Transmissão de dados em série

INESC-TEC

12 de Julho de 2017
v1.0

Data	Autor	Edição	Alterações
11 julho 2017	Marisa Oliveira	1.0	Lançamento Inicial

1 Introdução

Este manual apresenta os aspetos relevantes sobre a arquitetura implementada em FPGA que permite a transmissão a transmissão de dados em série de uma barra de cores gerada na FPGA e também da imagem proveniente da fonte HDMI.

2 Objetivo

Esta arquitetura tem como principal objetivo a transmissão em série de uma barra de cores em *FULL HD* com uma taxa de atualização vertical de 60 Hz e também da imagem proveniente da fonte HDMI para o dispositivo final. São abordadas duas arquiteturas desenvolvidas: a arquitetura D (transmissão apenas da barra de cores) e arquitetura E (transmissão da barra de cores e dados de imagem proveniente da fonte HDMI).

É necessário ter que fase inicial do projeto optou-se por abordar de uma maneira direta a transmissão dos dados em série, sem o recurso à definição de todas a tramas de um pacote. Tal decisão foi tomada, ciente da importância das tramas num protocolo de comunicação, pois o módulo GTX disponibilizado pela *Xilinx* é muito complexo.

Para além disso é necessário também ter em conta os diferentes domínios de relógio que se encontrarão nestas implementações ilustrados na Fig. 1.

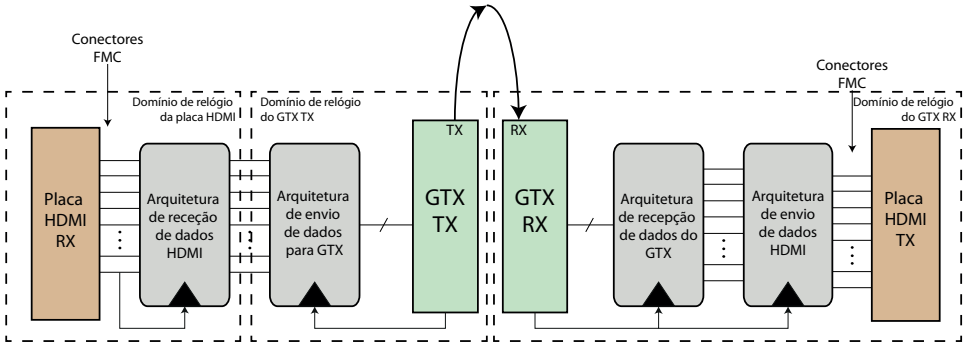

Figura 1: Dominios de relógio nestas implementações

3 Material Utilizado

Para a implementação destas arquitetursa são utilizados vários equipamentos, entre os quais os seguintes:

3.1 FPGA VC7203

É uma FPGA (*Field-programmable gate array*) que se caracteriza pelo seu elevado número de recursos e também pelas entradas em saídas de alta velocidade que possui, tal como indica (1). É utilizada para implementação do código desenvolvido em Verilog para esta arquitetura e ainda para conexão à placa HDMI pelos conectores FMC (*FPGA Mezzanine Card*).

3.2 TB-FMCH-HDMI2-TX e TB-FMCH-HDMI2-RX

Estas placsa HDMI caracterizam-se pela capacidade de transmissão e receção de dados HDMI. Para esta implementação, a placa deve estar configurada por omissão cujos detalhes se encontram em (2).

4 Arquitetura

4.1 Geração do módulo GTX

Quando se gera o módulo GTX através da interface disponibilizada pelo *software* VIVADO existem algumas decisões que necessitam de ser tomadas para além das que já foram mencionados. Essas passam de seguida a ser detalhadas:

Do lado do transmissor tomaram-se as seguintes principais decisões:

a. **Tamanho interno dos dados:** Esta escolha envolve o número de caminhos de dados (*datapath*) que são utilizados e ainda o valor da

frequência de TXUSRCLK. Escolheu-se 40 bits para tal, o que implica o uso de apenas um *datapath*, obtendo-se as frequências idênticas de TXUSRCLK e TXUSRCLK2.

b. **Tipo de codificação:** Neste caso não se escolheu codificação porque não é possível (a interface com a FPGA é de 40 bits) e também numa fase inicial optou-se por simplificar o projeto.

c. **Escolha ente *Buffer* ou Bloco de Alinhamento de Fase:** Foi escolhido a utilização do *buffer* uma vez que é de mais fácil utilização não requerendo o uso de lógica extra (comparativamente ao bloco de alinhamento de fase) e ainda assim é robusto.

