Uso do Signal Tap no projeto Controle de Temperatura Veicular

Uma das ferramentas utilizadas no projeto de Controle de Temperatura Veicular foi o Signal Tap, na qual foi possível verificar, ainda no estágio de desenvolvimento, o correto funcionamento dos módulos presentes. A utilização do Signal Tap foi de suma importância, uma vez que não havia a disponibilidade de outra ferramenta para a visualização de dados. Ao final do projeto, o funcionamento por completo do projeto pode ser verificado pelo visor, sem a necessidade do Signal Tap. Este documento relata como a ferramenta foi adotada para auxiliar durante o processo de desenvolvimento.

Certificando a integridade dos dados durante a comunicação

O seguinte mapeamento dos pinos entre as placas (Tiva e FPGA) foi utilizado para realizar a comunicação considerando o barramento de controle (endereços):

	TIVA	FPGA
LSB	PP4	Arduino_IO[9]
	PN5	Arduino_IO[10]
MSB	PN4	Arduino_IO[11]

Assim, se quisermos transmitir da TIVA para a FPGA o valor 001 no barramento de controle, basta configurar PP4 = 1, PN5 = 0 e PN4 = 0 conforme destacado na Figura 1 (Transmissão de dados de controle).

Transmissão de dados de controle

```
GPIOPinWrite(GPIO_PORTP_BASE, GPIO_PIN_4, GPIO_PIN_4);     //controlBus[0]
GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_5, 0);
GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_4, 0);              //MSB

//Ativando o write enable da comunicação
GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, GPIO_PIN_5);
```

Transmissão de dados

```
GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, GPIO_PIN_4);     //MSB
GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, GPIO_PIN_5);
GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_0, 0);
GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_1, GPIO_PIN_1);
GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_2, GPIO_PIN_2);
GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_3, 0);
GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4, 0);              //LSB

//Desativando o write enable da comunicação
GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5, 0);
```

Figura 1: Exemplo de código para envio de dados na TIVA.

Com isso, para conseguir conferir a transmissão de dados, foi configurado um signal tap para conferir se os dados de controle estão sendo recebidos de forma correta, conforme a Figura 2.

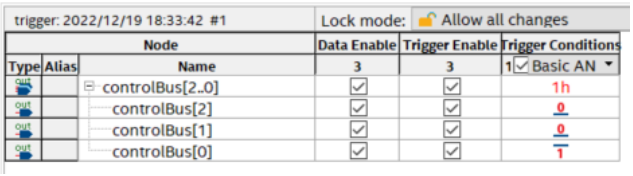

Figura 2: Lógica de trigger utilizada para o Signal Tap da comunicação.

Assim, quando o programa na TIVA chegar no ponto localizado pela seta em vermelho da Figura 1, o signal tap já deve ter identificado o valor 001 no barramento de controle (Arduino_IO[9] = 1, Arduino_IO[10] = 0 e Arduino_IO[11] = 0) se os dados foram transmitidos corretamente. A Figura 3 mostra o resultado do signal tap ao identificar o valor 001, demonstrando que a transmissão foi realizada corretamente.

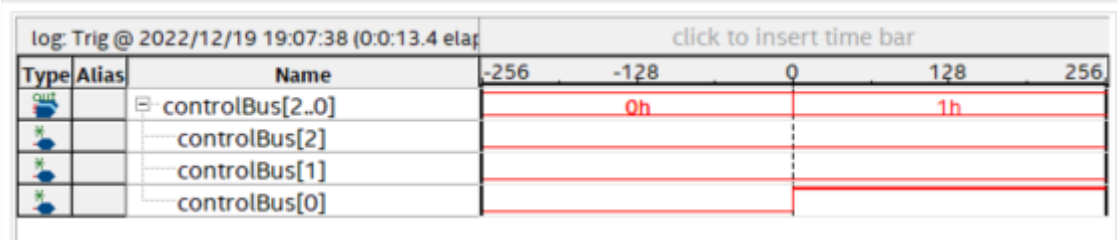

Figura 3: Resultado do Signal Tap ao identificar o valor 001 no barramento de controle.

Vale ressaltar que outros testes análogos foram realizados para o barramento de controle, certificando o funcionamento por completo da comunicação de dados TIVA → FPGA.

Certificando o correto acesso à memória RAM durante a escrita de dados na VGA

Para a escrita de dados na VGA, um bloco denominado ‘vga_sync’ na FPGA é responsável por realizar a varredura pixel por pixel, conforme a Figura 4, tendo como saída os valores para pixel_x (aout) e pixel_y (bout).

