Laboratório de Sistemas Digitais

Trabalho Prático nº 10

Depuração e Validação de Circuitos em FPGA
Programação da Memória Flash do Kit para Armazenamento não Volátil da Configuração

Objetivos

- Depuração de circuitos digitais implementados em FPGA com base em ferramentas de visualização de sinais capturados em tempo-real.
- Familiarização com os mecanismos e dispositivos de armazenamento não volátil da configuração da FPGA.

Sumário

A primeira parte deste trabalho pretende introduzir o princípio de funcionamento e as capacidades dos analisadores lógicos integrados disponibilizados pelos fabricantes de FPGAs. Estes analisadores consistem num conjunto de blocos de hardware e ferramentas de software que permitem a captura de sinais em tempo-real e a sua visualização através de uma aplicação de software, possibilitando assim a depuração de sistemas implementados em FPGA. Esta apresentação é baseada num projeto de um contador binário que permitirá ilustrar todos os passos necessários para a utilização do analisador lógico.
Na segunda parte do guião é apresentada a sequência de passos necessários para que a configuração da FPGA seja armazenada na memória FLASH do kit DE2-115 (externa à FPGA). Isto permite que quando o kit DE2-115 é ligado, a FPGA seja automaticamente configurada a partir de uma memória não volátil existente no kit, dispensando o cabo de programação. Esta capacidade é fundamental na passagem do sistema do laboratório para ambiente real, isto é, depois de concluída a fase de desenvolvimento e validação.

Parte I

1. Abra a aplicação "Altera Quartus II" e crie um novo projeto para a FPGA Altera Cyclone IV EP4CE115F29C7. Poderá designar o projeto e a entidade *top-level* como *DebugDemo*. O projeto consistirá num contador binário crescente, apresentado na figura 1 e descrito no ficheiro *CntUp.vhd.* O módulo *top-level DebugDemo.vhd* interliga os portos do contador a dispositivos do kit DE2-115 (figura 2). Edite os ficheiros com o código fonte fornecido, grave-os com os nomes indicados, importe o ficheiro DE2_115.qsf, compile o projeto, programe a FPGA (figura 3) e responda às questões dos pontos seguintes.
    a. Qual a frequência de incremento (atualização) do contador?
    b. Que bits de saída do contador não estão ligados aos LEDs?
    c. Qual a frequência do bit de saída mais significativo do contador?
    d. Qual a frequência do bit de saída menos significativo do contador?
    e. Qual a frequência do bit de saída menos significativo do contador visível nos LEDs?
    f. Quanto tempo demora um ciclo completo de contagem (entre 2 passagens por zero?

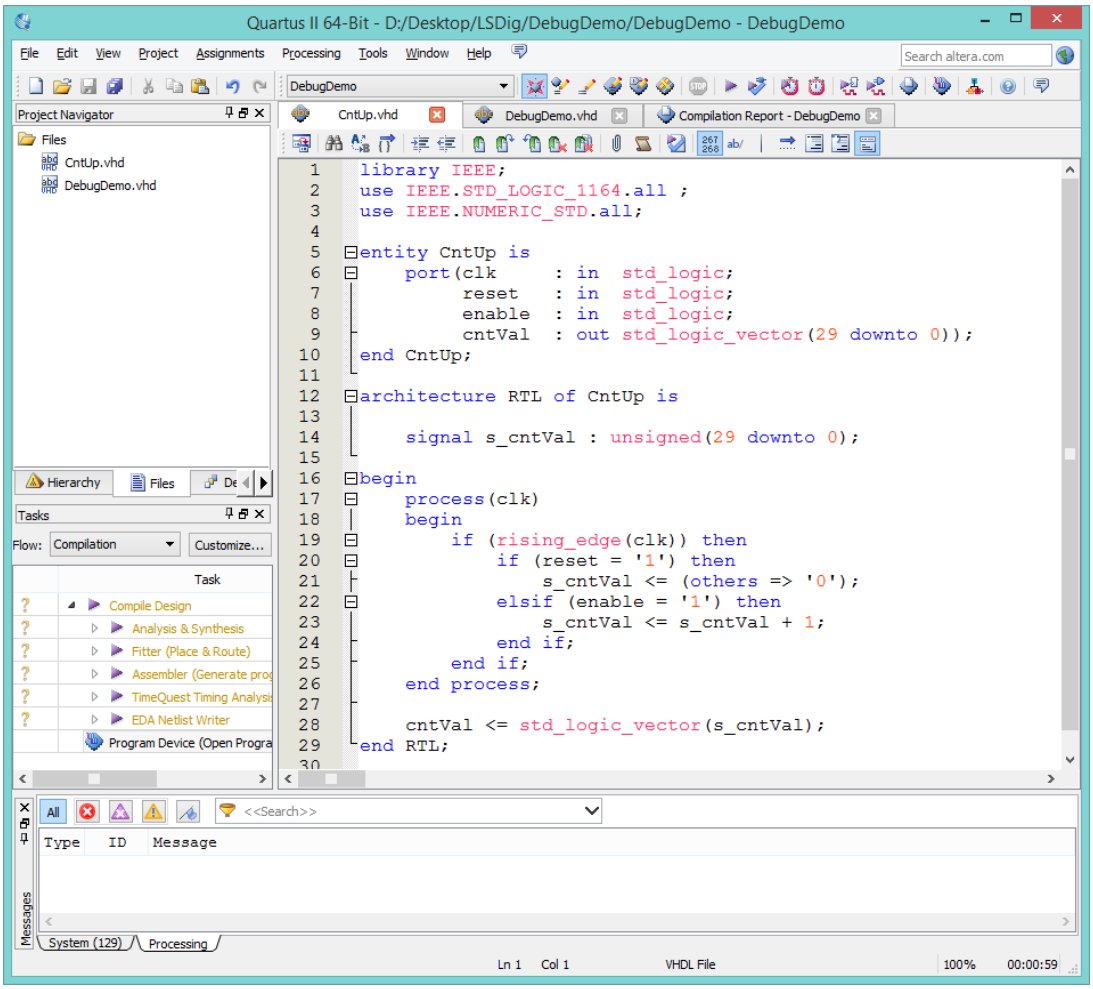

Figura 1 – Código fonte do módulo *CntUp.vhd*.