Do lado do recetor as principais decisões tomadas foram as seguintes:

a. **Tipo de equalização:** Apesar de este ser um projeto simples em que não se pretende inserir o sinal em canais ruidosos, optou-se por utilizar um equalizador DFE uma vez que traz mais vantagens do que a utilização do equalizador LPM.

b. **Alinhamento de palavras:** Como palavra de alinhamento escolheu-se o símbolo K28.3 (mais informação pode ser encontrada em (**?**)). Contudo, é necessário ter em conta que não se está a utilizar codificação e, como tal, para efetuar o alinhamento da palavra o recetor não alinha pelo símbolo K28.3 codificado, mas sim, não codificado. Ou seja, na realidade quando encontrar a palavra "7C" assume que é a palavra de alinhamento e a trama passa a estar alinhada para esse limite. Para além disso, optou-se por ativar a porta "RXSLIDE" que ativa o alinhamento manual dos bits, pois é de esperar que seja necessário ativar o alinhamento manual para ligações cuja taxa de débito seja superior a 5 Gbit/s.

c. **Tipo de descodificação:** Não foi utilizada nenhuma descodificação, pois do lado do transmissor também não há codificação.

d. **Escolha ente *Buffer* ou Bloco de Alinhamento de Fase:** Optou-se pela escolha do *buffer*, porque no caso do recetor para além não requerer lógica extra e ter uma inicialização mais rápida (comparativamente ao bloco de alinhamento de fase), também não exige que a correção do sinal de relógio seja realizada fora do transcetor, tal como indicado em (**?**).

Na tabela 1 é apresentada uma tabela sumário do módulo gerado. Esta tabela foi adaptada da interface que gera o módulo.

Tabela 1: Sumário do módulo GTX gerado para a transmissão em série de uma barra de cores gerada na FPGA (adaptada do *software*)

Característica	GT
TX Line Rate (Gbit/s)	5,94
TX Reference Clock (MHz)	148,5
Enconding	None
TX Internal Data Width	40
TX External Data Width	40
TXUSRCLK (MHz)	148,5
TXUSRCLK2 (MHz)	148,5
TX Buffer Enabled	TRUE
RX Line Rate (Gbit/s)	5,94
RX Reference Clock (MHz)	148,5
Decoding	None
RX Internal Data Width	40
RX External Data Width	40
RXUSRCLK (MHz)	148,5
RXUSRCLK2 (MHz)	148,5
RX Buffer Enabled	TRUE

4.2 Arquitetura D

Na figura **??** visualiza-se o diagrama de blocos desenvolvido para esta arquitetura com os seus diversos sub-módulos.

Bloco gerador de uma Barra de Cores

Este bloco gerador de barra de cores tem a seguinte funcionalidade: gera uma barra de cores em *FULL HD* com uma frequência de 148,5 MHz. Nas suas entradas encontram-se as portas vindas diretamente do exterior, tal como nas outras arquiteturas que utilizam este bloco. Estas são os botões definidos pelo utilizador (*reset* e *start*) e também um sinal de relógio diferencial de 200 MHz. Nas suas saídas encontram-se os sinais referentes à imagem gerada que são posteriormente enviados para um bloco sincronizador dos dados HDMI.

Bloco Sincronizador dos dados HDMI

O recurso a este bloco sincronizador dos dados HDMI deve-se aos diferentes domínios de relógio que existem no sistema: por um lado existe um bloco que gera uma barra de cores a uma determinada cadência, e por outro existe um bloco que gera as tramas a serem enviadas para o transcetor a outra cadência. Estes dois sinais de relógio podem ser iguais, no entanto

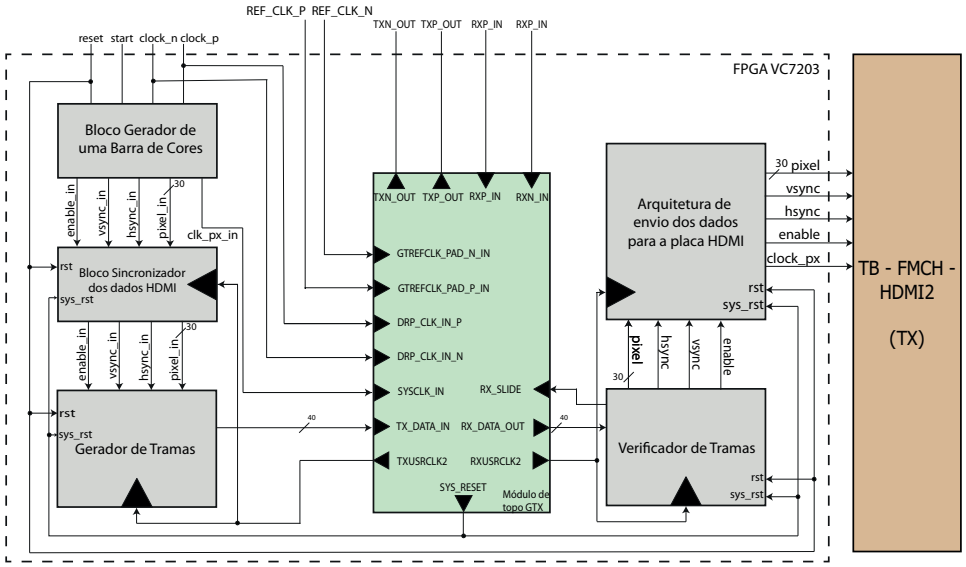

Figura 2: Diagrama de blocos da arquitetura D

podem não estar em fase o que é suficiente para haver problemas de meta-estabilidade.

