TRABALHO 2: MODELAGEM DE AMBIENTE COM MÚLTIPLOS CLOCKS

Trabalho idealizado pelo Prof. Dr. Rafael Garibotti

DEADLINE: 17/11/2023

1. OBJETIVO

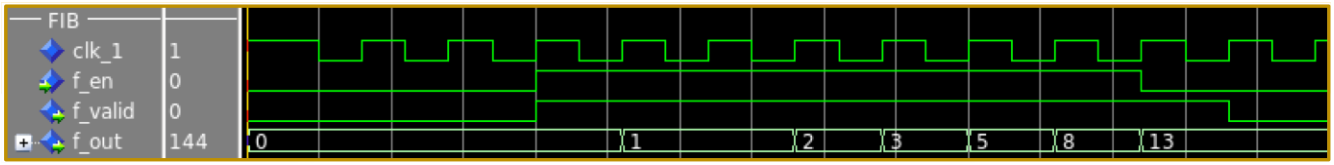
O **objetivo** desse trabalho é de simular um ambiente que contenha múltiplos clocks. Com esse propósito, deverá ser projetado um circuito que opera internamente com 2 clocks diferentes e trocam informações nesses dois domínios (simulando um ***circuito produtor-consumidor***). Sua tarefa é desenvolver os módulos internos e conectá-los como indicado na descrição deles, fazendo a prototipação do circuito em FPGA. Esse trabalho deverá ser realizado de forma **individual** ou **em dupla** na linguagem de sua preferência.

ESPECIFICAÇÃO DOS MÓDULO DO CIRCUITO

1.1. Fibonacci

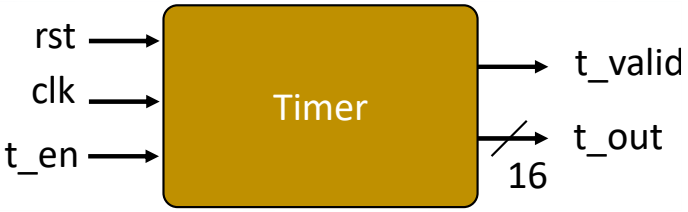
O módulo **Fibonacci** deve gerar os números inteiros da sequência de Fibonacci quando o sinal *enable* do módulo (**f_en**) estiver ativo (‘1’), isto é, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, etc. Os sinais deste módulo são mostrados abaixo e indicam: **rst**, sinal de reset do módulo que é ativo alto (‘1’); **clk**, clock deste módulo síncrono que nesta implementação usará o *clock rápido* de 10 Hz; **f_en**, sinal *enable* do módulo que indica quando produzir o dado de saída; **f_valid**, sin al que indica que o valor colocado no sinal de saída é válido; e **f_out**, sinal que contém o valor corrente da sequência de Fibonacci. Note que o sinal **f_out** tem um tamanho de 16 bits, logo, caso o valor produzido estoure essa representação numérica, você ***não precisa se preocupar*** com o valor informado, isto é, se ele está correto ou não.

A forma de onda abaixo mostra um exemplo de funcionamento do módulo **Fibonacci**.

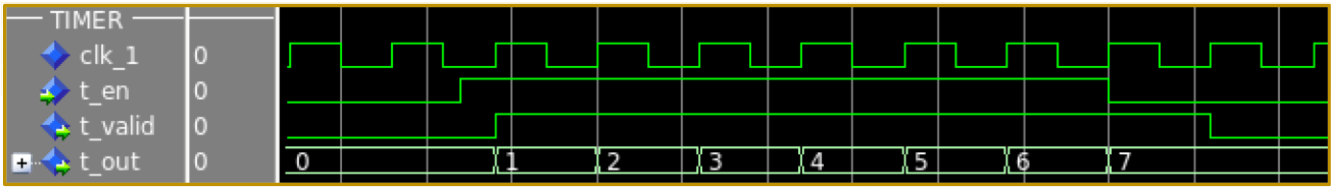


1.2. Timer

O módulo **Timer** deve gerar números positivos de forma crescente, isto é, começando no 1, seguido do 2, do 3, do 4, e assim por adiante. Os sinais deste módulo são mostrados abaixo e indicam: **rst**, sinal de reset do módulo que é ativo alto ('1'); **clk**, clock deste módulo síncrono que nesta implementação usará o *clock rápido* de 10 Hz; **t_en**, sinal *enable* do módulo que indica quando produzir o dado de saída; **t_valid**, sinal que indica que o valor colocado no sinal de saída é válido; e **t_out**, sinal que contém o valor positivo gerado. Note que o sinal **t_out** tem um tamanho de 16 bits, logo, caso o valor produzido estoure essa representação numérica, você deve apresentar ***sempre os 16 bits menos significativos***.



