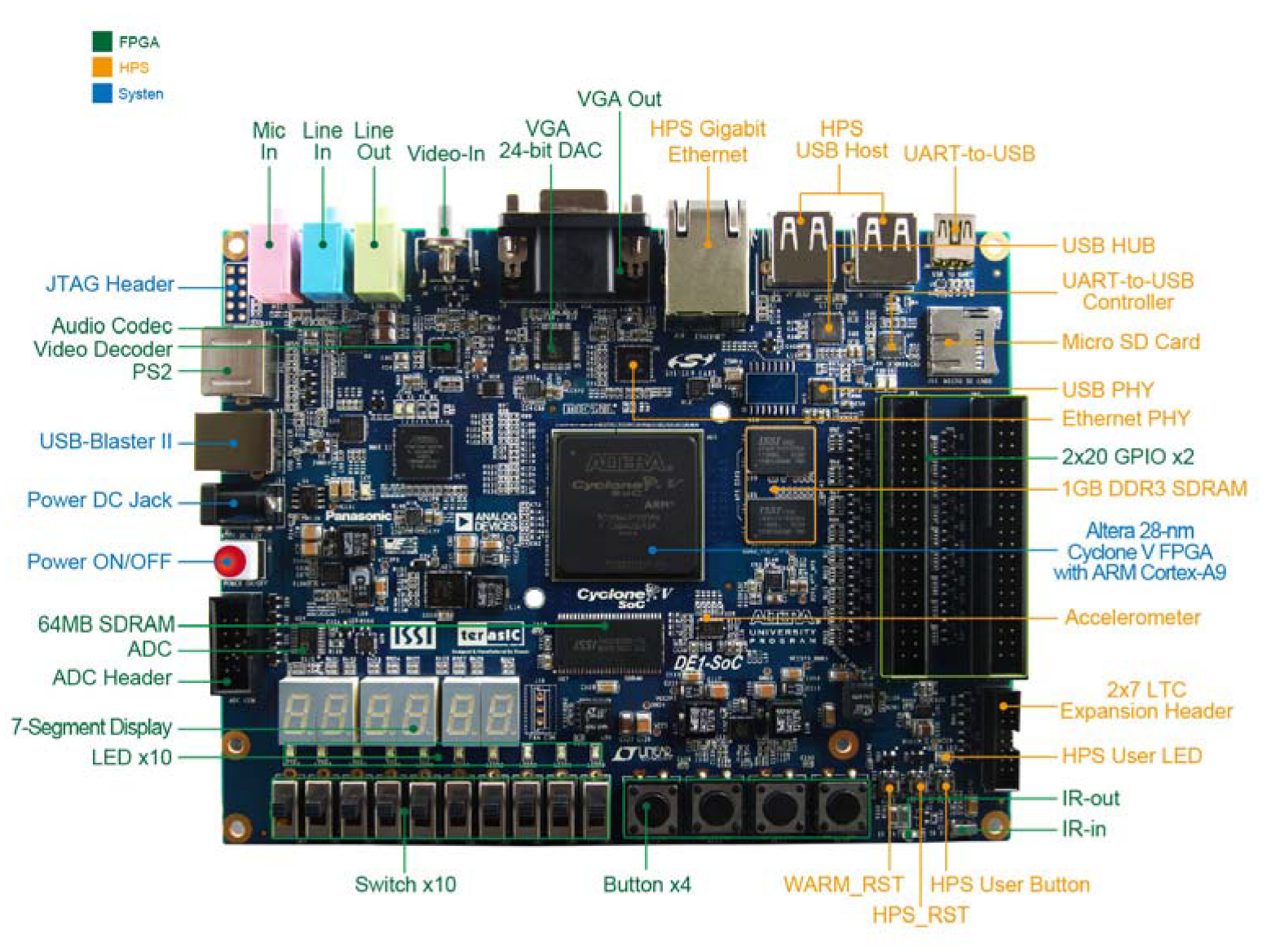


MAPA DE UTILIZAÇÃO DA DE1-SoC

Processador RISC-V versão 2.0



Serial para o PC

9 8 7 6 5 4 3 2 1 0

Timer

Enable

fdiv

Auto/Man

Reset

Clock Manual

Debug

Slow/Fast

Frame Select

PC/IR

BR/FPBR

Display : SW[9:8]==2'b11 ? PC :

SW[9:8]==2'b10 ? IR :

SW[9:8]==2'b01 ? Debug :

SW[9:8]==2'b00 ? Livre

Timer: SW[5] ? 10 segundos : Free running (Para verificar o desempenho de mudanças no processador)

Frame Select : SW[6] troca a frame que é apresentada no monitor

Reset: Assíncrono

Frequência do processador: CLK = 50 MHz / (SW[4:0]) Obs.: SW[4:0]=5’d00000 divide por 32

Modo Debug: SW[9]=1, visualização do PC e do IR no display de 7 segmentos e do Banco de Registradores no display VGA.