Por este motivo, este bloco de sincronização é apenas constituído por dois registos de deslocamento (*shift-registers*) para que problemas de sincronização que possam eventualmente existir sejam resolvidos. O diagrama de blocos deste módulo é apresentado na figura 3.

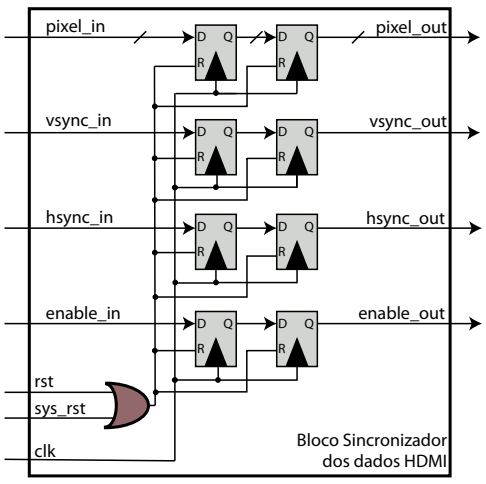

Figura 3: Bloco de sincronização de dados

Tal como se visualiza na figura, para além de nas suas entradas se en-

contrar os dados provenientes da fonte HDMI há também dois sinais : *rst* e *sys_rst*. O primeiro sinal refere-se ao sinal de *reset* controlado pelo utilizador e o segundo corresponde a um sinal de *reset* ativado automaticamente pelo módulo GTX.

O sinal de relógio a que opera este bloco trata-se do sinal de relógio proveniente do módulo GTX, tal como se visualiza na figura 2.

Gerador de Tramas

Este bloco é responsável pela criação das tramas a serem enviadas para o módulo GTX. Relembra-se que nesta abordagem inicial do projeto não se teve em conta a criação de tramas bem definidas para todos os momentos de transmissão.

Definiu-se a trama correspondente ao início da transmissão para que seja possível manter o alinhamento do lado do recetor e iniciar o processo. Esta é designada por SOP (*Start of Packet*) e é enviada em momentos de transmissão nulos, ou seja, quando os sinais de controlo da imagem se encontram inativos ($vsync = 0$, $hsync = 0$ e $enable = 0$), sendo constituída por 40 bits que em hexadecimal corresponde a: 000605047c. Este padrão sugerido pelo exemplo disponibilizado no *software* da *Xilinx* possui as seguintes particularidades:

a. No final da mesma encontra-se a palavra de alinhamento (conjunto dos primeiros bits a ser transmitido) que permite ao transcetor alinhar a trama internamente.

b. Nunca será confundida como trama de informação (apresentada na figura 4) devido à composição dos últimos 7 bits de ambas. A trama SOP termina em 1111100, e a trama de informação termina em 1010101.

Quando algum sinal de controlo referente à imagem está ativo é transmitida uma trama de informação com o formato que se visualiza na figura 4.

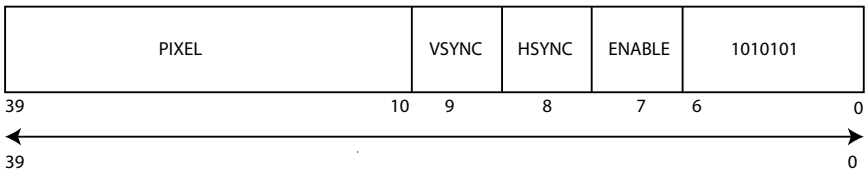

Figura 4: Estrutura das tramas de informação

Assim sendo, consoante os dados que recebe nas suas entradas (*pixel*, *vsync*, *hsync* e *enable*), este bloco ou envia SOP ou envia tramas de informação. A figura 5 exemplifica os diversos momentos de transmissão das

diferentes tramas numa imagem. A cinza correspondem os momentos da imagem em que os sinais de controlo da mesma estão inativos, e por isso são transmitidas tramas SOP. A castanho correspondem os momentos de transmissão em que algum dos sinais de controlo de imagem está ativo e por isso são enviadas tramas de informação. Sempre que o sinal *rst* ou *sys_rst* estiver ativo os dados são repostos e as tramas enviadas são SOP.