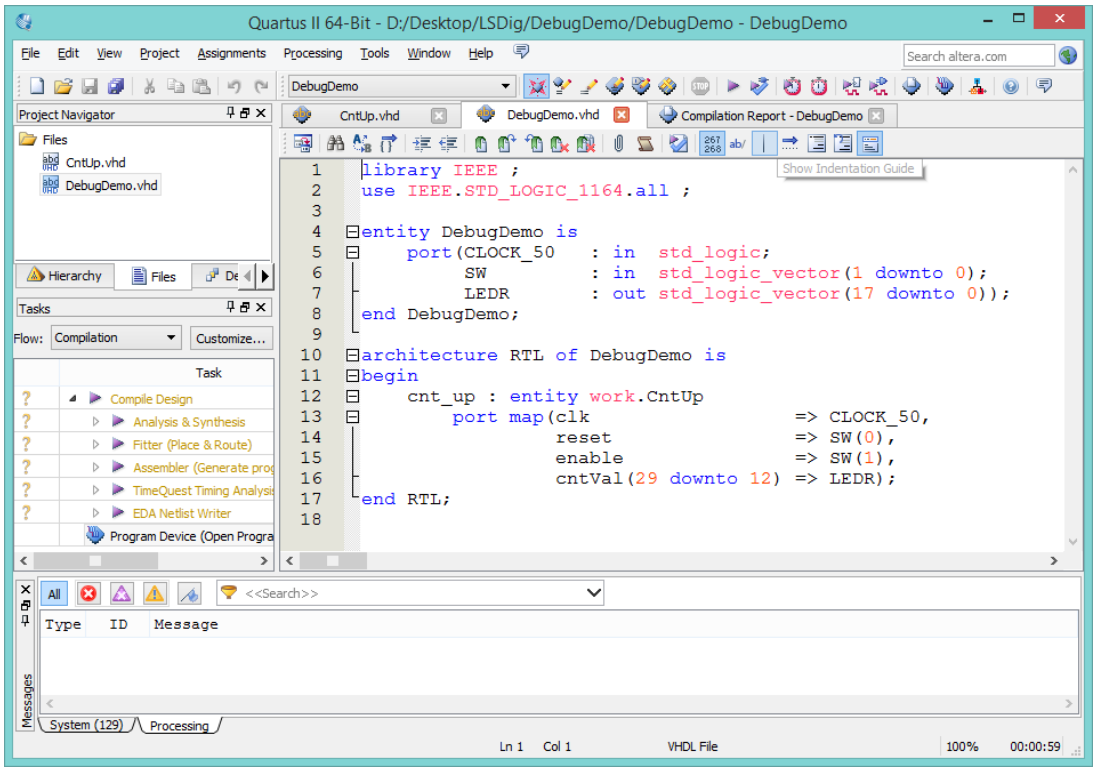

Figura 2 – Código fonte do módulo *DebugDemo.vhd*.

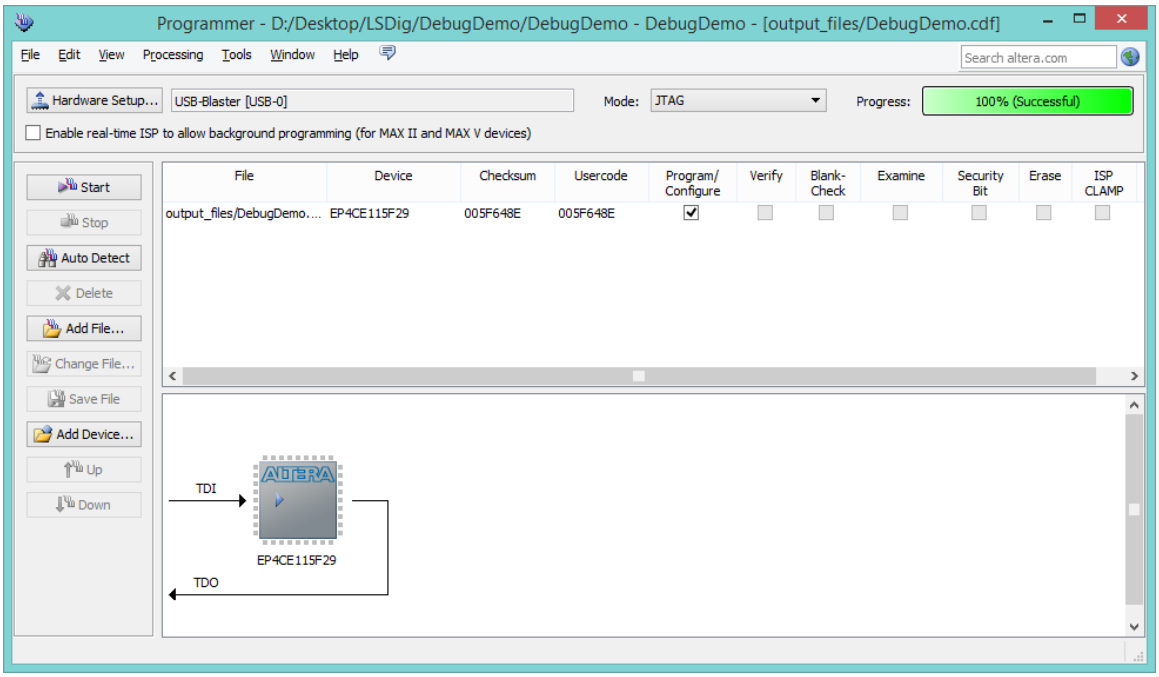

Figura 3 – Aplicação de programação após configuração bem sucedida da FPGA.

2. Devido à elevada frequência de comutação, a maioria dos bits do contador não podem ser devidamente observados a olho nu em dispositivos simples como os LEDs. Para resolver este problema e avaliar o funcionamento correto de todos os bits do contador vamos recorrer a uma ferramenta, designada *Analisador Lógico Integrado*, tipicamente disponibilizada pelos fabricantes das FPGA. Estas ferramentas permitem especificar os sinais do sistema que pretendemos visualizar, adicionar de forma transparente e automática a lógica necessária para a sua captura, armazenamento e transferência para um computador de desenvolvimento onde será realizada a sua visualização. A lógica adicionada para este efeito utiliza os próprios recursos lógicos programáveis da FPGA. A interface para transferir os sinais capturados é também o usado na programação da FPGA (denominado JTAG) o que é bastante conveniente. Para usar esta facilidade, o primeiro passo é a criação de um ficheiro no "Altera Quartus II" do tipo *SignalTap II Logic Analyser File* (figura 4).