A forma de onda abaixo mostra um exemplo de funcionamento do módulo **Timer**.



1.3. Digital Clock Manager (DCM)

O módulo **Digital Clock Manager** (**DCM**) deve gerar dois clocks a partir de um clock de referência. Note que o clock de referência é o fornecido pelo FPGA, neste caso, ele deve operar a 100 MHz. Outra informação importante é que os clocks gerados devem estar em fase com o clock de referência. Os sinais deste módulo são mostrados abaixo e indicam: **rst**, sinal de reset do módulo que é ativo alto ('1'); **clk**, clock de referência deste módulo síncrono que opera a 100 MHz; **prog_in**, é o sinal de programação de entrada de 3 bits que indica qual frequência o *clock lento* deve operar (abaixo segue uma tabela de conversão); **update**, é o sinal que indica que o *clock lento* deve ser atualizado para a frequência indicada pelo sinal *prog_in*; **clk_1**, é o *clock rápido* gerado, que opera a 10 Hz; **clk_2**, é o *clock lento* gerado, que pode operar entre 10Hz e 78.125 mHz; e **prog_out**, é o sinal de programação de saída de 3 bits que indica qual frequência o *clock lento* está gerando naquele momento.

Abaixo é mostrado a Figura com as portas de entrada e saída, além das Tabelas de conversão dos sinais de programação com os períodos dos clocks.

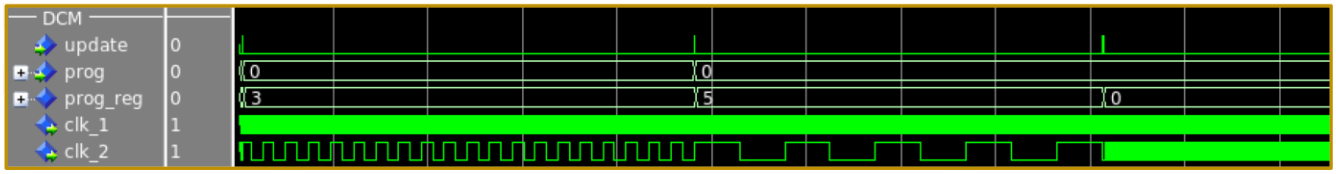
Clock rápido (**clk_1**):

time
0.1 s

Clock lento (**clk_2**):

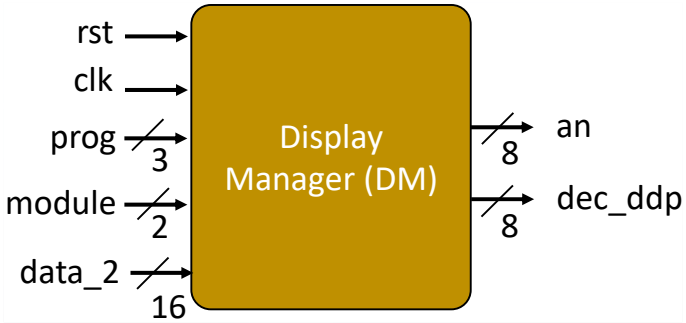
mode	Time
0	0.1s
1	0.2s
2	0.4s
3	1.0s
4	1.6s
5	3.2s
6	6.4s
7	12.8s

A forma de onda abaixo mostra um exemplo de funcionamento do módulo **DCM**. Note que nela, o sinal **prog** representa o **prog_in** e o sinal **prog_reg** representa o **prog_out**.