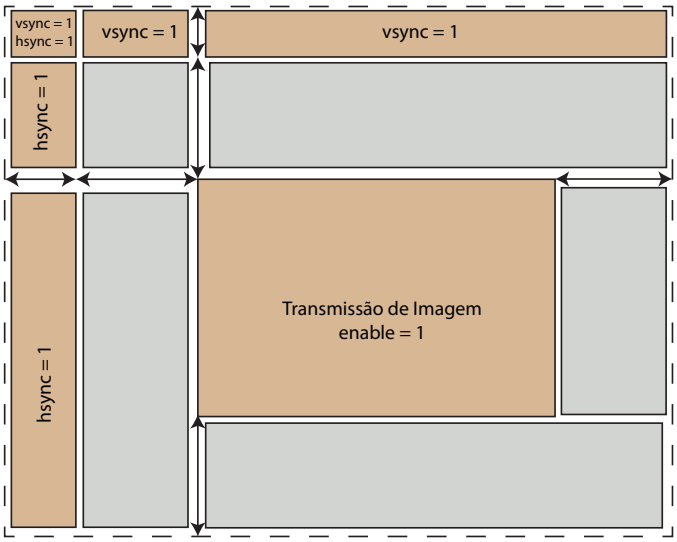

Figura 5: Momentos de transmissão de diferentes tramas

É de esperar que a transmissão demore algum tempo a alinhar do lado do recetor, sendo necessário que as tramas SOP sejam enviadas durante algum tempo. Por este motivo escolheram-se os momentos de transição de dados de controlo nulos, pois são suficientemente longos para que a transmissão possa ser alinhada. Assim, no pior dos casos, perde-se a primeira imagem completa que é transmitida, pois não são recebidos os primeiros dados de controlo (*vsync* e *hsync*) mas garante-se que todas as outras estão alinhadas e os dados serão corretamente recebidos do lado do transmissor.

Verificador de Tramas

Este bloco é responsável pela re-organização das tramas recebidas que são recebidas à cadência RXUSRCLK2. Apesar de poder parecer que com o alinhamento interno as tramas chegariam ao recetor tal como são enviadas do transmissor, isso raramente se verifica. As tramas são alinhadas para um determinado limite assim que o recetor encontra a palavra de alinhamento.

Segundo (**?**) e (**?**), quando a palavra de alinhamento é detetada no recetor, este assume que os dados a partir daí são válidos e alinha-os para um determinado limite. Este limite pode ser selecionado aquando a

criação do módulo GTX, tendo-se optado por alinhar os dados para o *byte* mais próximo. Apesar de não se estar a lidar com codificação e devido ao tamanho do *datapath* ser 40 bits, o bloco de alinhamento das palavras considera que 1 *byte* são 10 bits (a codificação 8B/10B codifica 8 bits em 10 bits). Assim, quando é detetada a palavra de alinhamento as tramas recebidas no recetor podem chegar de quatro maneiras diferentes ilustradas na figura 6.

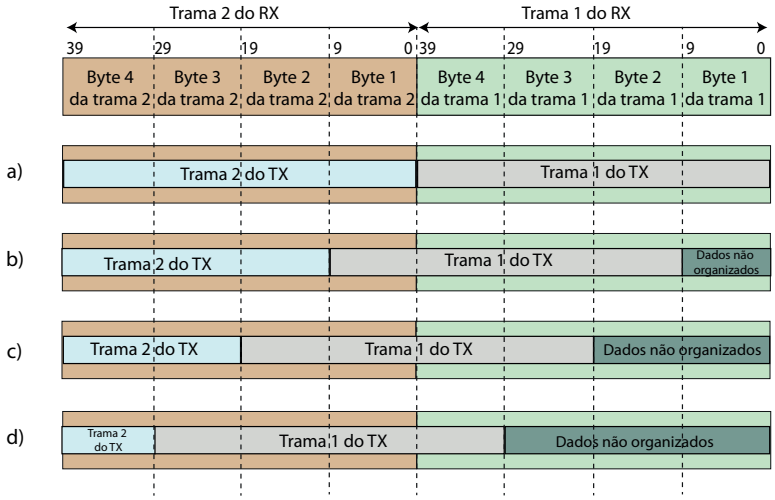

Figura 6: Alinhamento das tramas no transcetor

Daí a importância da trama de SOP: quando é detetada em algum dos limites assinalados a cinzento dá-se início à transmissão dos restantes dados. Para se proceder à organização dos dados recebidos é necessário recorrer a uma máquina de estados. Esta foi adaptada da que a ferramenta de *software* cria por omissão quando é gerado um exemplo deste módulo GTX.

A máquina de estados principal é dividida em três por questões de simplificação e de melhor entendimento da mesma. Cada uma destas divisões passa a ser detalhada, sem nunca esquecer que funcionam como um todo.

a. **Máquina de estados geral da receção de dados:**

   Esta máquina é composta por três estados que definem o estado geral da receção de dados. Estes estados são:

   - ***Begin:*** Estado inicial que aguarda a receção de dados válidos do recetor. Quando a trama de SOP é detetada pela primeira vez, então a máquina transita para um estado em que recebe dados válidos.
   - ***Track_data:*** Neste estado já foi detetato o início da transmissão (SOP) e por isso todos os dados que estão a ser recebidos são con-

siderados válidos. Quando é detetado algum erro (com a ajuda de uma máquina de estados detetora de erros) transita de estado.