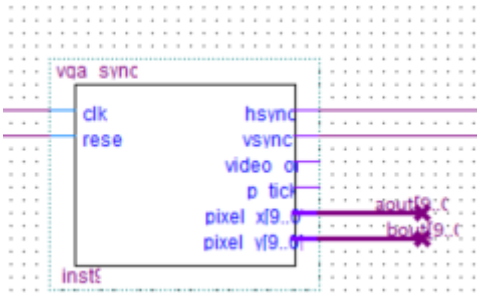

Figura 4: vga_sync

Assim, um outro bloco do projeto denominado ‘wrapper’ é responsável por receber o pixel de saída do vga_sync e determinar se ele deve ser pintado ou não. O bloco wrapper trabalha considerando diferentes regiões do visor, conforme a ilustração da Figura 5. Assim, como as informações de Temperatura Real Medida, Temperatura Desejada, Temperatura Da Mistura, Saída Selecionada, Velocidade Do Ventilador, Abertura Válvula Quente e Abertura Válvula Fria estão armazenadas em memória nessa ordem, portanto, conforme o

valor de pixel_y (bout), o endereço de memória deve ser escolhido corretamente para que nesta região o valor correto seja impresso no visor. Por exemplo, se bout = 79, então o endereço deve ser 001, se bout = 127, então o endereço deve ser 010, e assim por diante.

Para isso, um bloco foi desenvolvido com o intuito de receber como entrada o valor de bout e retornar o valor de endereço que deve ser acessado na memória RAM. O bloco está representado na Figura 6.

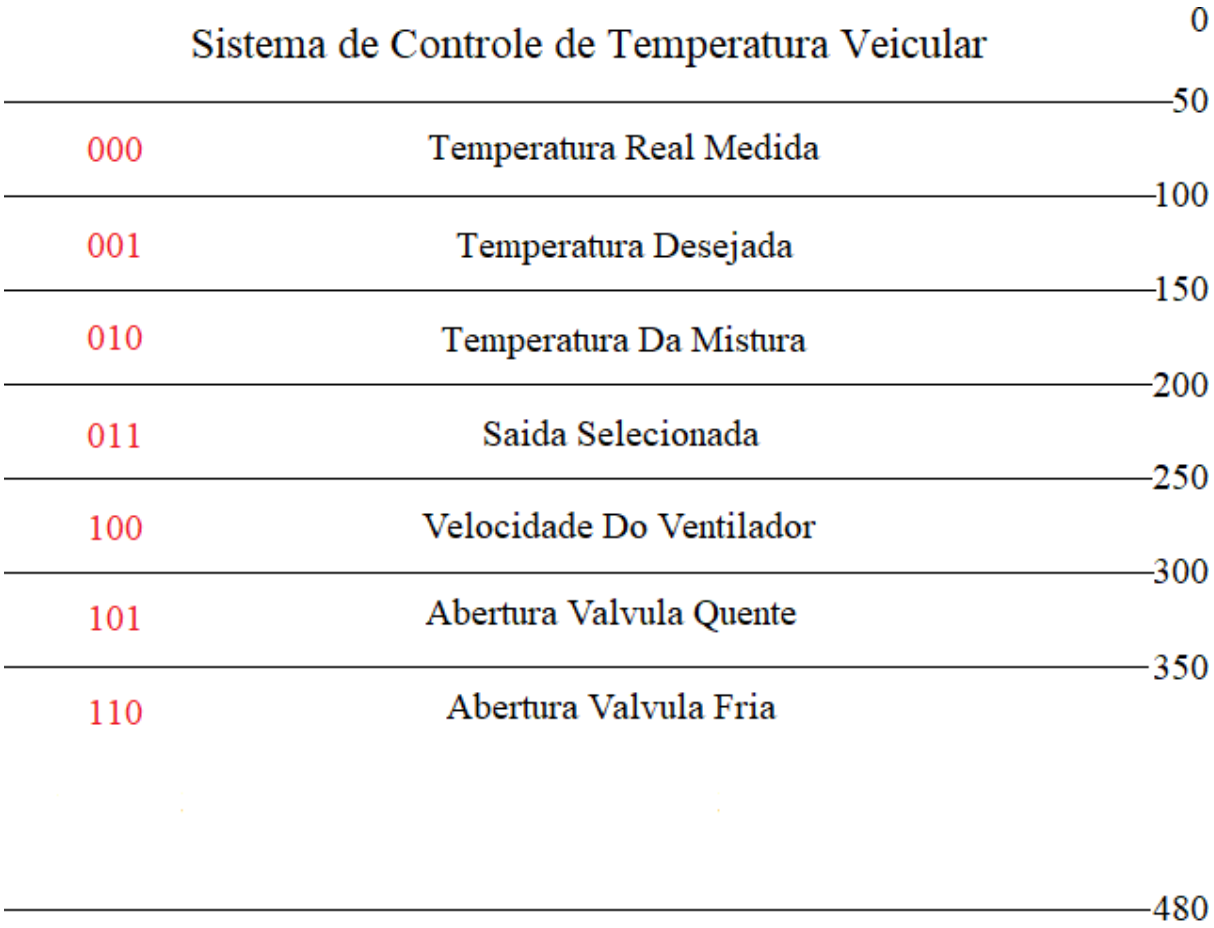

Figura 5: Regiões do visor (valores à direita indicam os limites das regiões e valores à esquerda indicam os endereços a serem acessados na memória RAM).

