ESPECIFICAÇÃO DETALHADA DOS TRABALHOS – TP

Os trabalhos especificados neste documento devem ser implementados sobre a plataforma XS40/XST-1 disponíveis no Laboratório de Arquitetura de Computadores e Sistemas Digitais. Todos os trabalhos deve ser simulados em VHDL e implementados sobre a plataforma em hardware mencionada. Quando o trabalho desenvolver um módulo a ser usado em conjunção com outros (caso típico dos drivers de dispositivo) existem duas opções para realizá-lo. A primeira é que os alunos do grupo em questão desenvolvam uma aplicação simples que valide sua implementação. A segunda é a implementação ser documentada de forma integrada com uma aplicação desenvolvida no contexto de outro(s) grupo(s) de alunos. Neste último caso, a interface entre os dois trabalhos deve ser bem definida e documentada, mas uma única documentação pode ser entregue para ambbos os grupos.

A documentação a ser entregue pelo(s) grupo(s) deve conter, no mínimo:

- Um arquivo texto que comente o trabalho, as dificuldade encontradas e o estágio de avanço alcançado na implementação.
- Código fonte VHDL comentado;
- Testbench usado para testar o projeto e formas de onda de simulação associadas;
- Projeto de hardware desenvolvido no ambiente Foundation em formato comprimido (use opção "Archive" do menu File do ambiente); Preferencialmente, este projeto deve conter o código VHDL simulado na ferramenta Active-HDL, com os correspondentes testbenches e formas de onda.

1 DRIVER DE MOUSE

❖ Alunos envolvidos - Guilherme Tesser e Juliano Vacaro

O objetivo deste trabalho é implementar um hardware de acionamento de um dispositivo de entrada do tipo mouse. A interface do driver deve ter o formato mostrado na Figura 1. Todas as linhas da Figura são de 1 bit, exceto quando explicitamente escrito sobre elas um valor indicando um sinal de múltiplos bits. As linhas **KB_CLK** e **KB_DATA** são os sinais de interface com o mouse, ambos bidirecionais, cuja funcionalidade é descrita pelo protocolo de interface de mouse PS/2, explicado, por exemplo, nas seguintes páginas da Internet: http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/mouse/mouse.html.

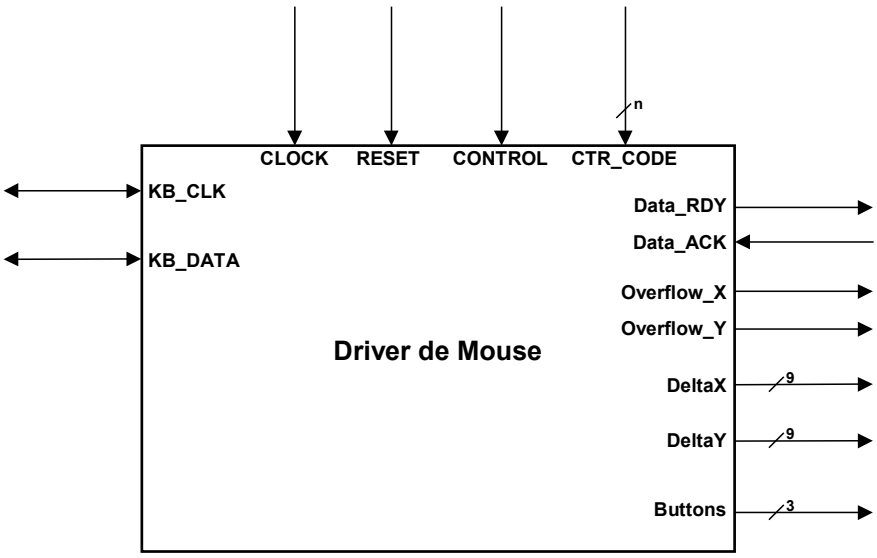

Figura 1 - Interface de entrada e saída do driver de mouse.