- ***Data_error_detected:*** Estado de deteção de erro que transita imediatamente para o estado inicial para aguardar a chegada de dados válidos novamente.

Sempre que for necessário fazer *reset* ao sistema, quer pela ativação do utilizador ou então pelo transcetor, volta-se ao estado inicial "*begin*".

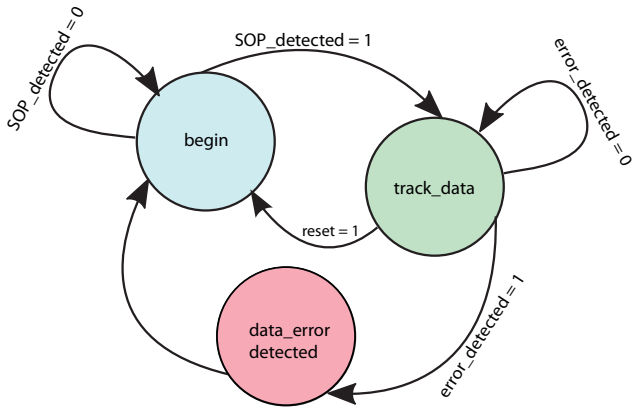

Figura 7: Máquina de estados de transmissão

A figura 7 representa a máquina de estados detalhada anteriormente. De notar que na imagem estão representadas algumas *flags* de decisão de transição de estados. A *flag* "*SOP_detected*" indica que se a trama de início de pacote foi encontrada nos dados recebidos e "*error_detected*" indica que foi detetado um erro nos dados recebidos. Ambas são definidas por máquinas de estados que serão explicadas mais adiante.

b. **Máquina de leitura de dados**

Esta máquina de estados é responsável pela constante procura de dados recebidos pelo recetor. Destes apenas são guardadas as últimas duas tramas porque da sua combinação resulta a indicação de início de transmissão, iniciando o funcionamento desta máquina de estados.

A figura 8 ilustra a máquina de estados aqui referida, contudo antes de se detalhar os diferentes estados da mesma, faz-se uma breve descrição das *flags* de transição de estados presentes na figura:

- *all_incoming_data*: Esta *flag* indica se todos os bits presentes nas duas últimas tramas recebidas no recetor são 0 ou não. Se sim, então a *flag* é igual zero, senão é um.

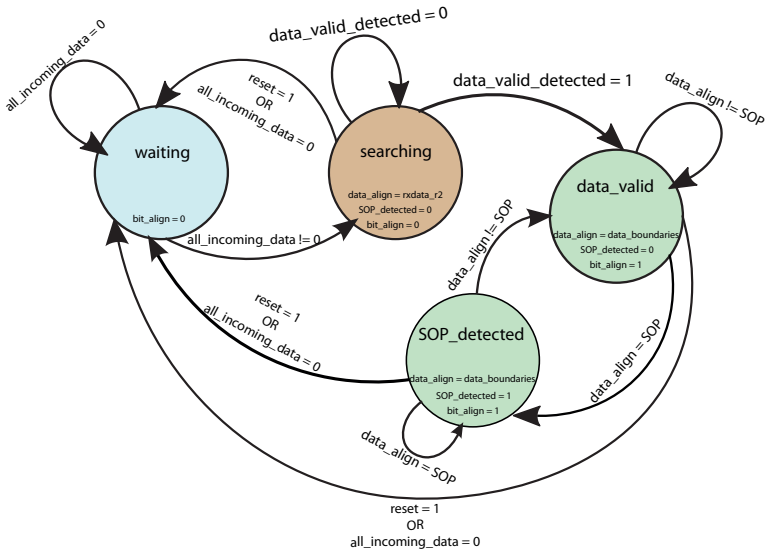

Figura 8: Máquina de estados leitura de dados

- *data_valid_detected*: Esta *flag* indica se foi encontrado nos dados recebidos (nas duas últimas tramas) a trama SOP em alguma das situações assinaladas a cinza presentes na figura 6.
- *reset*: indica se foi ativado o sinal para repôr os dados originais da máquina de estados.

