros centros da Infineon.

O segundo objetivo foi também atingido parcialmente, pois a ferramenta gráfica que foi desenvolvida fornece o acompanhamento do *status* tanto das licenças quanto dos *license-tokens*, no entanto, os gerentes de licença não podem controlar os limites impostos através desta ferramenta, pois somente os administradores da plataforma LSF têm o direito de fazer modificações nos arquivos de configuração LSF para criar ou redefinir *license-tokens*.

Ao fim deste estágio, um projeto de grandes dimensões foi iniciado para discutir a limitação de uso de licenças tanto a nível local quanto a nível global. Nossa contribuição para este projeto foi escrever um documentação detalhada orientando, passo a passo, os gerentes de licença em como implantar a solução proposta nos diversos centros da Infineon Technologies.

Conclusão

Este trabalho pode ser analisado segundo dois diferentes aspectos, o pessoal e o profissional.

Profissionalmente, estes seis meses de estágio durante os quais eu estive integrado a Infineon me deram oportunidade de aprender algo novo, em um domínio que eu conhecia muito pouco ou nada. Já que mesmo tendo estudado Eletrônica para Sistemas Embarcados na ESISAR, o estágio foi realizado na área de Informática e Redes. Eu pude aprender bastante sobre a plataforma computacional LSF, aprendi outra linguagem de programação, *Perl* e, em menor intensidade, aprendi sobre o gerenciamento de licenças realizado pelo grupo GLM. Além disso, ainda pude tomar conhecimento do fluxo de concepção de circuitos integrados usado na Infineon.

Pessoalmente, eu pude ter um bom relacionamento com a maioria das pessoas que encontrei na Infineon, o que me ajudou a ser integrado ao grupo facilmente. Os jogos de futebol nas sextas-feiras foram bons momentos de diversão com os colegas de trabalho. Outro ponto interessante, é que o grupo GLM era multicultural, uma vez que de 6 pessoas, existiam 4 nacionalidades diferentes.

Anexos

Limitação do uso de licenças a nível global:

A limitação do uso de licenças a nível global pode ser implementada usando o arquivo *End User Administration Options File* fornecido por FlexLM. Neste arquivo, os gerentes de licença podem restringir o uso de licenças a alguns usuários e também limitar o número de máximo de licenças que podem ser usadas ao mesmo tempo. A figura 56 ilustra como a solução funciona.

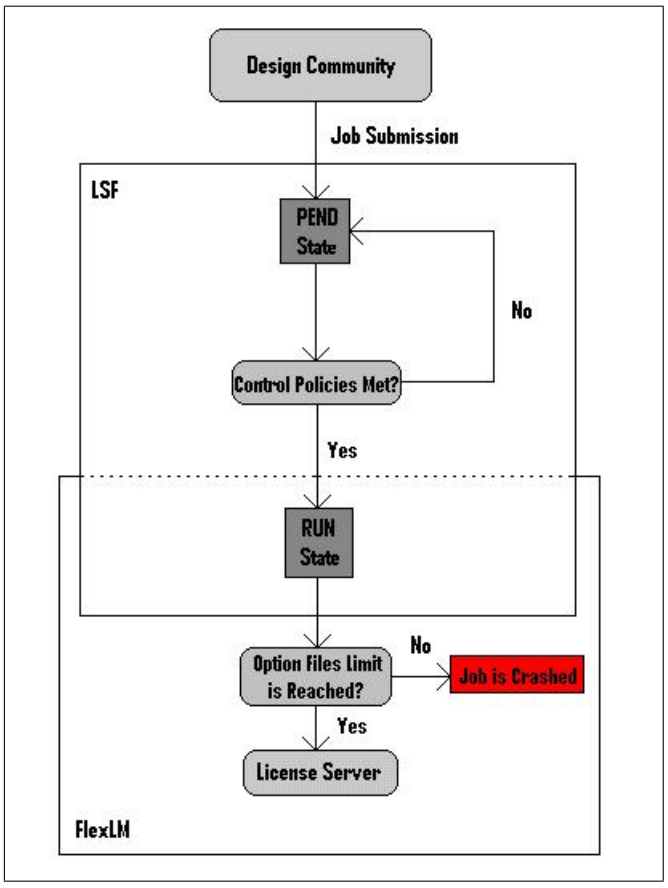

Figura 56: Solução para a limitação do uso de licenças a nível global.

A comunidade de projetistas submete seus *jobs* para a plataforma LSF que, em seguida, os coloca em estado de *PEND*, enquanto o *master host* checa se todas as regras/limites definidos estão sendo satisfeitos, em caso afirmativo, o *job* é despachado

mudando seu estado de *PEND* pra *RUN*. Nesse caso, os limites de uso de licenças são impostos pelo arquivo *End User Administration Options File*, ou seja, no momento de fazer uso da licença, caso o limite tenha sido atingido, o *job* será perdido, caso contrário, ele se inicia normalmente.

Referências

[1] B. Pouliquen, *Introduction au Langage Perl*, 1st ed. Laboratoire d'Informatique Medicale de la Faculté de Medecine de Rennes, 2003.

[2] S. Lidie and N. Walsh, *Mastering Perl/Tk*, 1st ed. O'Reilly, 2002.

[3] *FlexLM End User's Guide*, 9th ed. Macrovision Corporation, 2003.

[4] *Administering Platform LSF*, 6th ed. Platform Computing Corporation, 2006.

[5] *LSF Reference Guide*, 4th ed. Platform Computing Corporation, 2001.

[6] *Platform LSF API Reference*, 6th ed. Platform Computing Corporation, 2004.

[7] *Platform LSF Administrator's Primer*, 6th ed. Platform Computing Corporation, 2004.

[8] *Using Platform LSF License Scheduler*, 6th ed. Platform Computing Corporation, 2005.

[9] *Cost Reduction Proposal*, 2nd ed. Infineon Technologies Bangalore, 2008.

[10] *BRSLSFIW Quick Admin*, 2nd ed. Infineon Technologies Bristol, 2004.

[11] *Job Submission Guide*, 2nd ed. Infineon Technologies Bristol, 2004.

[12] *Inway Release 5.2.2 - User Manual*, 2nd ed. Infineon Technologies Munich, 2006.

[13] *Inway Release 5.2.2 - Release Notes*, 2nd ed. Infineon Technologies Munich, 2006.

[14] *Inway Tutorial Release 5.2.2 - Inway Documentation*, 2nd ed. Infineon Technologies Munich, 2007.

[15] G. Costa and S. Muller, *License Cost Reduction*, 1st ed. Infineon Technologies Sophia Antipolis, 2008.

[16] S. Muller and G. Costa, *Local Optimization of License Usage*, 1st ed. Infineon Technologies Sophia Antipolis, 2008.