Banco de Registradores principal SW[7]=0 ou Banco de Registradores de Ponto Flutuante SW[7]=1

**Monitoramento dos Sinais de Controle:**

|  |  |
| --- | --- |
| **UNICICLO** | |
| **LED** | **Sinal** |
| LEDR[0] | Clock |
| LEDR[1] | Slow/Fast |
| LEDR[2] | Manual/Automático |
| LEDR[3] | Não usado |
| LEDR[4:9] | Não usado |

|  |  |
| --- | --- |
| **PIPELINE** | |
| **LED** | **Sinal** |
| LEDR[0] | Clock |
| LEDR[1] | Slow/Fast |
| LEDR[2] | Manual/Automático |
| LEDR[3] | Não usado |
| LEDR[4:9] | Não usado |

|  |  |
| --- | --- |
| **MULTICICLO** | |
| **LED** | **Sinal** |
| LEDR[0] | Clock |
| LEDR[1] | Slow/Fast |
| LEDR[2] | Manual/Automático |
| LEDR[3] | Não usado |
| LEDR[4:9] | Estado |

**Simulação por forma de onda: Waveform1.vwf** ou **Waveform2.vwf**

**Break Point para Debug Manual:**

O processador irá parar (Clock em Manual) ao iniciar a execução da instrução localizada no endereço armazenado no segmento BRK0 do *In-System Memory Contend Editor*. Isto é, coloque no BRK0 o endereço onde você quer que o processador pare.

No Arquivo Parametros.v defina o tipo do processador a ser sintetizado

**`define UNICICLO**

**`define MULTICICLO**

**`define PIPELINE**

e a ISA a ser implementada

**`define RV32I** sintetiza a ISA básica de inteiros

**`define RV32IM** acrescenta as operações de multiplicação e divisão

**`define RV32IMF** acrescenta as operações de ponto flutuante

**Mapeamento da Memória:**

E/S Mapeada em Memória (MMIO)

|  |  |  |
| --- | --- | --- |
| **Endereço** | **Tamanho** | **Uso** |
| 0xFF00 0000 | 76800 bytes | Frame0  Memória de  vídeo da VGA |
| ... |
| 0xFF01 2C00 |
| 0xFF10 0000 | 76800 bytes | Frame1  Memória de  vídeo da VGA |
| ... |
| 0xFF11 2C00 |
| ... |  | Não Alocado |
| 0xFF20 0000 | 4 bytes | KDMMIO Key Ctrl |
| 0xFF20 0004 | 4 bytes | KDMMIO Key Buffer |
| 0xFF20 0008 | 4 bytes | KDMMIO Disp Ctrl |
| 0xFF20 000C | 4 bytes | KDMMIO Disp Buffer |
| ... |  | Não Alocado |
| 0xFF20 0100 | 4 bytes | Buffer0 Teclado |
| 0xFF20 0104 | 4 bytes | Buffer1Teclado |
| ... |  | Não Alocado |
| 0xFF20 0120 | 1 byte | Rx RS232 |
| 0xFF20 0121 | 1 byte | Tx RS232 |
| 0xFF20 0122 | 1 byte | Rx/Tx Crtl |
| ... |  | Não Alocado |
| 0xFF20 0160 | 4 bytes | Áudio inL |
| 0xFF20 0164 | 4 bytes | Áudio inR |
| 0xFF20 0168 | 4 bytes | Áudio outL |
| 0xFF20 016C | 4 bytes | Áudio outR |
| 0xFF20 0170 | 4 bytes | Áudio Ctrl1 |
| 0xFF20 0174 | 4 bytes | Áudio Ctrl2 |
| 0xFF20 0178 | 4 bytes | NOTE\_SYSCALL\_ADDRESS |
| 0xFF20 017C | 4 bytes | NOTE\_CLOCK |
| 0xFF20 0180 | 4 bytes | NOTE\_MELODY |
| 0xFF20 0184 | 4 bytes | MUSIC\_TEMPO\_ ADDRESS |
| 0xFF20 0188 | 4 bytes | MUSIC\_ADDRESS |
| 0xFF20 018C | 4 bytes | PAUSE\_ADDRESS |
| ... | ... | Não Alocado |
| 0xFF20 0200 | 4 bytes | ADC\_CH0 |
| 0xFF20 0204 | 4 bytes | ADC\_CH1 |
| 0xFF20 0208 | 4 bytes | ADC\_CH2 |
| 0xFF20 020C | 4 bytes | ADC\_CH3 |
| 0xFF20 0210 | 4 bytes | ADC\_CH4 |
| 0xFF20 0214 | 4 bytes | ADC\_CH5 |
| 0xFF20 0218 | 4 bytes | ADC\_CH6 |
| 0xFF20 021C | 4 bytes | ADC\_CH7 |
| ... | ... | Não Alocado |
| 0xFF20 0500 | 4 bytes | IrDA\_CTRL |
| 0xFF20 0504 | 4 bytes | IrDA\_RX |
| 0xFF20 0508 | 4 bytes | IrDA\_TX |
| 0xFF20 0510 | 4 bytes | STOPWATCH\_ADDRESS |
| 0xFF20 0514 | 4 bytes | LFSR\_ADDRESS |
| 0xFF20 0520 | 4 bytes | KEYMAP0 00 a 1F |
| 0xFF20 0524 | 4 bytes | KEYMAP1 20 a 3F |
| 0xFF20 0528 | 4 bytes | KEYMAP2 40 a 5F |
| 0xFF20 052C | 4 bytes | KEYMAP3 60 a 7F |
| 0xFF20 0600 | 4 bytes | BREAK POINT ADDRESS |
| 0xFF20 0604 | 4 bytes | VIDEO FRAME SELECT |