Estas são as principais *flags* de transição de estados da máquina. De seguida serão detalhados todos os estados da mesma:

- ***Waiting*:** Este estado indica que o recetor ainda não está a receber dados, isto porque a *flag* *all_incoming_data* está inativa. Por defeito, quando não há transmissão de dados do lado do transmissor, o recetor recebe apenas dados iguais a zero. Assim que esta *flag* se ativa então passa-se para o estado *searching*.
- ***Searching*:** Neste estado, a máquina procura a trama que dá inicio à transmissão de dados (SOP). Uma vez que ela pode vir alinhada em diferentes *bytes* nas tramas do recetor, então a máquina procura a SOP nas quatro situações apresentadas na figura 6. Assim que encontra SOP, memoriza os limites das tramas em que esta foi encontrada para que se possa retirar todos os outros dados de transmissão e transita de estado.
- ***Data_valid*:** Este estado serve essencialmente para guardar os dados devidamente alinhados segundo os limites da trama SOP. Ativa-se a flag *bit_align* para que o sistema reconheça que os dados

se encontram alinhados. Se os dados que estão a ser guardados em *data_align* corresponderem à trama SOP então transita-se de estado para indicar que esta trama foi detetada.

- ***SOP_detected*:** Quando este estado está ativo, então os dados alinhados correspondem à trama SOP, e por isso deve-se ativar a *flag* "*SOP_detected*". Este estado mantém-se ativo enquanto os dados alinhados corresponderem à trama SOP e transita para o estado "*data_valid*" assim que tal deixar de ser verdade.

Sempre que a transmissão for interrompida (*all_incoming_data* se iguala a zero) ou o sinal de *reset* for ativo (quer por opção do utilizador ou pelo módulo GTX), então a máquina retorna ao estado inicial e repõe os dados originais do sistema.

Inicialmente o estado ativo é "*waiting*", e todas as *flags* de decisão de mudança de estado (*all_incoming_data* e *data_valid_detected*) estão igualadas a zero.

c. **Máquina de estados de verificação do alinhamento dos dados**

Tal como já foi mencionado anteriormente, para taxas de transmissão superiores a 5 Gbit/s , o transcetor pode alinhar falsamente os dados. Por esse motivo, as tramas recebidas podem não vir dentro dos limites apresentados na figura 6, sendo assim necessário o recurso a uma máquina de estados que valide o correto alinhamento.

A máquina de estados apresentada na figura 9 é responsável pela deteção do devido alinhamento das tramas recebidas de acordo com a figura 6. Caso tal não se verifique procede ao alinhamento manual das tramas.

Antes de se passar à descrição de cada estado, é de notar que existem algumas *flags* de decisão de transição de estados:

- *bit_align*: Esta *flag* indica se a palavra se encontra devidamente alinhada, tal como ilustrada na figura 6.
- *all_incoming_data*: Esta *flag* já foi mencionada e detalhada na máquina de estado que faz a leitura de dados.
- *count_slip_complete*: Esta é uma *flag* que fica ativa quando um contador termina (será explicado mais à frente).
- *reset*: Indica a reposição dos dados originais da máquina de estados quando ativa.

Esta máquina é composta por três estados, sendo estes:

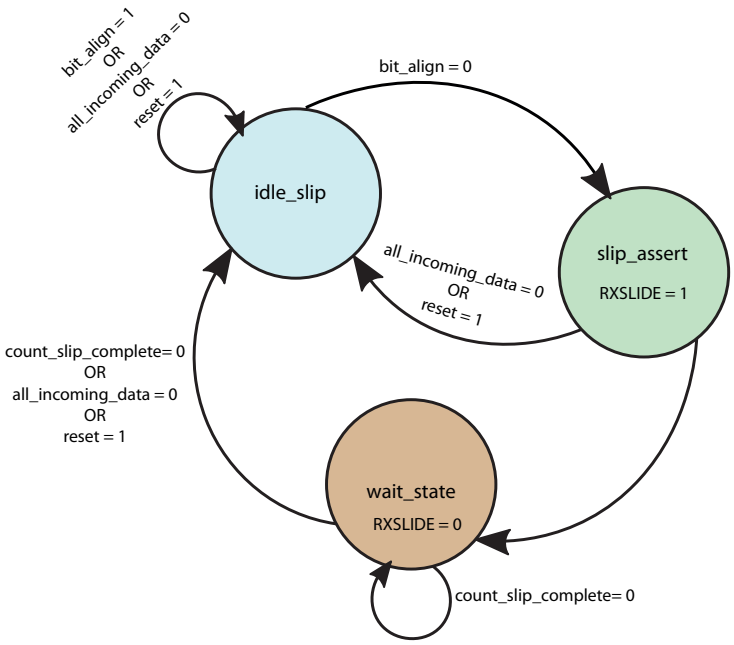

Figura 9: Máquina de estados de verificação do alinhamento dos dados

- ***Idle_slip*:** Este estado aguarda pela receção de dados e também pela deteção do desalinhamento da palavra. Há transição de estado quando se verifica que as tramas que chegam ao recetor não estão alinhadas de acordo com os limites que deveriam estar.
- ***Slip_assert*:** Este estado está ativo quando há deteção de falso alinhamento de palavras, e por isso a saída que se conecta ao GTX é ativada: RXSLIDE. A ativação deste sinal vai permitir que as tramas sejam deslocadas em 1 bit. Este estado transita imediatamente para o estado de *Wait_state*.
- ***Wait_state*:** Neste estado o sinal de saída RXSLIDE volta a estar inativo e aguarda-se que a operação de deslocamento de 1 bit se verifique. Como estamos perante um *datapath* de 40 bits, segundo (**?**) aguarda-se 64 ciclos de relógio. Após esta espera, a flag *count_slip_complete* fica ativa e há transição de estado.