A interface mostrada na Figura 1 pressupõe um protocolo de comunicação assíncrona entre o driver e o usuário deste, usando os sinais colocados à direita na Figura. O usuário deve gerar o sinal de **RESET** para inicializar o mouse no modo "stream" com relato de dados habilitado. Todos os sinais de controle devem ser ativos em "1".

As linhas **CONTROL** e **CTR_CODE** servem para enviar sinais de controle ao mouse, de acordo com o protocolo PS/2. Os alunos devem definir a largura destas linhas a partir do número de sinais de controle que o driver vai permitir enviar. Os códigos que vão ser enviados ao mouse não precisam ser exatamente os mesmos do protocolo, pode-se empregar uma codificação mais simples e o driver converter esta codificação para o formato esperado pelo mouse. Em particular, se um controle pressupõe o envio de vários bytes, tal como "Set Sample Rate", vários códigos diferentes podem ser usados, um para cada taxa de amostragem desejada. Neste caso, o driver decodifica este valor e gera os bytes a enviar ao mouse.

Quando um controle recebe dados do mouse e for necessário informar algo ao usuário do driver, isto é feito usando as mesmas linhas de relato de dados. Neste caso, cabe ao usuário decodificar esta instrução corretamente.

2 APLICAÇÃO MOUSE

❖ Alunos envolvidos – Mauro dos Santos e Gustavo Welp

Usando o driver descrito na Seção 1, este grupo deve desenvolver uma aplicação para demonstrar o funcionamento básico do mouse. A estrutura da aplicação deve pressupor o mouse movendo-se sobre um segmento de plano XY com coordenadas definidas por dois intervalos discretos idênticos de inteiros, na faixa [-256, 255]. A inicialização da aplicação deve colocar o cursor na posição central do segmento de plano, ou seja, na posição (0,0). A partir daí, a aplicação deve inicializar o driver de mouse e começar a receber deste deslocamentos relativos, estados dos botões e informação de transbordo nas coordenadas. A aplicação deve acumular os deslocamentos e relatar nos mostradores de 7 segmentos e nos diodos emissores de luz da placa XST-1 o valor das coordenadas de localização atual do mouse (X nos mostradores e Y nos diodos). O sinal das coordenadas deve ser mostrado em dois dos segmentos do mostrador da placa XS40. O mesmo deve ser feito com o estado dos botões e as informações de transbordo em X e Y. Opcionalmente, a informação de sinal pode ser colocada no ponto decimal dos mostradores da placa XST-1. A cada detecção de transbordo em pelo menos um dos eixos a aplicação deve parar e aguardar um sinal de reinício por parte do usuário externo. Além disso, a aplicação deve implementar alguns dos comandos de controle do mouse, para testar o funcionamento do driver neste aspecto.

3 APLICAÇÃO MOUSE - VÍDEO

❖ Alunos envolvidos – Aline Mello e Leandro Möller

Usando o driver de mouse descrito na Seção 1, e o driver de vídeo descrito no manual da plataforma XS40/XST-1 versão 1.3.2, este grupo deve desenvolver uma aplicação para demonstrar o funcionamento do mouse movimentando-se na tela.

A tela deve possuir uma imagem de fundo fixa, ocupando os cerca de 30 Kbytes descritos na documentação do driver de vídeo. Um cursor retangular pequeno deve ser criado no espaço restante na memória de 32Kbytes, e um processo de mapeamento dos pixels do cursor para uma posição da tela deve ser elaborado, que funcione em tempo real. As coordenadas fornecidas pelo driver de mouse não possuindo a mesma resolução que as coordenadas de tela (256 pontos x 256 pontos no mouse contra 480 linhas x 256 colunas no vídeo), um mapeamento das primeiras para as últimas deve ser realizado. Note-se que este mapeamento possui escalas distintas para cada um dos eixos. O movimento do mouse deve causar um movimento correspondente do cursor na tela. Não é necessário, mas é bem vindo fazer o tratamento de botões, transbordos nos eixos, etc.

APLICAÇÃO TECLADO - VÍDEO

Esta Parte envolve três grupos de trabalho. Aqui descreve-se a estrutura geral da implementação. A seguir, as Seções 4, 5 e 6 mostram a especificação do trabalho que cada grupo deve realizar.

