Projetando um periférico escravo AXI-lite personalizado

AULA 9

IOUL IIA SKL I AROVA

Hardware personalizado

Blocos de hardware personalizados são usados para:

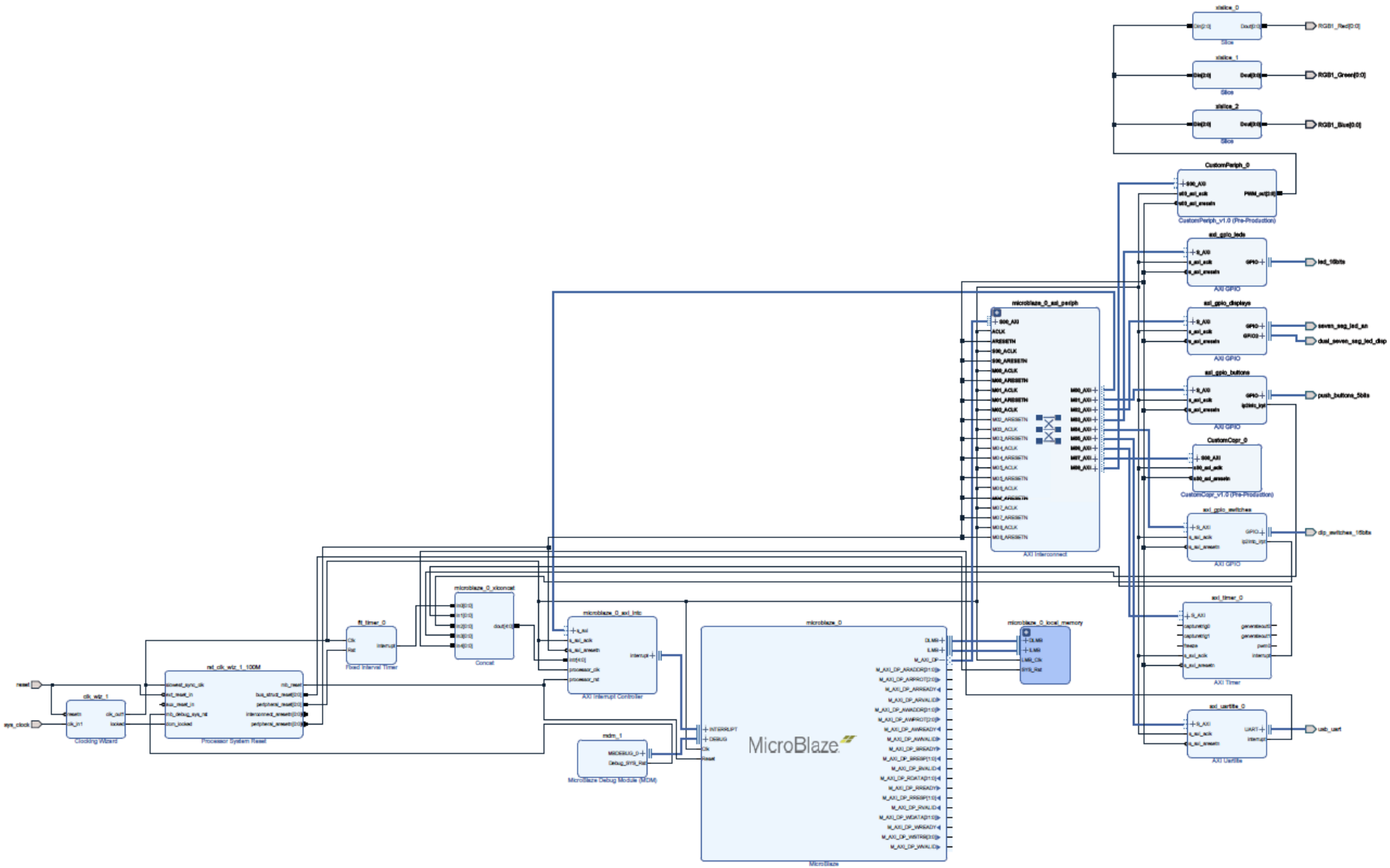
- Delegar ao hardware funções críticas em termos de tempo
- Economize recursos do MicroBlaze

Um exemplo será considerado, que inclui:

- Um coprocessador personalizado sem portas de saída
- Um periférico personalizado com portas de saída

O próximo laboratório (lab. 7) será dedicado ao design e uso de um IP periférico personalizado – Display Driver.