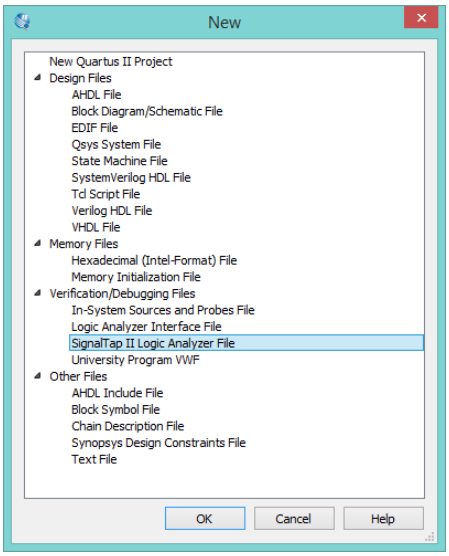

Figura 4 – Criação de um ficheiro do tipo *SignalTap II Logic Analyser File*.

3. Após a criação do ficheiro do tipo *SignalTap II Logic Analyser File* é apresentada a aplicação da figura 5, onde devem ser especificados os sinais que se pretende capturar e visualizar, o sinal de *clock* usado para estabelecer a frequência de amostragem dos sinais, o número de amostras capturadas e as condições que disparam a amostragem. A configuração destes parâmetros vai ser descrita nos próximos pontos.

4. O primeiro passo é a adição dos nodos (sinais e portos) do sistema que se pretende capturar. Para tal, deve ser selecionada a opção *Add Nodes...* do menu acessível com o botão direito do rato na área *Setup* mostrada na figura 5. Os portos a selecionar para captura e visualização no analisador lógico são os apresentados na figura 6.

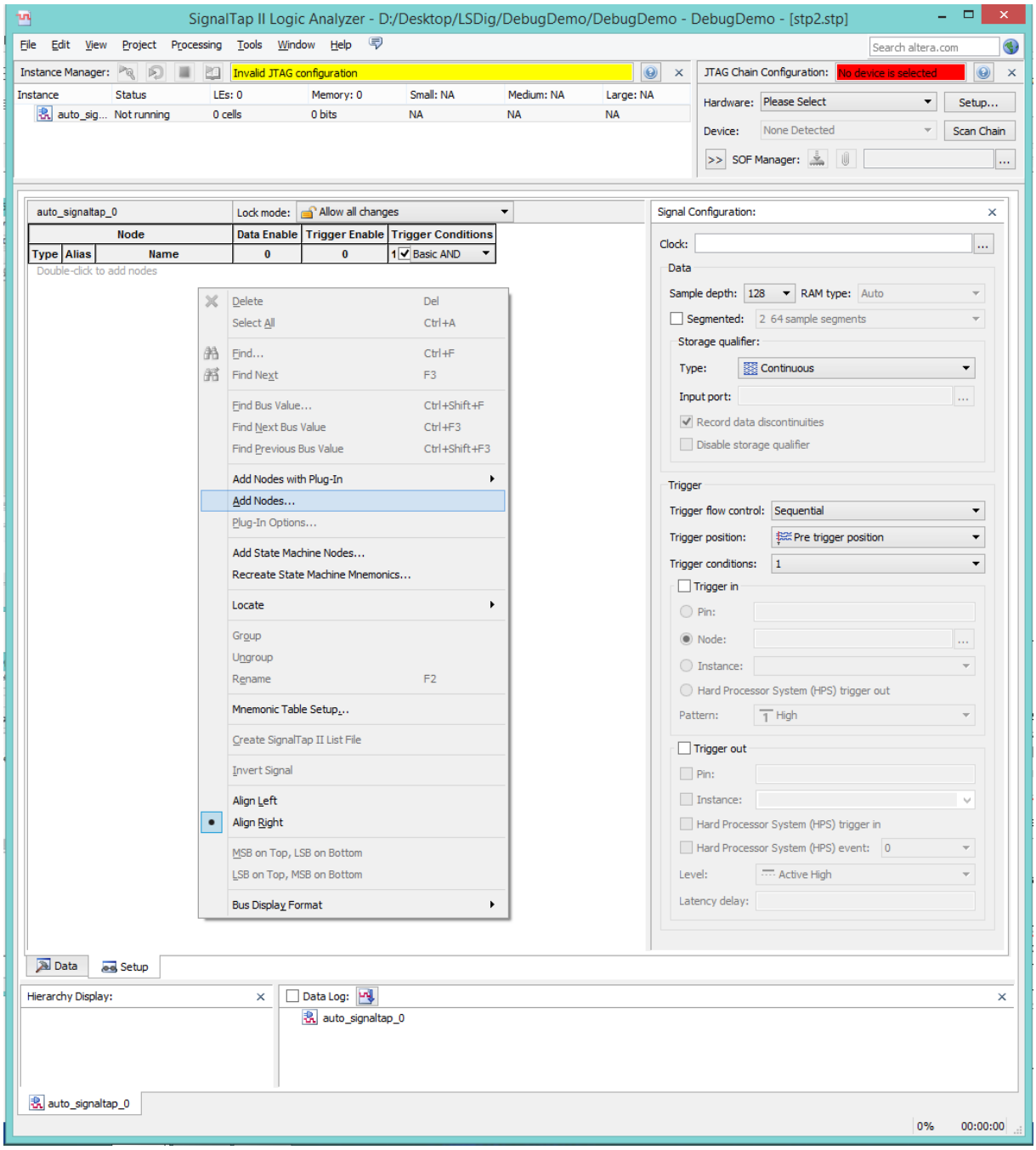

Figura 5 – Aspeto inicial da aplicação *SignalTap II Logic Analyser*.