Memória de Instruções (CodeMemory)

|  |  |  |
| --- | --- | --- |
| **Endereço** | **Tamanho** | **Uso** |
| 0x0040 0000 | 64 kibytes | .text  UserCodeBlock |
| ... |
| 0x0040 FFFF |
| ... |  | Não Alocado |

Memória de Dados (DataMemory)

|  |  |  |
| --- | --- | --- |
| **Endereço** | **Tamanho** | **Uso** |
| 0x1001 0000 | 128 kibytes | .data  UserDataBlock |
| ... |
| 0x1002 FFFF |
| 0x1003 0000 | 64 Mibytes | SRAM |
| ... |
| 0x1400 FFFF |
| ... |  | Não Alocado |

Valores definidos na inicialização do Processador:

Endereço da Base da Pilha: sp=0x1002 FFFC

Endereço do Contador de Programa: PC=0x0040 0000

Arquivos Default:

Memória de Programa do Usuário: de1\_text.mif

Memória de Dados do Usuário: de1\_data.mif

Os arquivos .mif devem ser gerados pelo MIF Dump Memory do RARS.

O arquivo SYSTEMv13.s contém a rotina de tratamento de exceções/interrupções e ecalls.

Deve-se configurar o Rars para:

-Habilitar Settings/Self-Modifying Code

-Desabilitar Settings/Exception Handler

Usar .include “..\macros2.s” no início e “..\SYSTEMv13.s” no final do programa que permite usar as definições .eqv dos endereços da tabela diretamente no programa.

* **Chamadas do Sistema:**

No endereço apontado por UTVEC encontra-se a rotina de tratamento de exceções/interrupções e chamadas do sistema:

Ecalls

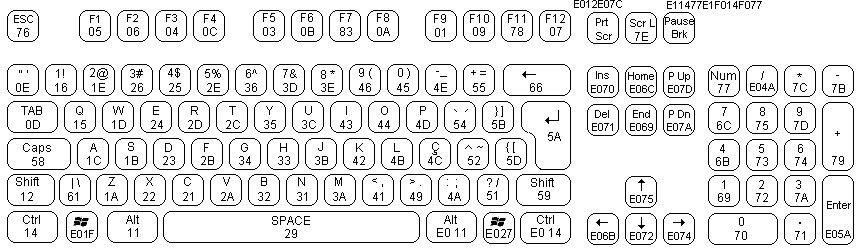
|  |  |  |  |
| --- | --- | --- | --- |
| **Serviço** | **$v0** | **Argumentos** | **Resultados** |
| Print Integer | 1  101 | a0=inteiro  a1=coluna  a2=linha  a3=cores  a4=frame | Imprime na frame a4 o número inteiro complemento de 2 a0 na posição (a1,a2) com as cores a3={0...0BBGGGRRRbbgggrrr} BGR fundo; bgr frente |
| Print Float | 2  102 | fa0=float  a1=coluna  a2=linha  a3=cores  a4=frame | Imprime na frame a4 o número float em fa0 na posição (a1,a2) com as cores a3 |
| Print String | 4  104 | a0=endereço string  a1=coluna  a2=linha  a3=cores  a4=frame | Imprime na frame a4 a string terminada em NULL (.string) presente no endereço a0 na posição (a1,a2) com as cores a3 |
| Read Int | 5  105 |  | Retorna em a0 o valor inteiro com sinal lido do teclado. |
| Read Float | 6  106 |  | Retorna em fa0 o valor float lido do teclado. |
| Read String | 8  108 | a0 endereço do buffer de entrada  a1 número de caracteres máximo | Retorna no endereço a0 o conjunto de caracteres lidos, terminando com /0. |
| Print Char | 11  111 | a0=char (ASCII)  a1=coluna  a2=linha  a3=cores  a4=frame | Imprime na frame a4 o caractere a0 (ASCII) na posição (a1,a2) com as cores a3 |
| Exit | 10  110 |  | Retorna ao sistema operacional. Na DE1-SoC trava o processador. |
| Read Char | 12  112 |  | Retorna em a0 código ASCII do caractere da tecla pressionada |
| Time | 30  130 |  | Retorna o tempo do sistema (número de ciclos de clock)  a0 = parte menos significativa  a1 = parte mais significativa (zero na DE1-SoC) |
| MIDI Out  assíncrono | 31  131 | a0 = pitch  a1 = duração ms  a2 = instrumento (1)  a3 = volume | Gera o som definido e retorna imediatamente |
| Sleep | 32  132 | a0=tempo(ms) | Aguarda a0 milissegundos |
| MIDI Out  síncrono | 33  133 | a0 = pitch  a1 = duração ms  a2 = instrumento (1)  a3 = volume | Gera o som definido e retorna apenas após o término |
| print integer  hexadecimal | 34  134 | a0=inteiro  a1=coluna  a2=linha  a3=cores  a4=frame | Imprime na frame a4, em hexadecimal, o número inteiro de 32 bits em a0 na posição (a1,a2) com as cores a3 |
| Rand | 41  141 |  | a0 = número randômico de 32 bits |
| Clear Screen | 48  148 | a0 = cor  a1=frame | Limpa a frame a1(0 ou 1) com a cor a0 |

Os ecalls 1xx são para utilização com o Bitmap Display Tool e o Keyboard Display MMIO Tool do Rars, e também funcionam de maneira idêntica com a DE1-SoC.

* **Interface Teclado PS2**

O teclado PS2 utiliza um protocolo que envia serialmente um código de 8 bits (scancode) para cada tecla pressionada. O “despressionamento” de uma tecla é identificada pelo envio do marcador 0xF0 antes do scancode da tecla que foi solta.

A figura abaixo mostra o mapeamento dos scancodes (em hexa) das teclas.



Teclas especiais (Pause, Prt Scr, setas direcionais, etc,) utilizam códigos de 2 ou mais bytes.

Há 3 formas implementadas para acessar os dados fornecidos por um teclado PS2.

1. Buffer de scancodes:

Nos endereços Buffer0Teclado e Buffer1Teclado está um buffer de 8 bytes que armazena os scancodes enviados pelo teclado.

Buffer1 Buffer0

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 5 |  | 4 | 3 | 2 | 1 |

O código mais recente está localizado no byte 1 do Buffer0 e o mais antigo no byte 4 do Buffer1, permitindo assim a análise da sequência de pressionamento das teclas ao longo do tempo.

1. Mapeamento das teclas (KeyMap):

Nas words dos endereços KeyMap0, KeyMap1, KeyMap2 e KeyMap3 está o mapeamento em 128 bits de todos os scancodes das teclas, onde bit 1 significa que a tecla está acionada no momento, 0 a tecla não está pressionada.

Ex.: Memoria[KeyMap0] & (1<<(29-1)) indica se a tecla W (scancode 0x1D = 29) está acionada ou não

A tecla ENTER (0x5A) é mapeada no bit 90, isto é, Memoria[KeyMap2] & (1<<(90-64)-1)

Deste modo pode-se verificar o pressionamento simultâneo de teclas.

Obs.: Cuidar, pois vários teclados PS2 comerciais não permitem o acionamento simultâneo de determinadas combinações de 3 ou mais teclas.

1. Compatibilidade com o Keyboard Display MMIO Tool do Rars

O endereço KDMMIO\_Ctrl possui no seu bit menos significativo a informação se há ou não uma tecla pressionada.