Design de Bloco (BD)



Protocolo AXI4

Consultar**Guia de referência do Vivado AXI**e**Especificação do protocolo AMBA AXI**

Existem três tipos de interfaces AXI4:

- **AXI4**: Para requisitos de mapeamento de memória de alto desempenho.
- **AXI4-Lite**: Para comunicação mapeada em memória simples e de baixo rendimento (por exemplo, de e para registros de controle e status).
- **Fluxo AXI4**: Para transmissão de dados em alta velocidade.

AXI-Lite:

- todas as transações têm comprimento de rajada 1
- todos os acessos a dados usam toda a largura do barramento de dados
- AXI4-Lite suporta largura de barramento de dados de 32 ou 64 bits
- todos os acessos são não modificáveis, não podem ser armazenados em buffer
-

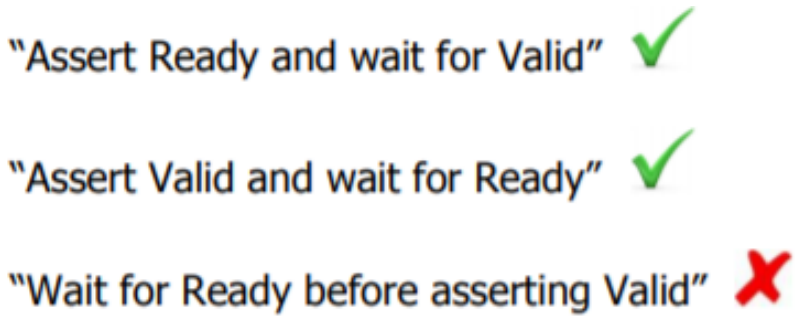
Global	Write address channel	Write data channel	Write response channel	Read address channel	Read data channel
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
–	AWADDR	WDATA	BRESP	ARADDR	RDATA
–	AWPROT	WSTRB	–	ARPROT	RRESP

Sinais de aperto de mão AXI4-lite

Consistente nos cinco canais.

Baseado em um princípio simples de “Pronto” e “Válido”:

- “Pronto” é usado pelo destinatário para indicar que está pronto para aceitar uma transferência de dados ou valor de endereço.
- “Válido” é utilizado para esclarecer que os dados (ou endereço) fornecidos naquele canal pelo remetente são válidos para que o destinatário possa então experimentá-los.



Exemplo 1 – Somador com 3 Operandos

Importar design de bloco

Criar e empacotar novo IP (*CustomCopr*)

Adicionar IP

Editar no IP Packager

CustomCopr:

- Adiciona o conteúdo de 3 registros (escritos por software)
- Coloca o resultado em 4◦registrar (ler por software)

Aplicar opções da “Aula 6” (Problemas e Resultados – slide 19)