A idéia é implementar um hardware que leia teclas e imprima-as na tela. O formato é o seguinte:

- Os caracteres mostrados na tela possuem formato fixo, cabendo aos alunos do Grupo 5 definir o mapa de pixels exato para cada caracter. Eles devem ser em número não superior a 85 e devem incluir as 26 letras maiúsculas, as 26 letras minúsculas, os 10 dígitos decimais. Os caracteres restantes devem ser escolhidos pelo grupo, mas alguns caracteres não devem faltar no grupo, tais como o espaço em branco, o ponto e a vírgula. Cada caracter será representado por um mapa de pixels de dimensão 7x5. Mais detalhes na especificação do Grupo 6.
- A tela deve mostrar na sua região visível uma área de texto de 32x32=1024 caracteres. Dada a dimensão do mapa de pixels, uma parte da tela não conterá texto. A área de texto deve ser um retângulo com o canto superior esquerdo correspondendo ao primeiro pixel visível na tela.
- O fundo da tela poderá corresponder a uma imagem armazenada nos primeiros 30Kbytes da memória RAM da plataforma XS40/XST-1 (correspondendo aos endereços 0000H a 77FFH) ou ser um fundo com todos os pixels de uma determinada cor. Neste caso, a cor do fundo deve ser parametrizável, e não é necessário ler da memória nenhuma informação além dos caracteres e sua manipulação. Os 1024 caracteres devem ser armazenados nos últimos 1024 bytes da RAM (correspondendo aos endereços 7C00H a 7FFFH, denominados coletivamente de **Buffer de Teclado**). Os endereços restantes (7800H a 7BFFH) serão usados pelo Grupo 5, conforme descrito mais abaixo, implementando duas tabelas de manipulação de caracteres denominadas **Tabela de Localização do Mapa de Pixels do Caracter** e **Tabela de Mapas de Pixels dos Caracteres**.

A Figura 2 abaixo mostra a proposta de estrutura para o mapa de memória para a Aplicação Teclado-Vídeo.

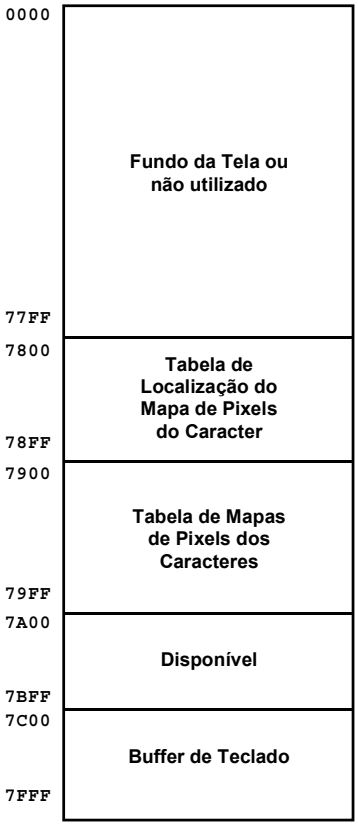

Figura 2 - Mapa de memória aconselhado para a Aplicação Teclado-Vídeo.

4 DRIVER DE TECLADO – Parte 1

❖ Alunos envolvidos – Rafael Santa Maria e Luciano Castro

A partir do driver de teclado descrito no manual da plataforma XS40/XST-1 versão 1.3.2, este grupo deve desenvolver um novo driver que leia os dados de cada tecla apertada e armazene um código de 8 bits desta em um **Buffer de Teclado** ocupando 1024 posições de memória em seqüência.

As teclas que possuírem mais de um byte de código "make" devem ser todas ignoradas. Aconselha-se a usar o código "make" da tecla como o código do caracter sempre que possível. Existe uma situação onde isto não pode ser feito, que é o tratamento de maiúsculas/minúsculas, pois uma mesma tecla, ou seja, um mesmo código "make" pode gerar dois caracteres diferentes, tais como "A" e "a". O tratamento de maiúsculas deve considerar que o usuário está apertando simultaneamente a tecla de "shift" quando aperta a letra correspondente. Ou seja, o driver deve entrar no modo "caixa baixa" entre receber o código "make" da tecla "shift" e antes de receber o código "brake" desta. Neste caso, toda tecla com dupla interpretação deve gerar como código o correspondente ao modo "caixa baixa". Senão, o driver deve estar no modo "caixa alta", o modo por omissão.