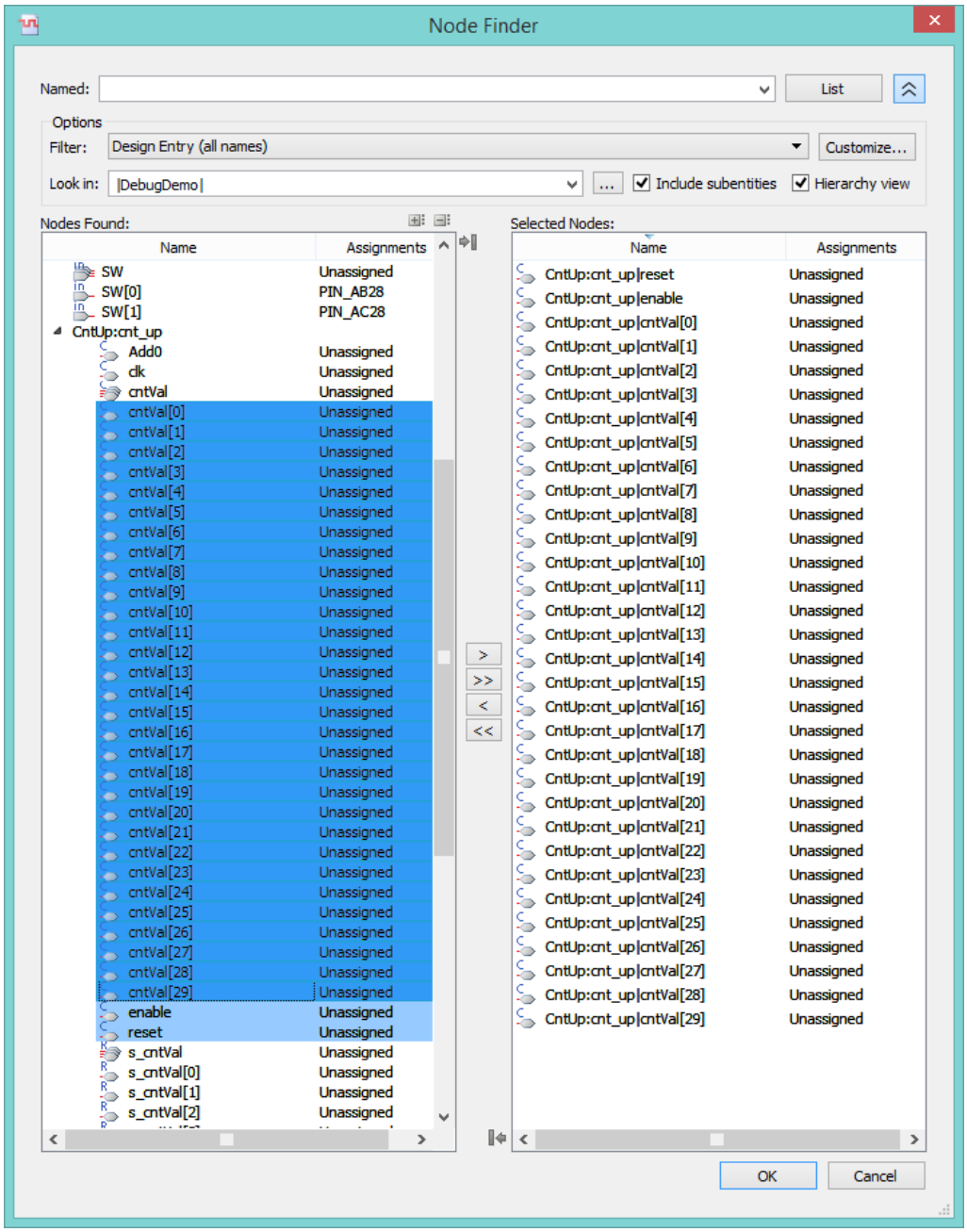

Figura 6 – Portos do contador selecionados (**`reset`**, **`enable`** e **`cntVal`**).

5. Após premir OK, deverá configurar na janela principal os seguintes parâmetros (figura 7):
   - *Hardware*: USB-Blaster (dispositivo/interface usado para programação e depuração)
   - *Clock*: CLOCK_50 (sinal de relógio usado para estabelecer os instantes de amostragem / frequência de amostragem – 50 MHz)
   - *Sample Depth*: 4K (número de amostras consecutivas a capturar)
   - *Trigger Conditions* (condições que levam ao disparo da captura de amostras; neste caso *reset = 0* e *enable = 1* para capturar o instante em que é libertado o *reset*)
     - CntUp:cnt_up|reset: 0
     - CntUp:cnt_up|enable: 1