Gerar produtos de saída

Criar wrapper HDL

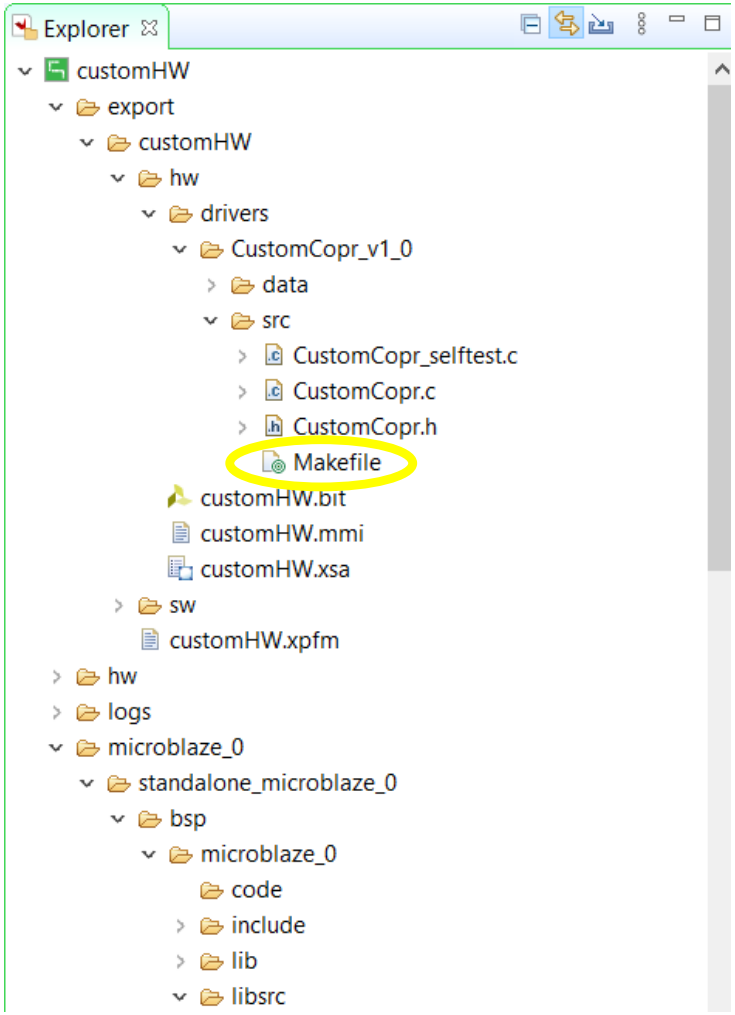
Gerar fluxo de bits

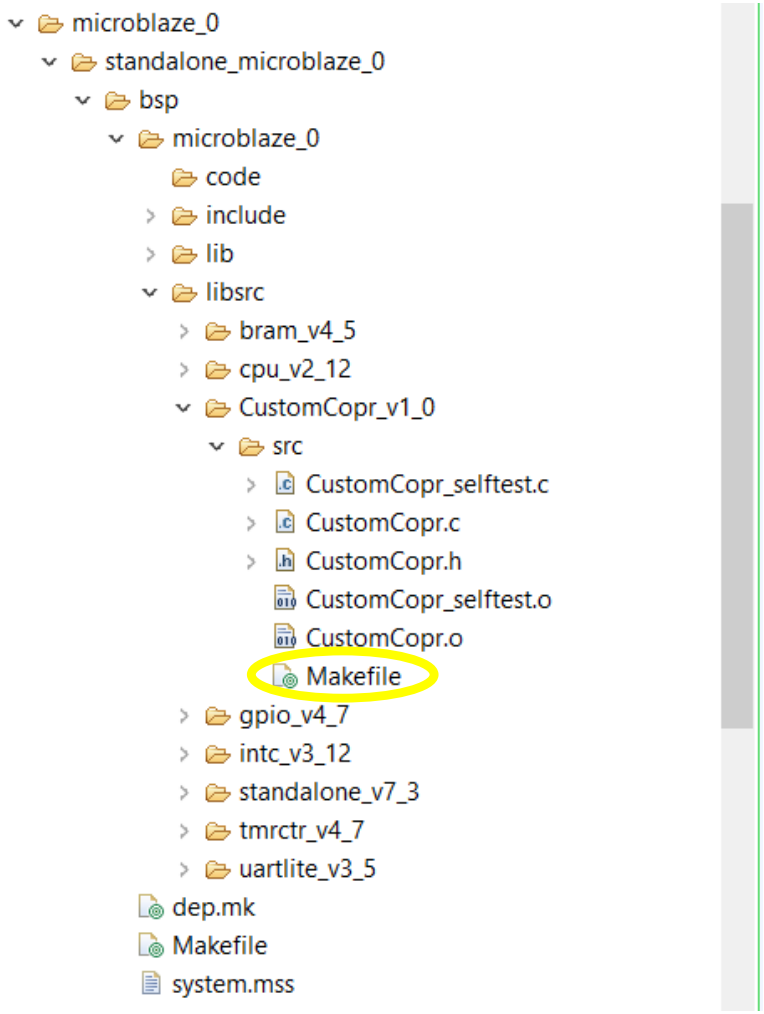
Exportar hardware

Lançar Vitis

Vitis

Se erros de compilação aparecerem, atualize**todos os makefiles**relacionado ao IP personalizado:





Makefile correto

```
COMPILADOR=
ARQUIVADOR =
CP=cp
COMPILER_FLAGS=
EXTRA_COMPILER_FLAGS=
LIB=libxil.a

RELEASEDIR=../../../lib
INCLUDEDIR=../../../include INCLUDES=-
I./. -I${INCLUDEDIR}

INCLUDEFILES=$(curinga *.h)
LIBSOURCES=$(curinga *.c)

OBJETOS = $(addsuffix .o, $(basename $(wildcard *.c))) ASSEMBLY_OBJECTS = $
(addsuffix .o, $(basename $(wildcard *.S)))

bibliotecas:
        echo "Compilando CustomCopr..."
        $(COMPILER) $(COMPILER_FLAGS) $(EXTRA_COMPILER_FLAGS) $(INCLUDES) $(LIBSOURCES) $(ARCHIVER) -r
        ${RELEASEDIR}/${LIB} ${OBJECTS} ${ASSEMBLY_OBJECTS}
        fazer limpo

incluir:
        ${CP} $(INCLUDEFILES) $(INCLUDEDIR)

limpar:
        rm -rf ${OBJETOS} ${ASSEMBLY_OBJECTS}
```