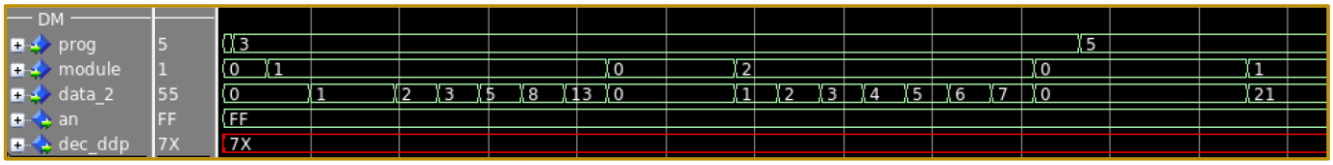


1.4. Display Manager (DM)

O módulo **Display Manager** (**DM**) controla as saídas dos displays de 7 segmentos. Os sinais deste módulo são mostrados abaixo e indicam: **rst**, sinal de reset do módulo que é ativo alto ('1'); **clk**, clock de referência deste módulo síncrono que opera a 100 MHz; **prog**, sinal de programação de 3 bits que indica qual frequência o *clock lento* está operando; **module**, é um sinal de 2 bits que indica se o sistema está consumindo algum dado produzido, e se sim, de qual módulo se esta consumindo (isto é, módulo *Fibonacci* ou módulo *Timer*); **data_2**, é o valor de 16 bits que o sistema está consumindo naquele momento, o qual foi previamente gerado pelo módulo *Fibonacci* ou módulo *Timer*; **an**, sinal ativo baixo do ânodo que contém 8 bits, representando cada um dos displays disponíveis no FPGA; e **dec_ddp**, é um sinal de 8 bits contendo o valor decodificado do dígito a ser mostrado no instante atual pelo display. Note que os valores decodificados dos dígitos estarão distribuídos da seguinte maneira: display #8 contém o valor de **prog**; display #6 contém o valor de **module**; e displays de #4, #3, #2 e #1 contém o valor de data_2, sendo o display #4 os bits mais significativos e o display #1 os bits menos significativos.



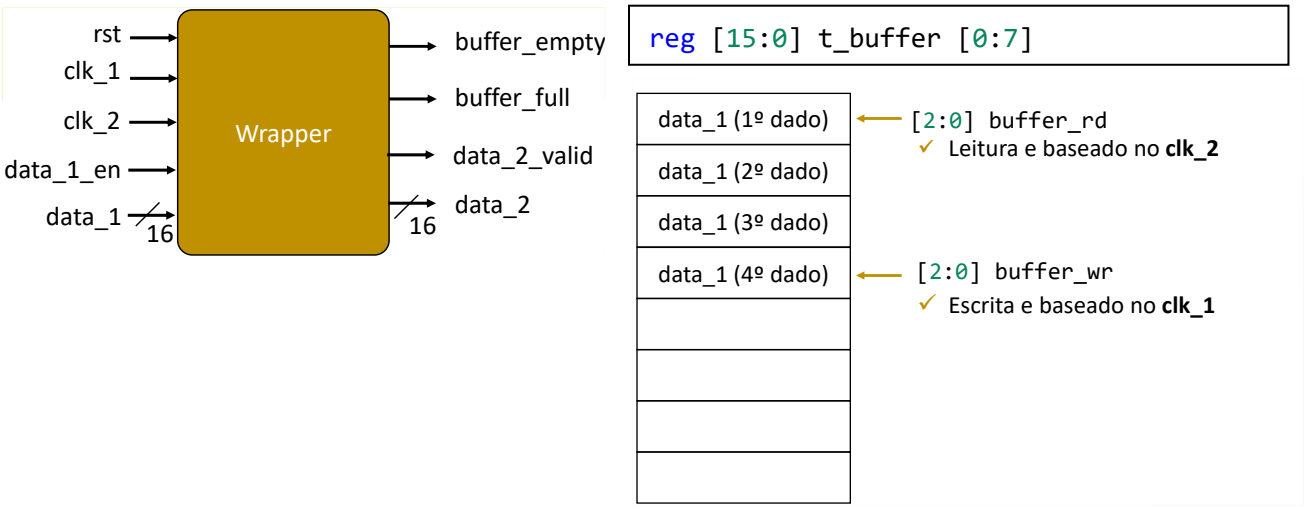
A forma de onda abaixo mostra um exemplo de funcionamento do módulo **DM**.