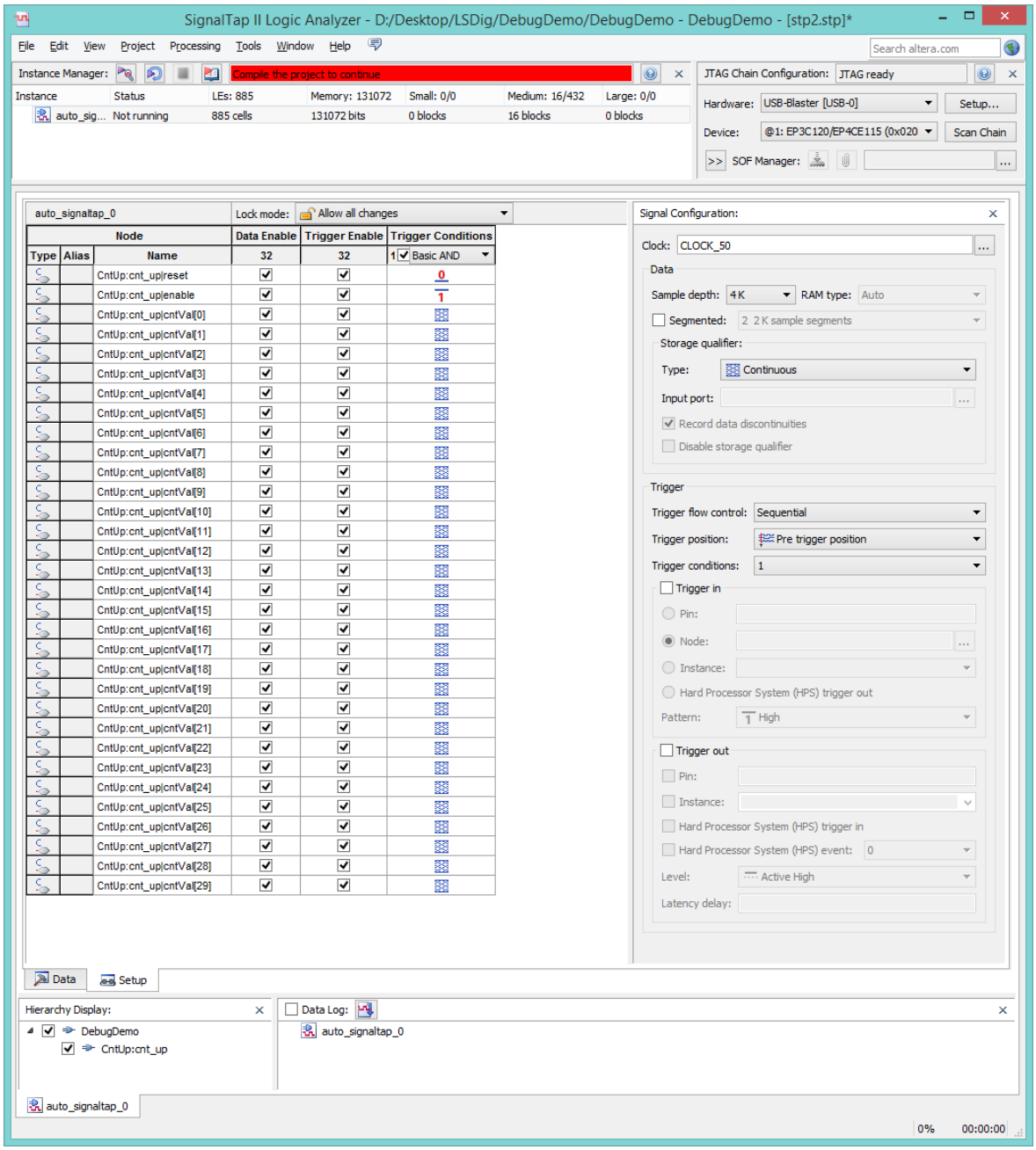

Figura 7 – Aspeto da aplicação *SignalTap II Logic Analyser* com parâmetros configurados.

6. Seguidamente, grave o ficheiro com o nome *DebugDemo.stp* (figura 8):

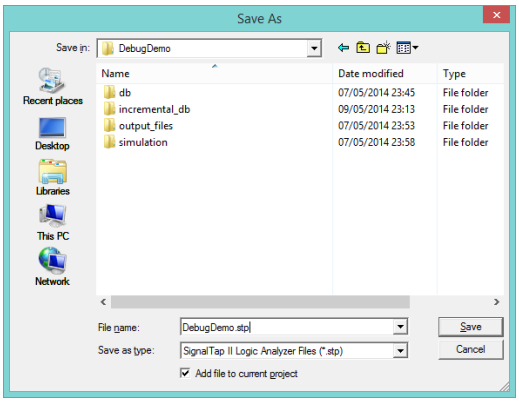

Figura 8 – Gravação do ficheiro na aplicação *SignalTap II Logic Analyser*.

7. Uma vez que as componentes de hardware do analisador lógico usam recursos lógicos programáveis da FPGA, antes de efetuar a captura e visualização dos sinais, é necessário voltar a compilar o projeto no "Altera Quartus II". O projeto consiste nos ficheiros *DebugDemo.vhd*, *CntUp.vhd* e *DebugDemo.stp*. O ficheiro *top level* deve continuar a ser o *DebugDemo.vhd* (figura 9). Após compilação, execute a aplicação "SignalTap II Analyser" tal como ilustrado na figura 10.

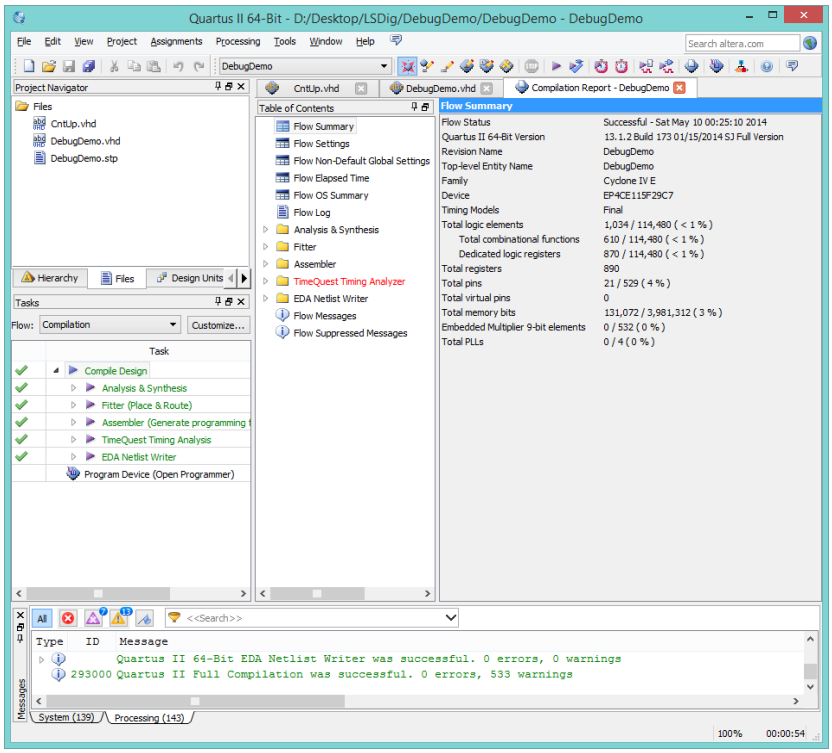

Figura 9 – Aplicação "Altera Quartus II" após compilação do projeto.

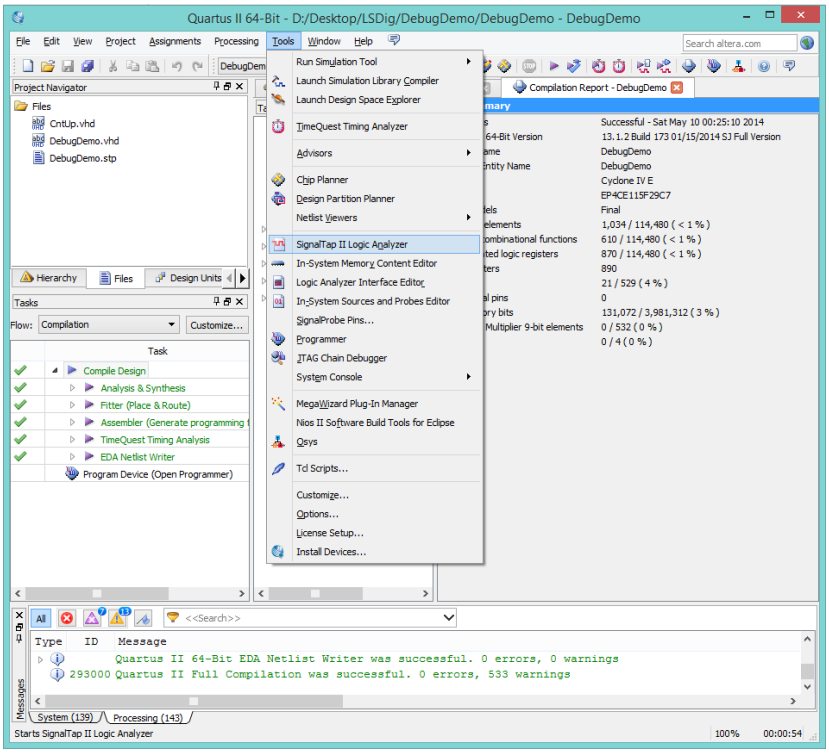

Figura 10 – Execução da aplicação "SignalTap II Analyser".