Depois de corrigir os Makefiles

1. Limpe os projetos

2. Construa a plataforma de hardware

3. Analise o arquivoparâmetros x.h(corrija o código principal se necessário)

4. Construa o aplicativo de software

5. Execute o aplicativo (código fonte disponível no eLearning)

LEDs tricolores Nexys-4

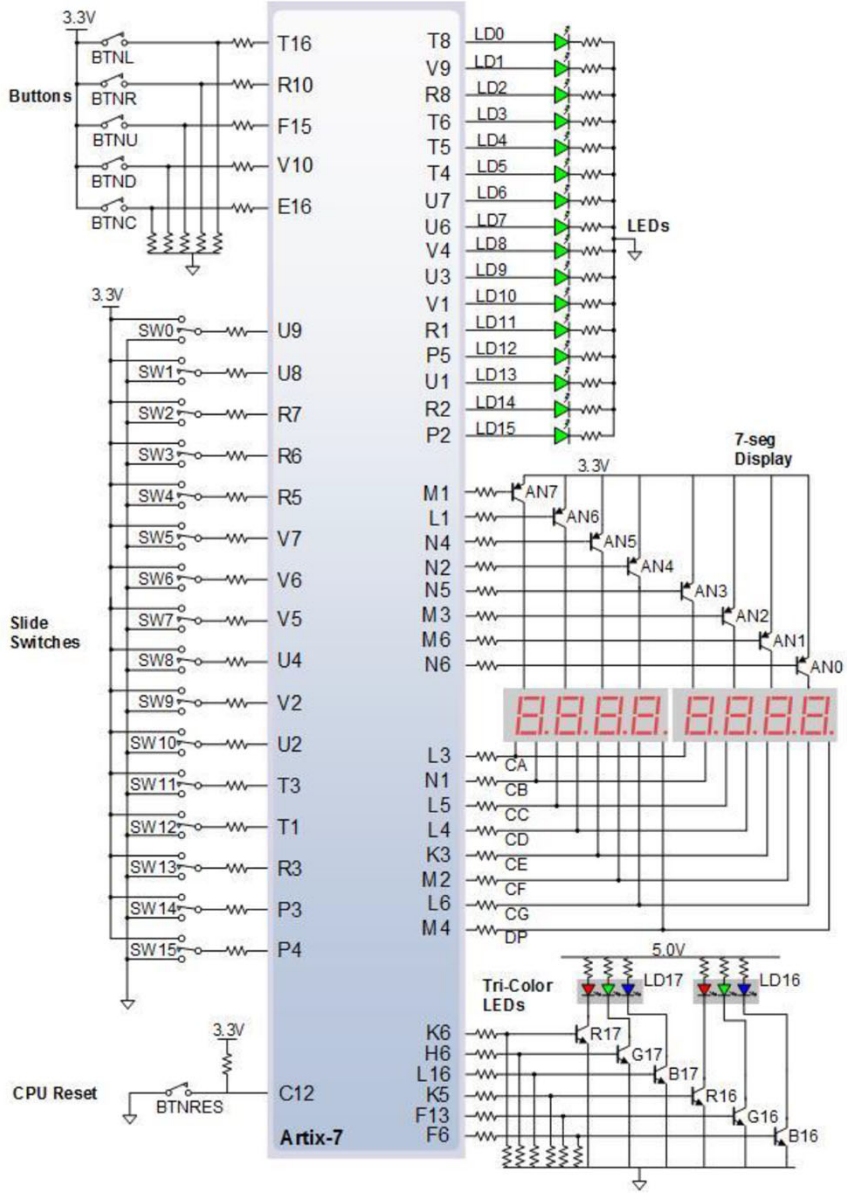
Cada LED tricolor possui três sinais de entrada que acionam os cátodos de três LEDs internos menores: um vermelho, um azul e um verde.

Dirigir o sinal correspondente a uma dessas cores para alto iluminará o LED interno.

Os sinais de entrada são acionados pelo FPGA através de um transistor, que inverte os sinais. Portanto, para acender o LED tricolor, os sinais correspondentes precisam ser elevados.