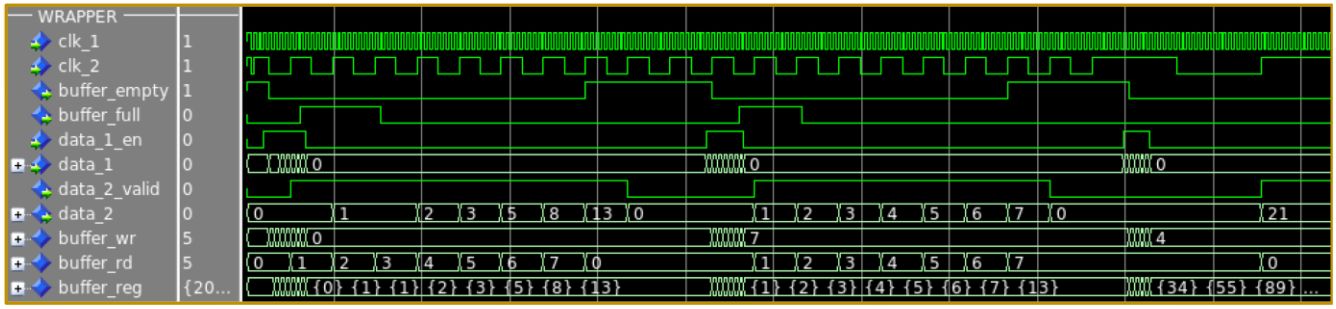
1.5. Wrapper

O módulo **Wrapper** deve gerenciar um ***buffer*** onde o dado é produzido em uma frequência (isto é, escrito) e consumido em outra frequência (isto é, lido). Como este ***buffer*** tem tamanho finito (isto é, 8 dados de 16 bits cada), o módulo **Wrapper** possui *flags* que indicam se o buffer está vazio ou cheio, os quais auxiliarão no controle do sistema. Os sinais deste módulo são mostrados abaixo e indicam: **rst**, sinal de reset do módulo que é ativo alto ('1'); **clk_1**, é o *clock rápido*, que opera a 10 Hz; **clk_2**, é o *clock lento*, que pode operar entre 10Hz e 78.125 mHz; **data_1_en**, sinal *enable* que indica que o dado de entrada é válido; **data_1**, é o valor de 16 bits que foi produzido pelo sistema, sendo este proveniente do módulo *Fibonacci* ou do módulo *Timer*; **buffer_empty**, *flag* de controle que indica que o ***buffer*** interno do módulo está vazio; **buffer_full**, *flag* de controle que indica que o ***buffer*** interno do módulo está cheio; **data_2_valid**, sinal que indica que o valor a ser consumido é válido; e **data_2**, é o valor de 16 bits que o sistema está consumindo naquele momento, o qual foi previamente gerado pelo módulo *Fibonacci* ou pelo módulo *Timer*.

Abaixo é mostrado a Figura com as portas de entrada e saída deste módulo, além de um exemplo de gerenciamento do ***buffer*** interno (chamado de **buffer_reg**). Note que neste exemplo foi gerado 2 sinais internos que indicam os ponteiros atuais de leitura (**buffer_rd**) e escrita (**buffer_wr**) de um ***buffer*** circular. Contudo, a implementação desse módulo é livre e vocês tem liberdade de propor outra arquitetura (mantendo as portas de entrada e saída do módulo inalteradas).

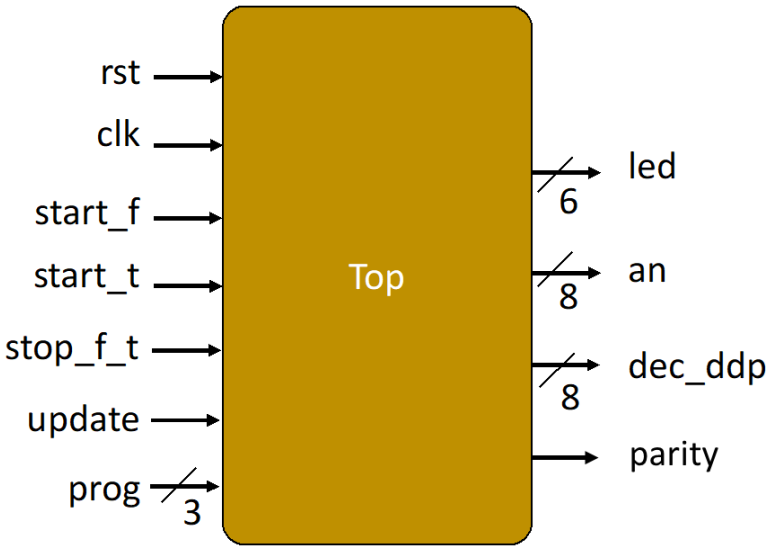


A forma de onda abaixo mostra um exemplo de funcionamento do módulo **Wrapper**.