8. Especifique o ficheiro SOF a usar para configurar a FPGA (*output_files/DebudDemo.sof*) e programe a FPGA (canto superior direito da figura 11). O ficheiro SOF inclui quer a configuração da lógica do sistema desenvolvido, quer a configuração da lógica correspondente às componentes de hardware do analisador lógico. Durante a programação da FPGA a aplicação deve apresentar o aspeto da figura 12.

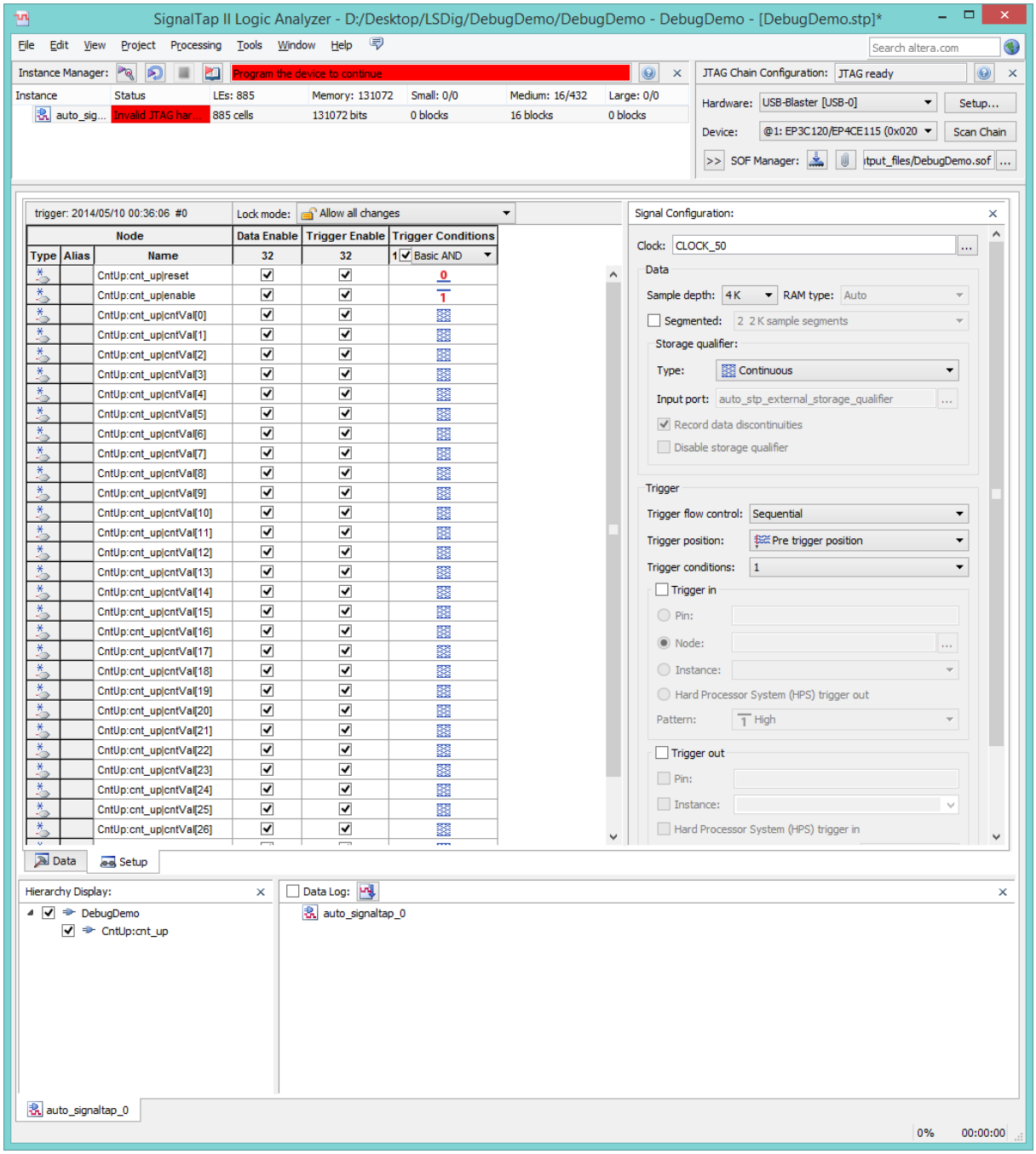

Figura 11 – Aplicação "SignalTap II Analyser" antes da programação da FPGA.