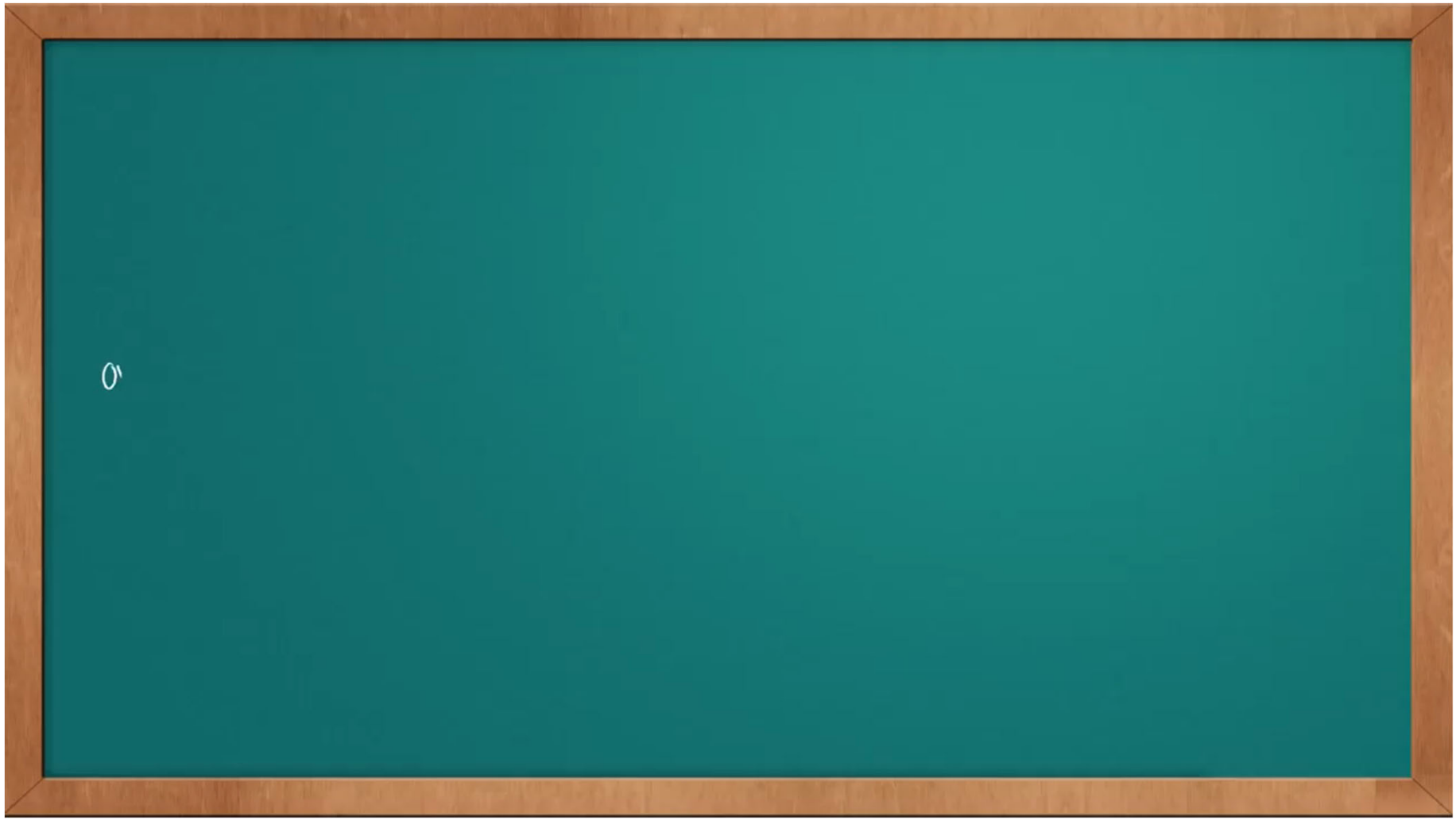
O LED tricolor emitirá uma cor dependente da combinação de LEDs internos que estão sendo iluminados no momento.

Nota: Conduzir qualquer uma das entradas para uma lógica estável '1' resultará na iluminação do LED em um nível desconfortavelmente brilhante. Você pode evitar isso garantindo que nenhum dos sinais tricolores seja acionado com um ciclo de trabalho superior a 50%.