O endereço KDMMIO\_Data possui o código ASCII da tecla pressionada. Permitindo assim a detecção do pressionamento de uma tecla de cada vez.

Executar o testePS2.s e analisar o valor dos registradores mostrados na tela que fica mais claro as 3 formas de adquirir os dados do teclado

* **Interface VGA:**

A interface VGA é idêntica ao Bitmap Display Tool do Rars.

Estão implementados 2 frames buffers localizados nos endereços VGAADDRESSINI0 = 0xFF000\_0000 e VGAADDRESSINI1 = 0xFF10\_0000 selecionáveis pelo endereço VGAFRAMESELECT, que deve conter a word 0 ou 1, ou pela chave SW[6].

O arquivo frame0.mif é carregado na memória VGA frame0 e o arquivo frame1.mif na frame1 ao inicializar o processador.

Mapeamento da tela VGA para resolução 240 linhas(Y) × 320 colunas(X) pixels:

Mapeamento: (X,Y) => Endereço Base (0xFF**0**0 0000) + Y × 320 + X

0xFF**0**00000

0xFF**0**0013F

0xFF**0**12BFF

0xFF**0**12AC0

Com X de 0 a 319 e Y de 0 a 239.

|  |  |  |
| --- | --- | --- |
| 7-6 | 5-3 | 2-0 |
| BB | GGG | RRR |

Cor do Pixel: 8 bits

Obs.: A cor 0xC7 (Magenta) corresponde ao transparente, tanto na DE1-SoC quanto no Bitmap Display Tool do Rars.

Ver exemplo: testeVGA.s

Obs: O programa bmp2oac2.exe converte um arquivo .bmp (24 bits/pixel gerado pelo **paint.net**) para:

1. .bin, para uso no Rars (leitura de arquivo)
2. .mif para carregar diretamente na memória de vídeo VGA (via definição do arquivo default no .v)
3. .s em uma linha Assembly para incluir no .data

* **Interface de áudio CODEC:**

As amostras de 16 bits de áudio estéreo são lidas e escritas nos endereços indicados na tabela de mapa de memória a uma frequência de 44.1kHz.

As amostras são números de 16 bits similares às amostras de um arquivo de áudio no formato RAW.

Sincronismo do Processador com o CODEC segue o seguinte protocolo de comunicação:

1. Processador coloca o Ctrl2 em 0, indicando ao CODEC para enviar uma amostra;
2. Processador aguarda CODEC colocar Ctrl1 em 1, que indica que uma amostra está pronta;
3. Processador lê os valores nos endereços inL e inR e escreve os valores nos endereços outL e outR;
4. Processador coloca o Ctrl2 em 1, indicando ao CODEC que terminou de ler e escrever as amostras;
5. Processador aguarda CODEC colocar Ctrl1 em 0, que indica que o CODEC está pronto novamente
6. Reinicio do ciclo

Onde o bit [0], nas palavras de controle Ctrl1 e Ctrl2, corresponde ao canal R e o bit [1] ao canal L. Com isto é possível o controle, aquisição e síntese de áudio para os 2 canais de forma independente.

Ver exemplo: testeAUDIO.s

* **Interface de áudio: Sintetizador Polifônico (by Lucas Carvalho)**

O sintetizador de áudio utilizado no curso de OAC é um sintetizador polifônico compacto capaz de tocar até 8 notas simultaneamente. Foi baseado no padrão MIDI e é formado por três sistemas:

### Oscilador:

Gera formas de onda em várias frequências. É responsável por criar o som base de cada nota em 3 possíveis formatos:

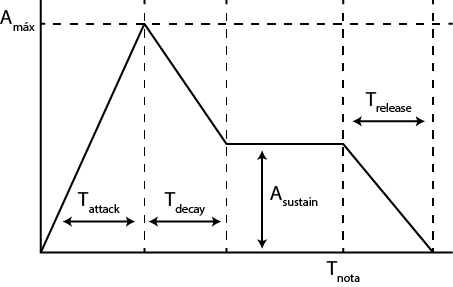


### Filtro:

Um filtro passa-baixas pode ser ativado ou não na saída do oscilador. Sua frequência de corte é a própria frequência da nota e seu fator de qualidade é configurável.

### Envelope:

O tempo de vida de uma nota é dividido em 4 seções: *Attack*, *Decay*, *Sustain* e *Release*. Elas determinam uma variação na amplitude da onda resultando num som mais natural.



# Interface

Para a implementação do sintetizador no RISC-V é necessário utilizar o módulo *Sintetizador*, que encapsula todos os sistemas.