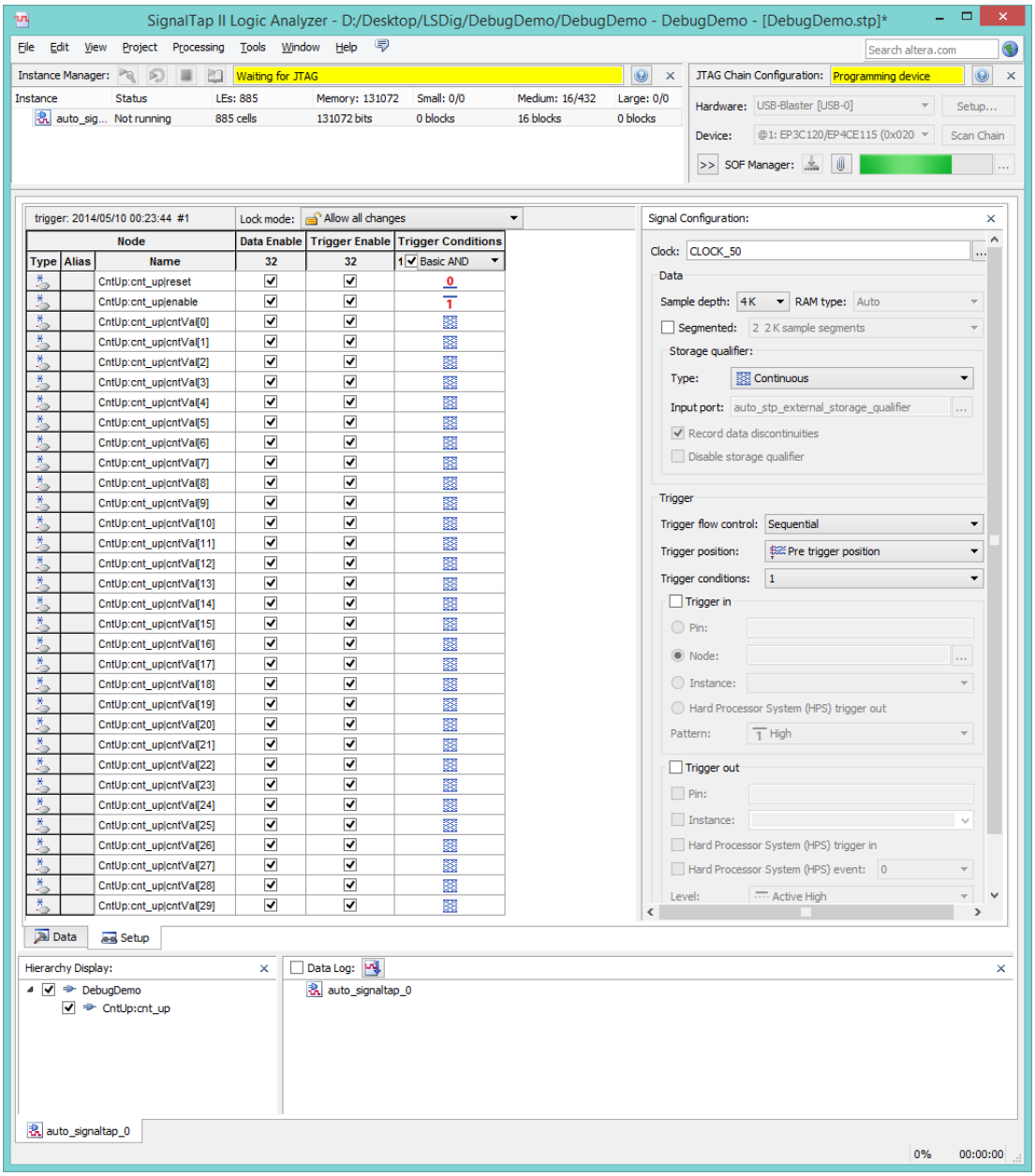

Figura 12 – Aplicação “SignalTap II Analyser” durante a programação da FPGA.

9. Após a programação da FPGA mude a janela principal da aplicação “SignalTap II Analyser” de *Setup* para *Data* (figura 13). Neste ponto está tudo configurado e preparado para ser iniciada a captura de amostras dos sinais pretendidos. Para tal, basta selecionar o comando “*Processing → Run Analysis*” e de seguida desativar a entrada de *reset* e ativar a entrada de *enable* do contador de forma a que seja satisfeita a condição de *trigger* e iniciada a captura dos sinais pretendidos. Uma vez recolhidas 4 K amostras de cada sinal, os respetivos valores são transferidos para o computador e visualizados na aplicação “SignalTap II Analyser” (figuras 14 e 15).

10. Podem ser visualizados instantes particulares da contagem se forem definidas condições de *trigger* mais restritas. A título de exemplo, pode ser visualizado o instante de *wrap-around* do contador através da atribuição do valor “11...110” a **`cntVal`**, o que leva a que a captura seja disparada um ciclo de relógio antes do contador voltar à contagem “00...00” (figura 16). Volte a executar o comando “*Processing → Run Analysis*” para efetuar a captura.

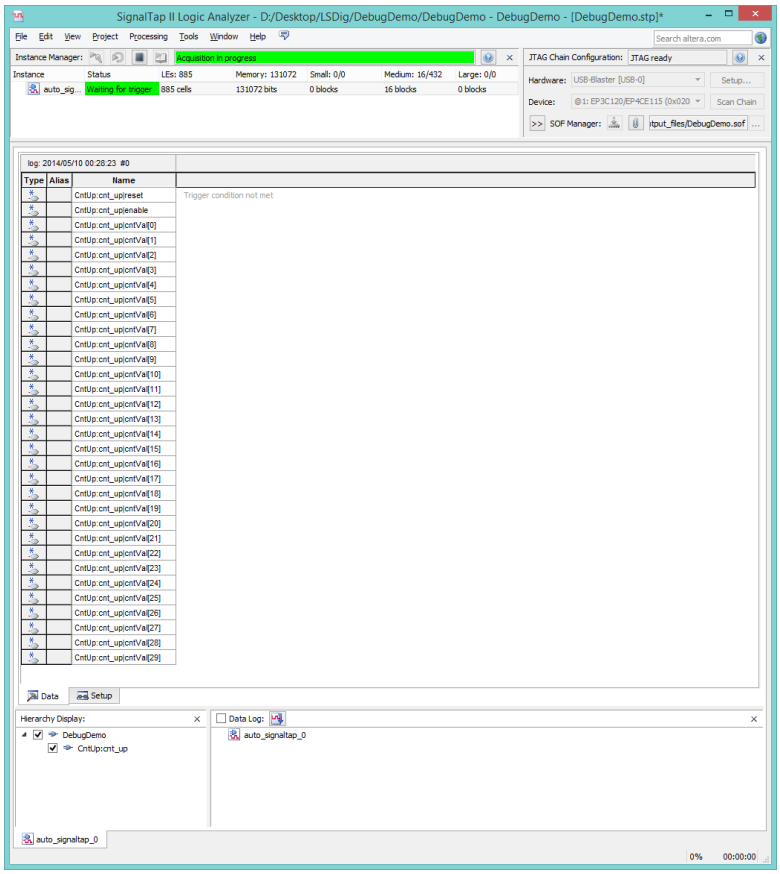

Figura 13 – Janela *Data* da aplicação "SignalTap II Analyser" (antes da captura).

