

SCE-703 Projeto e Implementação de Sistemas Embarcados I - PISE

Professor responsável: *Fernando Santos Osório*

Semestre: 2008/2

Horário: Segunda 09h20

E-mail: fosorio@icmc.usp.br

fosorio@gmail.com

Web: <http://www.icmc.usp.br/~fosorio/>

TRABALHO PRÁTICO - Projeto Final

Definição de 24/11/2008

[Descrição Geral]

Este trabalho consiste em desenvolver um controlador de um elevador, baseado em FPGA, usando o “soft-core” do processador NIOS-II. Baseado nos trabalhos anteriores de implementação de hardware em FPGA (TP01) e de implementação de um software em “C” para o NIOS-II (TP02), será implementado um trabalho final [Projeto Final – PRJF] que visa simular de modo bastante realista um elevador. A implementação será feita em linguagem “C” usando o MicroC/OS-II RTOS que roda sobre o processador configurável NIOS-II na placa Firefly-Cyclone da Altera (1C12). A implementação consiste de um módulo que irá ler os comandos do usuário do elevador e de um outro módulo que irá controlar o deslocamento do elevador, sendo que ambos os módulos irão rodar concorrentemente em 2 tasks simultâneas executadas sobre o sistema RTOS no NIOS-II.

>> Leia com atenção o detalhamento a seguir do projeto final <<

[Detalhamento do Projeto – Controlador de Elevador]

INSTRUÇÕES PRÉVIAS

Considerar neste projeto os seguintes exemplos de aplicações e comandos do NIOS-II:

PASSO 1:

- Criar um exemplo de programa no ambiente de desenvolvimento do NIOS II IDE usando o template do “Hello MicroC/OS-II” [NiosII IDE – File / New Project / Application => Example: Hello MicroC/OS-II]
- Compilar e testar esta aplicação. Criar a biblioteca (lib) correspondente e o executável (elf) que permita a execução deste programa em hardware, sendo rodado no Firefly-Cyclone FPGA com o processador NIOS II (Hardware) e o sistema operacional MicroC/OS-II RTOS;
- A execução deste projeto deve permitir que sejam apresentadas 2 tarefas em execução simultânea, sendo apresentada na console de execução uma mensagem que demonstra a alternância entre as tarefas.
- Dicas: use a especificação de hardware do kit 1C12 standard (setar o arquivo .ptf no wizard do projeto e o arquivo .sof configurado no QuartusII Programmer) => path c:/altera/kits/nios2_51/examples/verilog/niosII_cyclone_1c12_eval/standard/)

PASSO 2:

- Executar o programa compilado no NIOS II IDE fora da IDE, ou seja, carregar e executar diretamente em hardware o programa executável .elf, sem o uso da interface IDE. Usar apenas o NIOS II Console Shell:
 - > Abrir uma primeira janela com o NIOS II Command Shell e abrir o terminal da console do NIOS2
Executar o comando: `nios2-terminal`
 - > Abrir uma segunda janela com o NIOS II Command Shell, carregar e executar o arquivo ELF no NIOS2
Executar os comandos: `cd <diretório onde se encontra o arquivo executável .elf>`
`nios2-download -accept-bad-sysid -g <arquivo>.elf`
 - > Verificar se a execução do programa é igual a obtida quando o mesmo era executado na IDE do NIOS.

PASSO 3:

- Baixar da Internet um programa exemplo fornecido pelo professor, baseado no exemplo do Hello MicroC/OS-II, que possui 2 tasks, uma de leitura de teclado e outra de controle dos leds da placa de hardware. Compilar e executar diretamente na console o exemplo fornecido, como foi feito com o exemplo original do MicroC/OS-II.

PASSO 4:

- Agora você já sabe como fazer um programa que possui um módulo de interface com o usuário (teclado, que simula os botões de chamada/comando do elevador) e um módulo de controle do elevador (leds, que indicam o andar atual do elevador). Então mãos a obra, faça o seu controlador do elevador!!
“Just do it!” ;^)

CONTROLADOR DE UM ELEVADOR

Considerar neste projeto os seguintes elementos:

- Elevador projetado para um prédio de 8 andares. Cada um dos leds da placa Firefly-NIOS servirá para indicar o andar atual onde se encontra o elevador. Simule o movimento do elevador, indicando através dos leds o andar em que ele se encontra no momento;
- Definir um conjunto de comandos de chamada do elevador através do teclado: o usuário poderá chamar o elevador para os andares de 1 a 8, sendo que as chamadas deverão ser colocadas em uma fila de chamadas pendentes à medida que forem sendo recebidas (não precisa ser exatamente uma “fila”, pois podem ser tratadas prioridades em relação às solicitações). As chamadas são lidas pela *task* de leitura do teclado, e são então colocadas em uma fila de chamadas pendentes. Use as teclas de 1 a 8 para indicar qual andar está chamando o elevador;
- Definir um conjunto de comandos de envio do elevador através do teclado: o usuário que está dentro do elevador deve poder solicitar um andar de destino, sendo que esta solicitação deve ser colocada em uma fila de destinos pendentes à medida que forem recebidas (não precisa ser exatamente uma “fila”, pois podem ser tratadas prioridades em relação às solicitações). As solicitação devem ser lidas pela mesma *task* de leitura do teclado (usada para chamar o elevador), onde mudam apenas as teclas usadas para indicar as solicitações. Sugestão: use as seguintes teclas ‘q’ (andar 1), ‘w’ (andar 2), ‘e’ (andar 3), ‘r’ (andar 4), ‘t’ (andar 5), ‘y’ (andar 6), ‘u’ (andar 7), ‘i’ (andar 8), pois ficam próximas aos números que representam os andares solicitados;
- Concorrentemente a tarefa de leitura de comandos (chamada/destino) deve ser executada uma tarefa de controle do elevador, que considera estas solicitações e executa uma ação correspondente de controle do elevador, movendo este de andar em andar (e exibindo através dos leds esta movimentação) de modo a atender as solicitações;

- Sua “missão” é fazer com que o elevador atenda aos usuários, subindo, descendo, ou ficando parado aguardando uma nova solicitação;
- Sugestão: coloque um “*watchdog*”, um processo (*task*) que identifique se a fila de chamadas e destinos ficou muito grande (fora de controle) e assim realize um reset do elevador. Note que o “*reset*” não deve mudar o elevador de andar, apenas restaurar seu funcionamento, resetando a fila de comandos pendentes. O “*watchdog*” também pode ser usado para verificar se o programa sofreu alguma interrupção anormal e não está funcionando corretamente (por exemplo, na situação em que o elevador se encontra parado e existe uma ou mais chamadas pendentes).
- Você também pode usar a console para enviar mensagens indicando o estado do elevador (usado para debug) e até mesmo para indicar o disparo de um alarme em caso de pane (“*watchdog*” entrou em ação);

Fica a cargo do grupo a definição da estratégia de sequenciamento de ações do elevador. Exemplo:

- 1) O elevador sempre sobe até o andar mais alto que foi chamado (se tiver mais de uma chamada requisitada), e depois cumpre a seqüência de destinos indicados pelos usuários que estão dentro do elevador, ignorando novas chamadas;
- 2) O elevador atende apenas a primeira chamada identificada, aceitando apenas um destino quando estiver ocupado e for enviado para algum andar. Somente após atendida a chamada é que ele irá tratar as chamadas seguintes;
- 3) O elevador busca otimizar suas viagens, sempre tentando parar e pegar usuários quando estiver passando por um andar que possui uma chamada pendente;
- 4) O elevador considera uma prioridade de deslocamentos dos usuários, ou seja, quem está no térreo quer subir e quem está em andares acima quer descer. Usa uma política especializada para este tipo de deslocamento;
- 5) ...

[Alternativa de Projeto Final: Video Game]

Implementação de um GAME. Implementação de um jogo também consistindo de *tasks* que rodam simultaneamente sobre o processador NIOS-II com o MicroC/OS-II RTOS. Uma das tarefas é responsável pela leitura dos comandos dados pelo usuário através da console (teclado) e a outra tarefa é responsável por controlar o jogo e dar o feedback visual para o usuário (console e/ou leds) sobre o andamento do jogo. Os alunos que optarem por esta segunda alternativa de trabalho devem apresentar ao professor uma definição/descrição de seu jogo.

ENTREGA DO TRABALHO:

- * Envie um e-mail os arquivos com o projeto completo do NIOS-II ao prof. Osório (incluir .c, .h, .elf, ... todos arquivos que compõem seu projeto)
E-MAIL TO: **work2usp@yahoo.com** (Enviar o original para este email)
EMAIL CC: **fosorio@gmail.com** (Enviar com cópia para este email)
SUBJECT: **[SCE703] PRJF <nome_aluno>** (Assunto do email)
- * Escreva no corpo da mensagem de e-mail:
NOME: <seu nome> + <nome dos componentes do grupo> (Max. 3 Alunos)
INFORMAÇÕES SOBRE O PROJETO: <projeto desenvolvido>,
<informações que julgar necessárias para a avaliação e teste do seu projeto>

* Exemplo:

Nome: Fernando Osório + Fulano da Silva

Informações: NIOS II Verão X.Y

Controle do Elevador (Baseado no Exemplo: Hello MicroC/OS-II)

Documentação do projeto e instruções de uso - arquivo: Infos.pdf

Se você for enviar qualquer tipo de executável (.exe, .com, .bat, arquivo de scripts), o arquivo terá obrigatoriamente que estar compactado em formato **.rar** ou **.bz2** (OUTROS FORMATOS NÃO SERÃO ACEITOS, POIS SÃO RECUSADOS PELO SERVIDOR DE E-MAIL).

* Entregar até a data indicada no Site da Disciplina

<http://www.icmc.usp.br/~fosorio/SCE703/SCE703.html> (ver em Avaliações / Trabalhos Práticos)

===== THAT'S ALL FOLKS !!! =====