/\*

\* Entradas do CODEC de áudio da própria DE2.

\*/

**input** AUD\_DACLRCK**;**

**input** AUD\_BCLK**;**

Nestas entradas devem ser passados os pinos da DE1-SoC de mesmo nome. São dois pinos de entrada que controlam a taxa de amostragem do CODEC.

/\*

\* Comando de início/fim de uma nota.

\*/

**input** NOTE\_PLAY**;**

**input** **[**6**:**0**]** NOTE\_PITCH**;**

O sintetizador funciona recebendo comandos para que inicie ou termine uma nota. Por exemplo, caso queira tocar 3 notas simultâneas, você deve enviar 3 comandos de início seguidos.

Para enviar um comando atribua a *NOTE\_PLAY* um valor de 1 para iniciar ou 0 para terminar uma nota. *NOTE\_PITCH* determina a nota deste comando, seguindo o padrão MIDI (pesquise por *MIDI Note Table*).

Em toda subida de *AUD\_DACLRCK*, um comando é lido. Comandos repetidos ou inválidos (terminar uma nota que nunca foi iniciada) são entendidos como se não houvesse comando.

/\*

\* Configurações do oscilador.

\*/

**input** **[**1**:**0**]** WAVE**;**

**input** NOISE\_EN;

A entrada *WAVE* define qual forma de onda será gerada pelo oscilador, como mostra a seguinte tabela de referência.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| Mudo | Senoide | Quadrada | Dente-de-serra |

O oscilador adicionará um certo ruído à onda gerada se *NOISE\_EN* assumir valor 1. Este mecanismo é útil na criação de efeitos sonoros. Sintetizadores de vídeo games antigos utilizavam uma técnica similar para simular explosões, impacto, dano etc.

/\*

\* Configurações do filtro.

\*/

**input** FILTER\_EN;

**input** **[**7**:**0**]** FILTER\_QUALITY**;**

O filtro passa-baixas é ativado se a entrada *FILTER\_EN* estiver alta. Sua frequência de corte é a própria frequência da nota sendo filtrada e seu fator qualidade pode ser configurado pela entrada *FILTER\_QUALITY*, que é interpretada como ponto fixo Q8.

O fator de qualidade controla a banda de frequências que passa pelo filtro. Caso assuma o valor zero, o filtro se comporta como um passa-baixas comum, sem ressonância. Uma onda quadrada filtrada com fator de qualidade alto se aproxima bastante de uma senoide.

/\*

\* Configurações do envelope aplicado durante a vida de uma nota.

\*/

**input** **[**6**:**0**]** ATTACK\_DURATION**;**

**input** **[**6**:**0**]** DECAY\_DURATION**;**

**input** **[**6**:**0**]** SUSTAIN\_AMPLITUDE**;**

**input** **[**6**:**0**]** RELEASE\_DURATION;

Toda nota tem uma envoltória aplicada a sua onda enquanto é tocada. Uma tecla de piano, ao ser pressionada, gera um som alto inicial que decai e mantém uma altura constante. Quando a tecla é solta a nota some aos poucos até ficar inaudível. Este processo é simulado em sintetizadores por meio do envelope de 4 seções mostrado anteriormente.

A duração de cada fase é configurável, exceto a de *Sustain* cuja amplitude é variável. A escala destes valores é, assim como definido no padrão MIDI, dependente da implementação, ou seja, é necessária uma certa experimentação.

/\*

\* Amostra de saída do sintetizador.

\*/

**output** **[**15**:**0**]** SAMPLE\_OUT;

Os sistemas internos do sintetizador processam todas as notas e produzem uma nova amostra a cada subida do clock de amostragem *AUD\_DACLRCK*. Esta amostra deve ser repassada ao CODEC da DE1-SoC. Este processo não é parte do sintetizador e já está implementado no processador RISC-V do curso.

* **Interface de áudio: Sintetizador Polifônico (Grupo 1 - 2015/1 by Luiz Henrique Campos)**

1 - Foi criada uma memória de dados de duas portas UserDataBlockDouble. Uma porta de escrita e leitura para o uso da CPU e uma porta somente de leitura para o sintetizador (na pasta item6/core/memoria). FOI RETIRADA ESSA CARACTERÍSTICA

2 - Outra PLL foi usada para gerar o AUD\_CTRL\_CLK e "afinar" o sintetizador.