De notar que sempre que houver falha de transmissão ou o sinal de *reset* for ativo, os sinais são repostos para os originais e volta-se ao estado *Idle_slip*.

Inicialmente o estado ativo é o *Idle_slip* e as *flags* de sinalização de transição de estado são igualadas a 0.

Para além destas máquinas de estados apresentadas existe a possibilidade de implementação de uma outra que verifique os erros das tramas recebidas. Esta máquina torna-se útil quando há implementação de códigos detetores de erros nas tramas, ativando a *flag* "*error_detected*", sempre que o *checksum* recebido na trama não corresponder à mensagem. Contudo, nesta abordagem inicial tal não é realizado e por isso essa *flag* é globalmente igualada a zero.

Quando os dados estão alinhados procede-se à extração da informação contida na trama recebida, enviando-a para o bloco responsável pela sua transmissão para a placa HDMI, sendo necessário que o estado *track_data* esteja ativo. A informação retirada depende do tipo de trama recebida: caso a trama seja SOP, os dados são todos igualados a 0; caso contrário, então extrai-se os dados de acordo com o formato da trama de informação apresentado na figura 4.

Bloco de envio de dados para a placa HDMI

Este bloco é responsável pela receção dos dados alinhados provenientes do bloco verificador de tramas e do seu posterior envio para a placa HDMI transmissora. A figura 10 é apresentado o diagrama de blocos deste módulo.

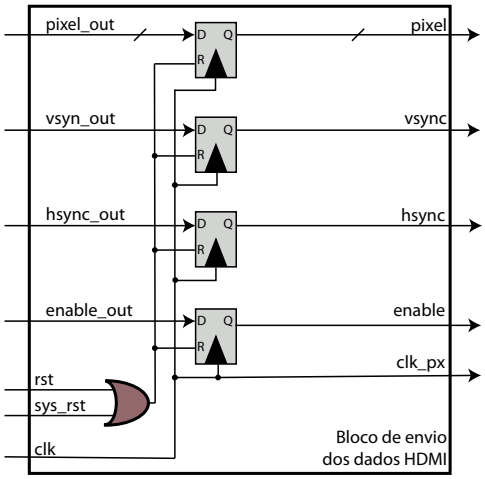

Figura 10: Diagrama de blocos de envio de dados para a placa HDMI transmissora

Os dados recebidos neste bloco são lidos para um registo ao flanco positivo do sinal de relógio que alimenta este módulo (RXUSRCLK2). Posteriormente, todos os dados presentes nos registos são enviados para a placa HDMI transmissora incluindo o sinal de relógio RXUSRCLK2.

Localizações das portas de saída do módulo de topo

Verifica-se que este bloco tem diversas entradas e saídas, e por isso para além de todo o código desenvolvido em Verilog para estes sub-módulo é necessário definir as localizações físicas da FPGA para cada porta de entrada e saída, estando as mesmas detalhadas na tabela 2.

Tabela 2: Localizações físicas das portas de entrada e saída da arquitetura desenvolvida

	Sinal	LOC na FPGA	Banco na FPGA
Entrada	clk_p	E19	38
Entrada	clk_n	E18	38
Entrada	reset	N41	19
Entrada	start	E42	19
Entrada	REF_CLK_P	AF8	114
Entrada	REF_CLK_N	AF7	114
Entrada	RXP_IN	AG6	114
Entrada	RXN_IN	AG5	114
Saída	TXN_OUT	AK3	114
Saída	TXP_OUT	AK4	114
Saída	clk_px	E34	35
Saída	enable	K35	34
Saída	hsync	M32	34
Saída	vsync	L31	34
Saída	pixel[0] a pixel [29]	Ver (**?**)	34 e 35

4.3 Arquitetura E

A figura 11 representa o digrama de blocos da arquitetura E.

Através da visualização do diagrama blocos simplificado apresentado na figura 11 é possível concluir que o único sub-módulo adicionado é o bloco recetor de dados HDMI. Todos os outros sub-módulos são exatamente iguais ao da arquitetura anterior, estando detalhados na subsecção 4.2. Assim sendo, apenas será apresentado nesta subsecção esse mesmo bloco.