Exemplo 2 – Gerador PWM triplo

PWM – Modulação por Largura de Pulso



PWM

Frequência–quantas vezes por segundo o LED é ligado e desligado:

- Não pode ser muito lento para evitar cintilação

Resolução-indica quantas etapas intermediárias (também chamadas de “unidades”) um ciclo PWM possui:

- No modo de 8 bits, existem 256 níveis de brilho

Uma maneira fácil de construir um gerador PWM é usar dois contadores:

- Contador 1 (*s_clkEnbCnt*) limita a frequência dos pulsos PWM
- Contador 2 (*s_pwmCounter*) controla o ciclo de trabalho dos pulsos PWM (largura de pulso) dentro da resolução escolhida
- Frequência PWM =

  = relógio do sistema / (resolução * counter_1_MaxValue)

Projeto Final - Operação

Selecione um**Operação**adequado para hardware (para aumentar o desempenho, para ter uma implementação mais clara)

- uma instrução/função não suportada por MB (pop~~cnt, op~~erações vetoriais, lógica bit a bit/shift complexa, periférico específico, criptografia...)

Não são permitidas operações repetidas entre os alunos Não

são permitidas operações consideradas durante as aulas

Traga suas propostas para o laboratório no dia 16 de maio ou envie-as para mim por e-mail

- título do projeto
- breve descrição da funcionalidade (uma frase)
- breve descrição da arquitetura proposta
- por que usar o módulo de hardware personalizado sugerido?
- procedimento de teste e interação do usuário

Projeto Final – Exemplo de Proposta

Título do projeto:

- Acelerando a contagem populacional com um coprocessador de hardware

Breve descrição da funcionalidade (uma frase):

- Sistema com acelerador de hardware para cálculo da contagem populacional em uma matriz de comprimento configurável de vetores de 32 bits

Breve descrição da arquitetura proposta:

- O sistema incluirá um módulo de hardware personalizado que executa a operação de contagem de população em uma entrada de 32 bits. O módulo irá interagir com o MicroBlaze através da interface AXI-Stream. Os resultados parciais serão acumulados em software. O desempenho das implementações de software e hardware será analisado e comparado.

Por que usar o módulo de hardware personalizado sugerido?

- Para reduzir o tempo de processamento.

Procedimento de teste e interação do usuário

- Os dados de entrada serão gerados aleatoriamente. O software verificará os resultados do hardware. Testbench para o acelerador. Interação do usuário através do UARTLite e terminal serial.

Laboratório. 7

Direcione as funções de atualização de exibição para hardware personalizado

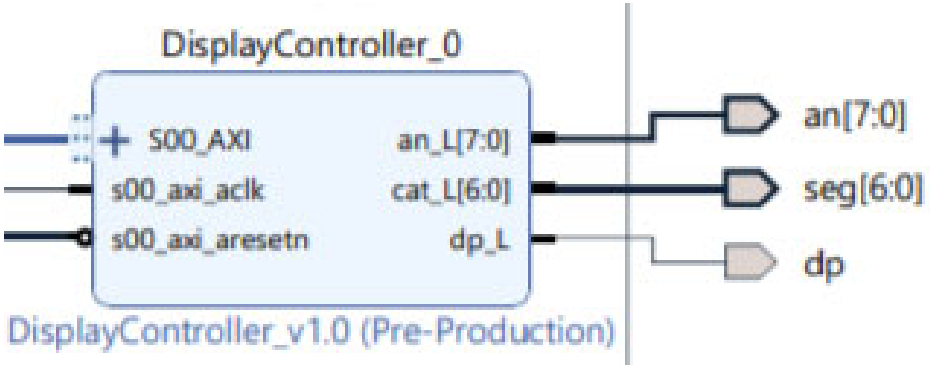
O driver de vídeo personalizado receberá do MicroBlaze (através da interface AXI-Lite):

- Habilita dígitos – 8 bits
- O ponto decimal permite - 8 bits
- Valores de dígitos - 32 bits (8 x 4 bits)

O driver de vídeo personalizado produzirá em suas saídas:

- um – 8 bits
- segmento – 7 bits
- DP – 1 bit

Os GPIO_Displays originais devem ser excluídos



Considerações finais

Ao final desta palestra você deverá ser capaz de:

- Projete módulos de hardware personalizados interagindo com o MicroBlaze por meio da interface AXI-Lite
- escrever programas C que fazem uso de hardware personalizado

Pendência:

- Construa as plataformas de hardware consideradas
- Teste os aplicativos fornecidos no Vitis
- Faça laboratório. 7