3 - Outros Parâmetros adicionados :

NOTE\_SYSCALL\_ADDRESS = 32'hFF20 0178,

NOTE\_CLOCK = 32'hFF20 017C,

NOTE\_MELODY = 32'hFF20 0180,

MUSIC\_TEMPO\_ADDRESS = 32'hFF20 0184,

MUSIC\_ADDRESS = 32'hFF20 0188,

PAUSE\_ADDRESS = 32'hFF20 018C

4 - Adição do módulo SynthControl que toca a música sequencialmente na memória de dados está na pasta **/core/Sintetizador**.

Existem 2 tipos de Configuração para a nota na memória, uma para ecall e outra para a música que será tocada sequencialmente. FOI RETIRADA ESTA CARACTERÌSTICA

**Configuração para notas que serão tocadas sequencialmente**:

**Melody** – 1 se for melodia; 0 se for acorde. Se esse bit for 1, a próxima nota só será tocada ao fim desta. Se for zero, a próxima nota será tocada na próxima borda de subida de AUD\_DACLRCK.

**Instrument** – 16 possibilidades de instrumentos (0 - 15).

**Volume** - 128 possibilidades de amplitude da nota (0 – 127).

**Pitch** – 128 possibilidades de notas musicais segundo o padrão MIDI (0 – 127).

**End** – A última nota da música deverá setar esse bit para 1 para o controlador do sintetizador parar de percorrer a memória.

**Repeat** – Se esse bit estiver setado para 1, a próxima nota será a nota salva no endereço inicial.

**Duration (Q4)** – Duração em ponto fixo de cada nota. O valor é relativo à duração em milissegundos da Nota Base (Semínima). O valor da duração da Nota Base em milissegundos será passado para o endereço 0xFFFF020C. A partir daí o hardware atribuirá a duração em milissegundos equivalente para cada nota baseado em seus 8 primeiros bits (Duração – Ponto Fixo Q8).

„= Semibreve 4 x Nota Base (0100.0000)

Ó= Mínima 2 x Nota Base (0010.0000)

Œ= Semínima Nota Base (0001.0000)

‰= Colcheia ½ x Nota base (0000.1000)

Ù= Semicolcheia ¼ x Nota Base (0000.0100)

Û= Fusa 1/8 x Nota Base (0000.0010)

Ì= Semifusa 1/16 x Nota Base (0000.0001)

Por exemplo, se a duração da Nota base for 400 milissegundos, a duração de (0000.0010) será 400 x 1/8 = 50 milissegundos.

**Configuração para notas do ecall (31 e 33)**:



A word será salva no endereço 0xFF10 0200.

**Melody** – 1 se for ecall 33; 0 se for ecall 31. Em resumo, se esse bit for 1, a próxima nota só será tocada ao fim desta. Se for zero, a próxima nota será tocada na próxima borda de subida de AUD\_DACLRCK.

**Instrument** – 16 possibilidades de instrumentos (0 - 15). (SÓ 1 IMPLEMENTADO)

**Pitch** – 128 possibilidades de notas musicais segundo o padrão MIDI (0 – 127).

**Duration** – Duração da nota em milissegundos.

* **Interface Infravermelho: (Grupo 2 02/2016)**

A sigla IRDA (*Infrared Data Association*) faz referência a um conjunto de protocolos para comunicação sem fio entre dispositivos. A transferência de dados é feita na forma de pacotes que são enviados serialmente. A transmissão começa com o envio de 1 bit de início, seguido por 8 bits de dados e 1 bit de paridade e termina com a transferência de 1 bit de parada. Não é possível enviar e receber pacotes de dados simultaneamente. Para realizar a comunicação entre dispositivos, são necessárias uma porta de recepção (*receiver*) e uma porta de transmissão (*transmitter*).

|  |  |  |
| --- | --- | --- |
| IRDA\_CONTROL | Controle do IRDA | 0xFF20 0500 |
| IRDA\_RXD | Receptor IRDA | 0xFF20 0504 |
| IRDA\_TXD | Transmissor IRDA | 0xFF20 0508 |

IRDA\_CONTROL (4 bytes)- wReadData[1] indica que tem dado pronto para leitura e wReadData[0] se existe dado sendo transmitido.

IRDA\_RXD (4 bytes)- é de onde vai ser lido o dado do Irda

IRDA\_TXD (4 bytes)– é o dado que vai ser transmitido pelo Irda (não tem na DE2-70, e não foi testado na DE1-SoC)

* **Interface Stopwatch: (Grupo 2 02/2016)**

STOPWATCH\_ADDRESS = 0xFF20 0510

Retorna o tempo em ms desde que a FPGA foi ligada.

Obs.: Muito útil para fazer temporização no jogo.