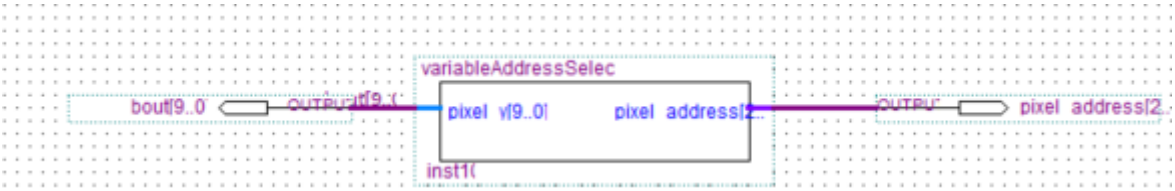

Figura 6: Seletor de endereço de memória com base no valor da coordenada y do pixel.

Assim, para verificar se o bloco da Figura 6 está funcionando corretamente, foi configurado um Signal Tap cuja lógica do trigger está ilustrada na Figura 7.

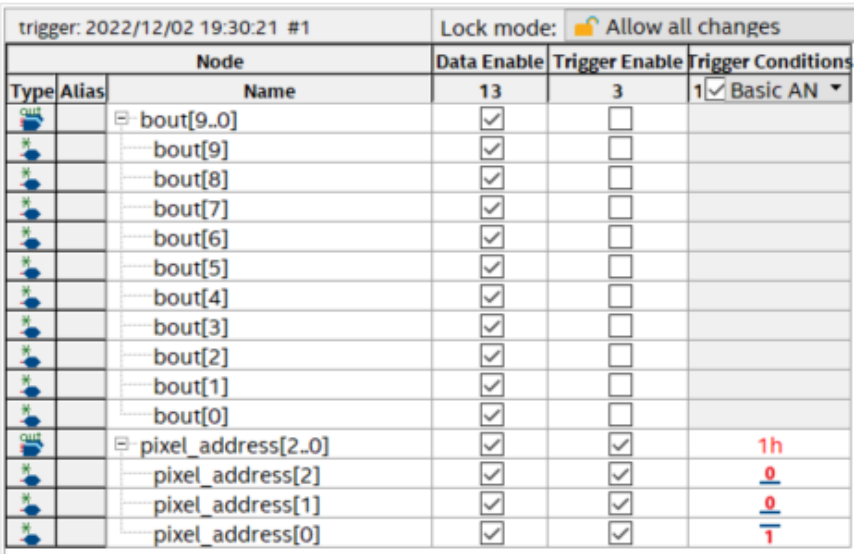

Figura 7: Lógica de trigger utilizada para o Signal Tap de endereço de RAM.

O Signal Tap configurado deve sinalizar quando o valor a ser acessado na memória RAM seja 001, ou seja, o valor de bout deve estar entre 50 e 100 (Neste momento o wrapper estaria trabalhando para imprimir a informação de Temperatura Desejada).

A Figura 8 mostra o resultado do signal tap ao identificar o valor 001 para o endereço a ser acessado na RAM, na figura vemos que o trigger foi disparado no momento que bout valor 0x65 (101 em decimal), certificando assim o funcionamento da lógica.

Vale ressaltar que inúmeros testes foram realizados com o signal tap para averiguar o funcionamento das outras regiões do visor.

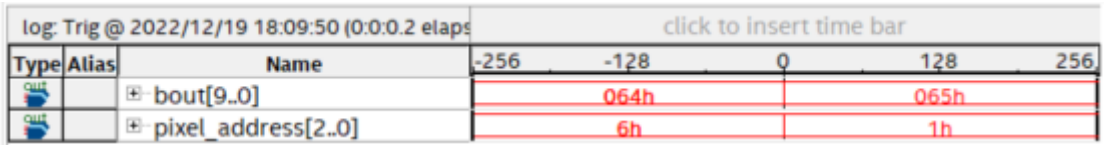

Figura 8: Resultado do Signal Tap ao identificar o valor 001 no endereço a ser acessado na RAM.