O Buffer de Teclado deve ocupar as últimas 1024 posições da memória externa ao FPGA, e deve inicialmente conter em todas as posições o código do caracter espaço em branco. O Buffer de Teclado deve ser implementado como uma fila circular, ou seja, ao chegar à última posição do Buffer, a próxima posição é a primeira do Buffer de Teclado. Cada tecla válida é acrescentada imediatamente na próxima posição do Buffer.

5 DRIVER DE TECLADO – Parte 2

❖ Alunos envolvidos – Gilk Rodrigues e Rafael Chanin

A idéia desta parte é mapear os códigos de caracter contidos no Buffer de Teclado para uma tabela contendo o mapa de pixels deste caracter.

A implementação desta parte do driver de teclado na realidade não usará diretamente o teclado, mas manipulará os códigos de tecla contidos no Buffer de Teclado. Este driver deve operar sobre duas tabelas. A primeira é denominada de **Tabela de Localização do Mapa de Pixels do Caracter**, e deve residir na memória RAM externa ao FPGA. Esta tabela possui 256 posições (ocupando, digamos, os endereços 7800H a 78FFH). Os códigos de tecla de 8 bits armazenados no Buffer de Teclado são usados como um deslocamento a partir da posição inicial da tabela. O conteúdo de cada posição da tabela é um deslocamento para acesso à segunda tabela, denominada **Tabela de Mapas de Pixels dos Caracteres**. esta segunda tabela, também de 256 posições (ocupando, digamos, os endereços 7900H a 79FFH) armazena em cada 3 bytes consecutivos um mapa de pixels de um caracter a ser mostrado na tela (por isso a limitação de usar até 85 caracteres, pois 85*3=255 bytes). Dado um deslocamento obtido da Tabela de Localização do Mapa de Pixels do Caracter, indexado pelo código do caracter, obtém-se o endereço inicial da trinca de bytes que define o mapa de pixels do caracter em questão. Para ver uma explicação do porque usar 24 bits para representar uma matriz de 7x5 bits, ver especificação do trabalho do Grupo 6.

As duas Tabelas aqui citadas são constantes e carregadas no momento de download da aplicação para hardware. Os alunos devem estabelecer o conteúdo destas tabelas em combinação com o grupo do trabalho anterior.

Dado que este trabalho estará tentando fazer acesso à memória de forma concorrente com o hardware do Grupo 4, um esquema de arbitragem deve ser elaborado enter os dois. Aconselha-se que o Grupo 5 proponha o esquema de arbitragem.

Parte do trabalho do Grupo 5 é elaborar o mapa de pixels dos caracteres. Esta estrutura deve estar na documentação final. Um exemplo de mapa de pixels para um caracter aparece na Figura 3. Um documento que contém uma definição de mapa de pixels de caracteres, e que pode ser usado como exemplo para a elaboração desta parte do trabalho encontra-se em http://www.inf.pucrs.br/~calazans/undergrad/topicosI/ipd2131.pdf. Aquele documento é um dispositivo eletrônico que implementa um mostrador alfa-numérico usando uma matriz de diodos emissores de luz 7x5.

6 APLICAÇÃO TECLADO NA TELA

❖ Alunos envolvidos – Emanuel Grohs e Luís Cargnini

Usando as definições do dois trabalhos anteriores, a idéia desta aplicação é mostrar os caracteres contidos no Buffer de Teclado na tela.

Deve-se definir uma área de texto com dimensões 32 linhas x 32 caracteres, perfazendo um total de 1024 caracter, ou seja, o tamanho exato do Buffer de Teclado.