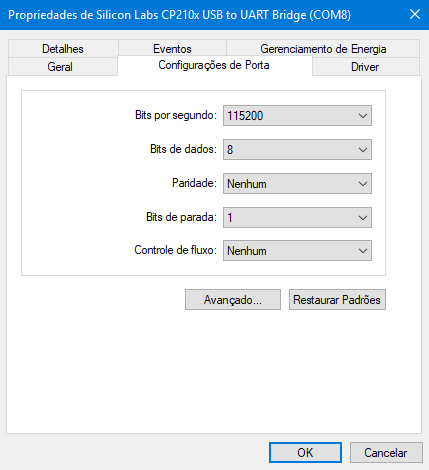
* **Interface LSFR: (Grupo 2 02/2016)**

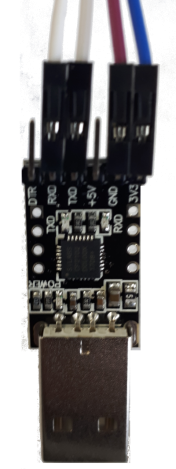
LFSR\_ADDRESS = 0xFF20 0514

Retorna um número pseudoaleatório que foi gerado pela técnica de *Linear Feedback Shift Register* (LFSR).

* **Interface Serial RS232C/USB:**

Comunicação do processador RISC-V com o PC é feita pela transmissão/recepção serial byte a byte através de um módulo conversor RS232C/USB CP1202 para Arduino, mostrado na figura abaixo:





GND

3.3V

Tx

Rx

GND

GPIO\_0[26] : Rx

Tx : GPIO\_0[27]

3.3V

Este módulo cria uma porta serial COM no PC.

A configuração desta porta, através do Gerenciador de Dispositivos / Portas (COM e LPT), deve seguir ao mostrado na figura acima.

No programa em C, a ser executado no PC, a linha que define o número da porta a ser usada e o baud rate é:

int cport\_nr = 7, /\* usar o número da COM - 1 \*/

bdrate = 115200;

Endereços de MMIO do RISC-V:

RX\_ADDRESS = 0xFF200120

TX\_ADDRESS = 0xFF200121

CRTL\_ADDRESS = 0xFF200122

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | x | x | x | x | Ready | Busy | Start |

Byte de Controle:

Procedimentos:

Transmissão: RISC-V 🡪 PC

1. RISC-V escreve o byte a ser enviado no endereço TX
2. RISC-V ativa o bit Start do controle no endereço CTRL
3. RISC-V desativa o bit Start do controle no endereço CTRL
4. RISC-V aguarda o bit Busy do controle ser desativado para fazer nova transmissão

Recepção: RISC-V 🡨 PC

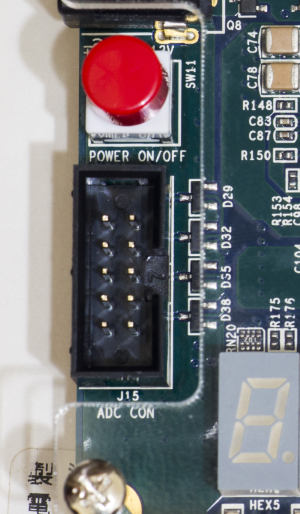
1. RISC-V aguarda o bit Ready do controle ser ativado
2. RISC-V lê o byte recebido do endereço RX
3. RISC-V aguarda o bit Ready do controle ser desativado

Ver exemplos: testeRS232.s e testeRS232.c, demo\_tx.c, demo\_rx.c

Detalhes em http://www.fpga4fun.com/SerialInterface.html

* **Interface com Conversor Analógico-Digital: (falta implementar!)**

A DE1-SoC possui um conversor ADC com 8 canais de 12 bits cada. O acesso se dá através dos pinos da interface ADC COM abaixo listados.



GND

CH7

CH5

CH3

CH1

+5V

CH6

CH4

CH2

CH0

Os sinais de entrada analógicos devem estar entre 0 e 5V, e os valores convertidos em 12 bits são acessáveis pelos endereços:

ADC\_CH0 = 0xFF200200

ADC\_CH1 = 0xFF200204

ADC\_CH2 = 0xFF200208

ADC\_CH3 = 0xFF20020C

ADC\_CH4 = 0xFF200210

ADC\_CH5 = 0xFF200214

ADC\_CH6 = 0xFF200218

ADC\_CH7 = 0xFF20021C

A frequência de amostragem deve ser controlada pelo software, pois não há sinal de interrupção.

Ver exemplo: testeADC.s