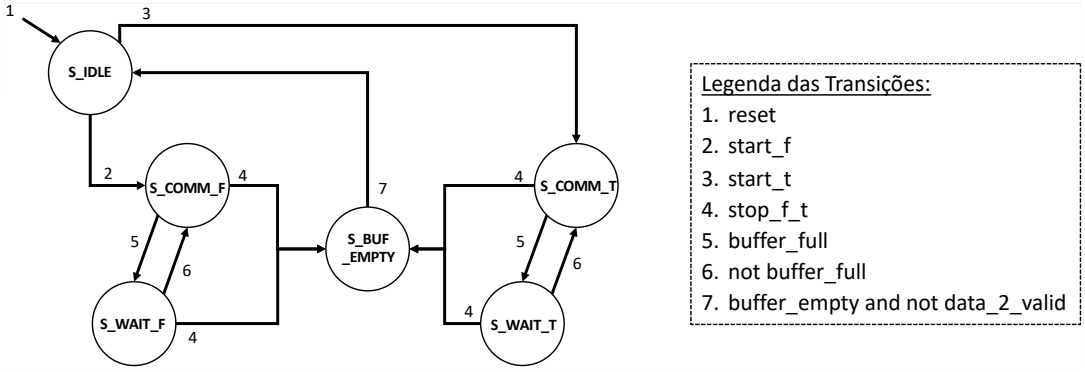


2. DESCRIÇÃO DO TOP

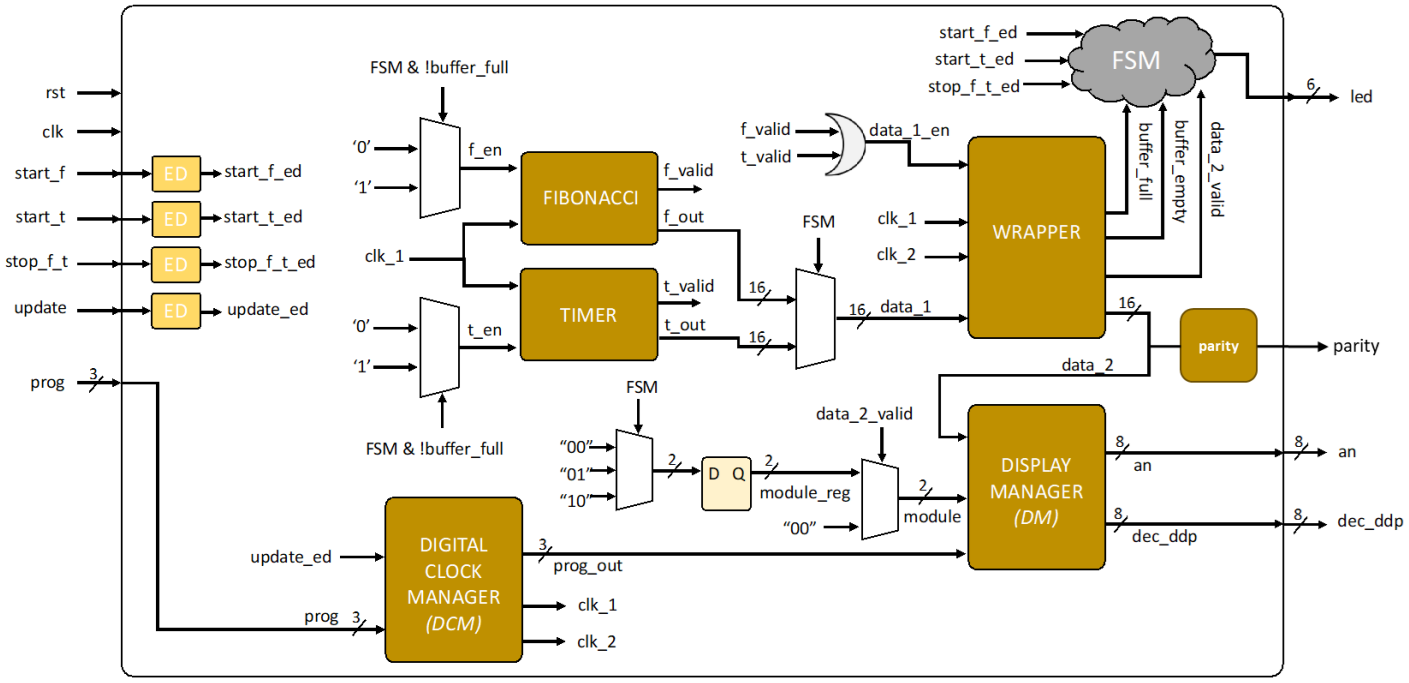
O módulo **Top** é o que descreve o ***circuito produtor-consumidor de múltiplos relógios***, ele irá instanciar os 5 módulos descritos anteriormente (**Fibonacci**, **Timer**, **DCM**, **DM** e **Wrapper**) e irá fazer a ligação com os pinos do FPGA. Note que para evitar ruídos nos sinais de entrada, é utilizado e instanciado também módulos ***Edge Detector*** (***ED***). Os sinais deste módulo são mostrados abaixo e indicam: **rst**, sinal de reset do módulo que é ativo alto ('1'); **clk**, clock de referência deste módulo síncrono que opera a 100 MHz; **start_f**, é o botão que indica o início ou continuação da produção de dados pelo módulo *Fibonacci*; **start_t**, é o botão que indica o início ou continuação da produção de dados pelo módulo *Timer***; stop_f_t**, é o botão que indica a parada de produção de dados dos módulos *Fibonacci* e *Timer*; **prog**, é o valor de programação de entrada de 3 bits que indica qual frequência o *clock lento* deve operar; **update**, é o botão que indica que o *clock lento* deve ser atualizado para a frequência indicada pelo valor *prog*; **led**, sinal visual que indica em que estado da máquina de estado o sistema está operando; **parity** é um bit de paridade gerado a partir da saída data do módulo *wrapper;* **an**, é o ânodo ativo baixo de 8 bits que controla a ativação de cada um dos displays disponíveis no FPGA; e **dec_ddp**, é o valor decodificado do dígito de 8 bits a ser mostrado no instante atual pelo display.



O módulo **Top** contém uma máquina de estados de 6 estados, descritos a seguir: **S_IDLE**, estado inicial em repouso; **S_COMM_F**, estado de produção e consumo dos dados do módulo *Fibonacci*; **S_WAIT_F**, estado onde a produção de dados do módulo *Fibonacci* é parada temporariamente pois o *buffer* está cheio; **S_COMM_T**, estado de produção e consumo dos dados do módulo *Timer*; **S_WAIT_T**, estado onde a produção de dados do módulo *Timer* é parada temporariamente pois o *buffer* está cheio; e **S_BUF_EMPTY**, estado de consumo e esvaziamento do *buffer*.