A partir daí, o driver de vídeo descrito no manual da plataforma XS40/XST-1 versão 1.3.2 deve ser alterado, visando desenvolver uma aplicação para demonstrar o funcionamento de teclado conjugado com a tela. O driver de vídeo deve ser alterado para, sobre um fundo pré-definido, computar as posições de cada mapa de pixels de caracter e jogar na tela os pixels de um caracter no momento certo. Cada caracter será representado por uma matriz retangular de pixels 7x5, mas a coluna mais à direita e a linha inferior da matriz são implicitamente composta de pixels apagados. Sobre então uma matriz de pixels 6x4, donde esta ser representada usando 24 bits, ocupando 3 bytes em seqüência na memória, armazenados na Tabela de Mapa de Pixels de Caracter. A organização aconselhada desta é que cada byte seja composto por duas linhas de pixels. Cada pixel é representado por um bit em 0 (representando cor de fundo) ou em 1 (representando a cor do caracter). Deve existir um variável global que armazena a cor do texto, que deve ser sobreposta à cor de fundo quando o pixel do caracter em questão for 1. Ver Figura 3 Abaixo para uma ilustração do formato do mapa de pixels de um caracter.

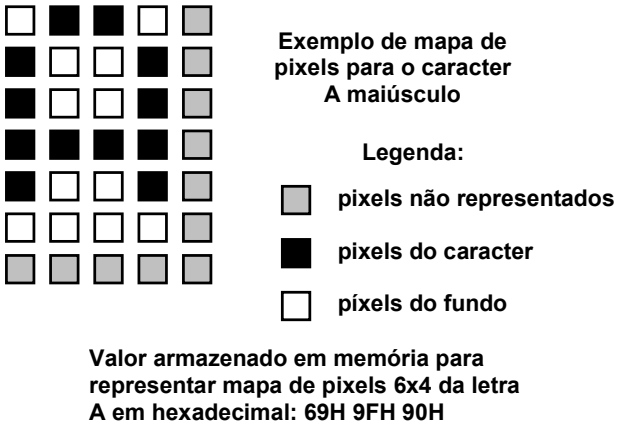

Figura 3 - Formato de um mapa de pixels de caracter. Note-se que os pixels em cinza não são explicitamente representados na Tabela de Mapas de Pixels dos Caracteres, devendo ser inferidos pela aplicação e inseridos usando a cor do fundo. A penúltima linha do caracter sempre estará em branco se se tratar de uma letra maiúscula, para permitir a representação coerente de letras minúsculas que possuem pixels de caracter abaixo desta linha, tais como as letras "p" e "q".

7 APLICAÇÃO TECLADO – 1

❖ Alunos envolvidos – Letícia da Rosa e Micheline Parizotto

A idéia deste trabalho é modificar o driver de teclado para realizar a tarefa de converter *scan codes* de letras para código ASCII correspondente, mostrando o código ASCII de letras no mostrador de sete segmentos e deixando todos os outros caracteres imutáveis, mostrando apenas seu *scan code*.

O circuito acionador de teclado que vem no manual da plataforma XS40/XST-1fica permanentemente mostrando o último byte recebido do teclado, que normalmente corresponde ao último byte do código "break" de uma dada tecla que terminou de ser apertada e liberada. No caso de se apertar uma tecla e se ficar apertando ela, o caracter que aparece no mostrador é o último byte do código "make" da tecla.

O trabalho aqui proposto deve ignorar todos os códigos de "break" e ficar mostrando o código ASCII de cada tecla de letra apertada, ou mostrando o último byte do scan code se a tecla apertada não for uma letra. O tratamento de maiúsculas/minúsculas deve considerar que o usuário está apertando simultaneamente a tecla de "shift" quando aperta a letra correspondente. Ou seja, o driver deve entrar no modo "caixa baixa" entre receber o

código “make” da tecla “shift” e antes de receber o código “brake” desta. Neste caso, toda tecla de letra deve gerar como código ASCII correspondente ao modo “caixa baixa”, ou seja, uma letra maiúscula. Senão, o driver deve estar no modo “caixa alta”, o modo por omissão, onde as letras são minúsculas.