5 Configuração do *setup*

Na **??** está representado o *setup* de teste que deve ser utilizado para esta arquitetura. A configuração de cada um dos componentes passa a ser de seguida descrito.

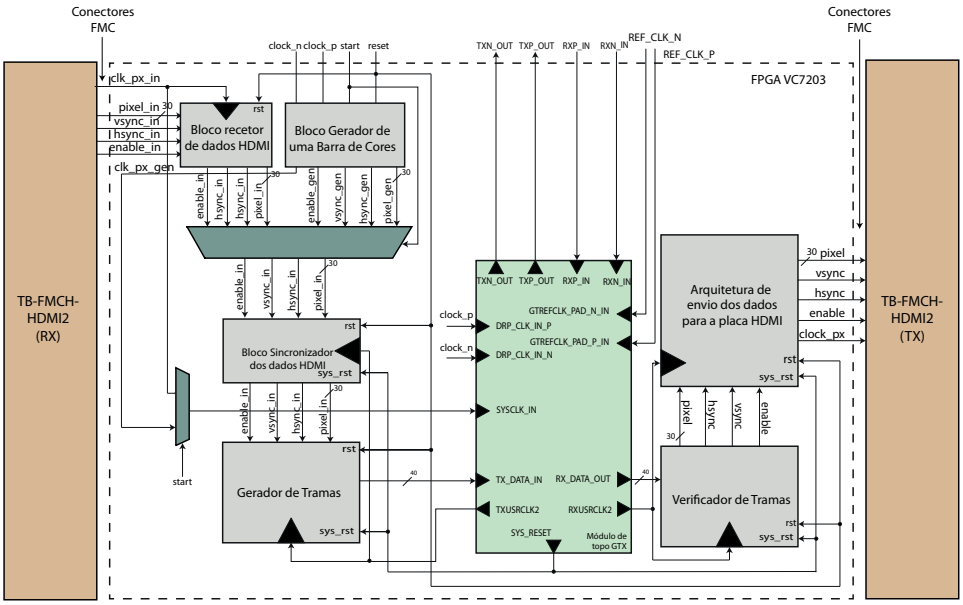

Figura 11: Diagrama de blocos da arquitetura E

5.1 TB-FMCH-HDMI TX

Estas placas devem estar configuradas por omissão. Para tal deve-se recorrer ao *software iMPACT* e ainda ao programador JTAG (tal como indicado em (3)):

- O programador deve estar devidamente conectado à placa HDMI e ao computador;
- O *software iMPACT* deve ser inicializado;
- Com o botão do lado direito do rato deve ser escolhida a opção "Inicializar cadeia";
- A PROM (*Programmable read-only memory*) cujos detalhes podem ser encontrados em (4) da placa transmissora deve ser programada com o ficheiro com o nome "*tx_fpga_top.mcs*" que se encontra na pasta "*finalFolder/HDMI_programming/default*".

5.2 FPGA VC7203

A placa FPGA *Virtex-7* deve ser programada com o *bitstream* gerado pós-implementação da arquitetura no *software* VIVADO.

- A FPGA deve estar devidamente conectada ao computador através da porta USB;

- Após inicializado, deve ser escolhida a opção "*Open Hardware Manager*" no VIVADO;
- De seguida deve ser escolhida a opção "*Open New Target*" dentro de "*Open Target*";
- Na nova janela aberta deve-se seleccionar a FPGA **XC7VX485T-3** e terminar até a janela fechar;
- De seguida, selecciona-se com o botão do lado direito do rato o componente **XC7VX485T-3**, escolhendo-se a opção "*Program Device*";
- Nessa janela é seleccionado o bitstream que tem o nome "*colorBarGenerator.bit*" na pasta '*finalFolder/planA*"

Mais informações sobre o *software* VIVADO pode ser encontrado em (5).

6 Utilização dos recursos da FPGA

Na Tabela 3 são apresentados os valores dos recursos utilizados pela FPGA nesta arquitetura (denominada por "A").

Tabela 3: Recursos utilizados pela arquitetura na FPGA

Recurso	**Arquitetura A**	
	Utilização	**%**
FF	31	0,01
LUT	59	0,02
I/O	38	5,43
GT	0	0

Os valores percentuais ocupados pela arquitetura são muito baixos, possibilitando a expansão da mesma caso necessário.

Referências

[1] Xilinx and Inc, *VC7203 Virtex-7 FPGA GTX Transceiver Characterization Board User Guide*, 1.3 ed., outubro 2014.

[2] Inrevium, *Manual do Utilizador de TB-FMCH-HDMI2 Hardware*, 1.04 ed., agosto 2014.

[3] Xilinx, *Xilinx Platform Cable USB II.*

[4] Xilinx, *Platform Flash In-System Programmable Configuration PROMs*, 2.19 ed., junho 2016.

[5] Xilinx, *Vivado Design Suite Tutorial*, 2015.1 ed., maio 2015.