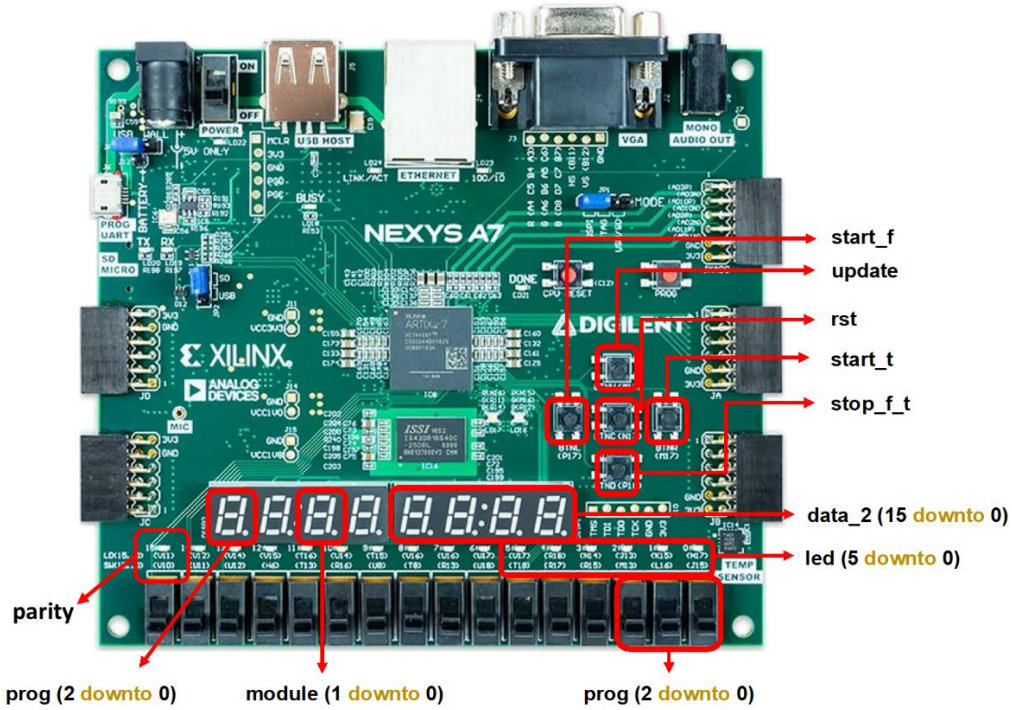


O desenvolvimento do módulo **Top** é livre. Contudo, abaixo é mostrado um diagrama de blocos de referência, que pode ajudar o aluno a desenvolver o módulo **Top**.



3. PROTOTIPAÇÃO EM FPGA

O ***circuito produtor-consumidor*** deverá, após validação em simulação, ser prototipado em FPGA. Neste sentido, a Figura abaixo mostra as ligações do módulo **Top** com o FPGA Nexys A7 da Xilinx.



MATERIAL DE APOIO

O material de apoio disponibiliza os protótipos dos arquivos a serem implementados (tanto em VHDL quanto em Verilog), além de arquivos de apoio, como o *template* .xdc, o *testbench* e os scripts para simulação. Esses são definidos abaixo:

- **fibonacci**, **timer**, **dcm**, **dm**, **wrapper**, **top**: esses são os protótipos dos arquivos a serem implementados;
- **edge_detector**, **edge_detector_sintese**: são módulos usados para filtrar os sinais provenientes dos botões do FPGA;
- **dspl_drv_NexysA7**: este é um módulo que implementa a interface de hardware necessária para acionar os displays de sete segmentos do FPGA;
- **Nexys-A7-100T-TP2.xdc**: este é o *template* do arquivo que mapeia os pinos do FPGA com os sinais de entrada e saída do módulo Top. Note que se deve descomentar as linhas necessárias e renomear os pinos usados;
- **tb**: esta é a versão inicial do *testbench* feito para testar o circuito produtor-consumidor. elos teus colegas da equipe de verificação. Note que basicamente é o arquivo vazio, ele foi feito inicialmente para testar os scripts. Logo, você deve incrementá-lo;
- **sim.do**: este é o script de simulação;
- **wave.do**: arquivo que cria as formas de onda padrões do circuito produtor-consumidor.

4. ATIVIDADES

1. Você deve implementar todos os módulos descritos nessa especificação, mostrando o seu funcionamento *primeiramente* em **simulação**.

2. Após a validação do circuito utilizando simulação, você deve **implementá-lo** na FPGA. **Devem ser entregues todos os arquivos (.v/.vhd), além dos scripts de simulação e também *bitstream* para o FPGA.**
   Atenção: Mantenha a pasta do projeto organizada e não entregue arquivos desnecessários. Lembre-se, você está se tornando um profissional, espera-se organização. Portanto, falta de organização é ***passível de descontos*** diretamente na nota final.

5. AVALIAÇÃO

- **[2 pontos]** – Apresentação do correto funcionamento da simulação.
- **[2 pontos]** – Apresentação do correto funcionamento da prototipação.
- **[6 pontos] –** Correta modelagem dos módulos:
  - **[1 ponto] –** Módulo Fibonacci
  - **[1 ponto] –** Timer
  - **[1 ponto] –** Display Manager - DM
  - **[1 ponto] –** Digital Clock Manager - DCM
  - **[1 ponto] –** Wrapper
  - **[1 ponto] –** Módulo Topo - Top