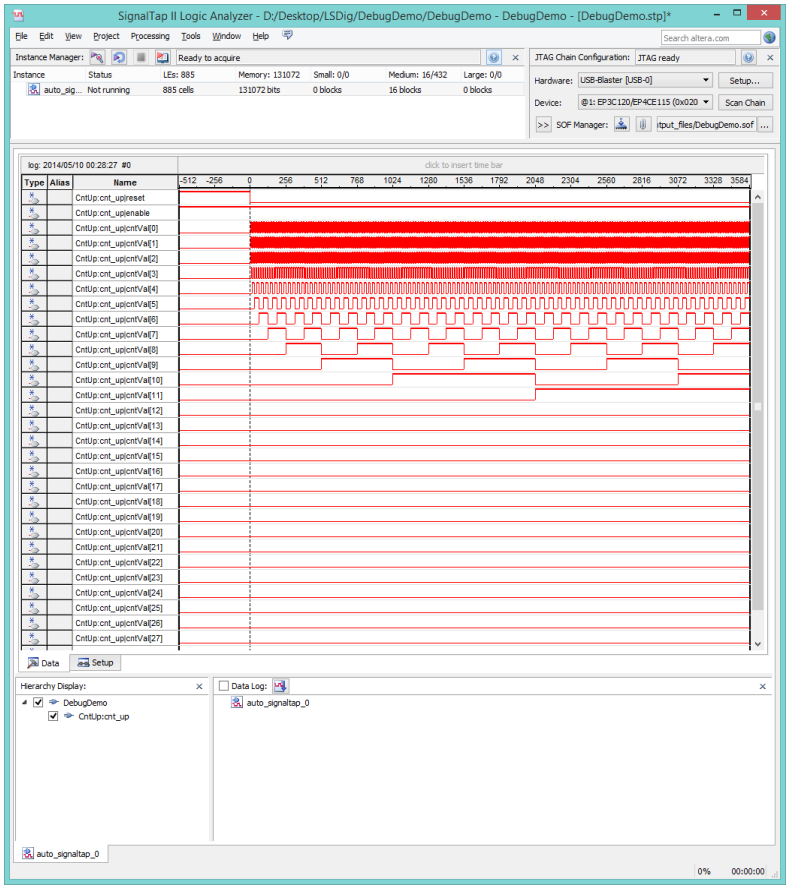

Figura 14 – Janela *Data* da aplicação "SignalTap II Analyser" (depois da captura).

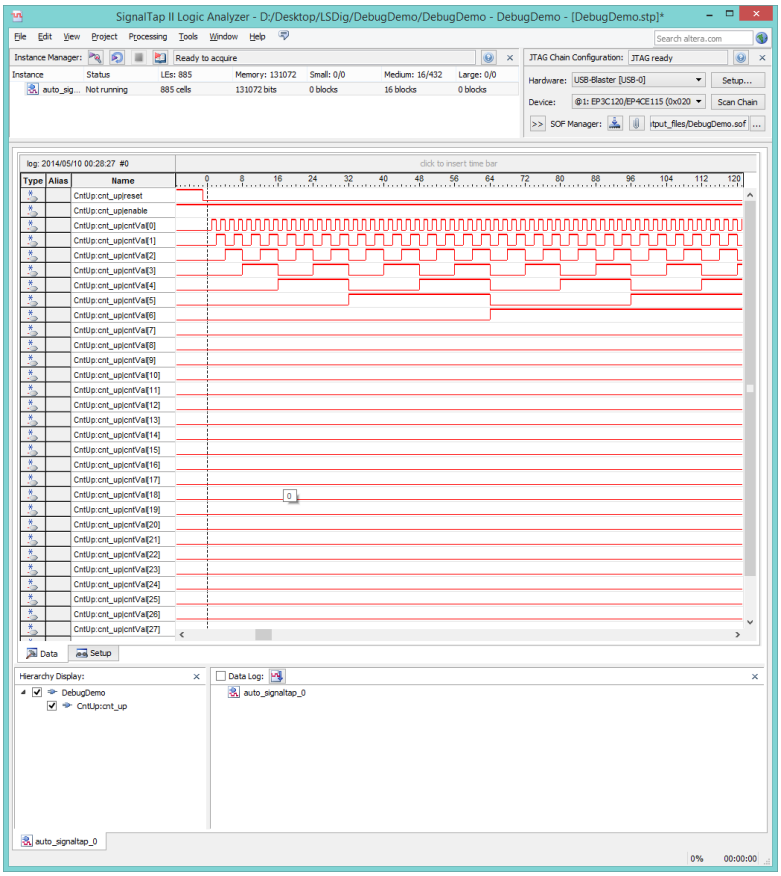

Figura 15 – Janela *Data* da aplicação “SignalTap II Analyser” (vista ampliada da captura).

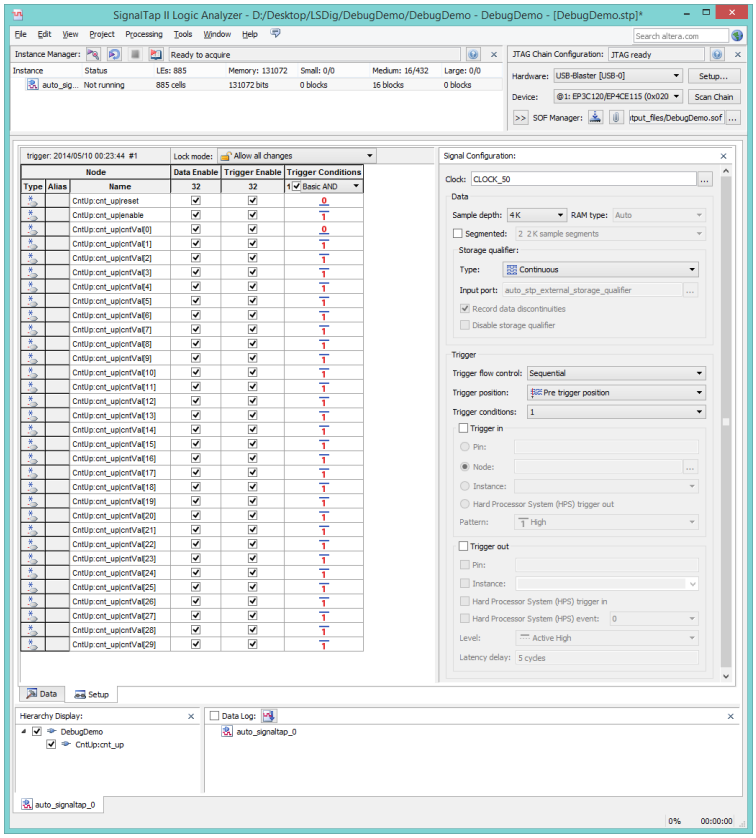

Figura 16 – Configuração do instante de disparo da captura (trigger) com base no valor dos sinais **`reset`**, **`enable`** e **`cntVal`**.

Parte II

Programar a memória FLASH (não volátil) existente no kit DE2-115 com a configuração da FPGA é útil para, uma vez concluído o desenvolvimento e validação do sistema, o kit poder ser ligado e a FPGA programada sem a intervenção do computador. Para tal é necessário realizar um conjunto de passos descritos nesta parte do guião. **Nota:** por conveniência vamos usar o projeto *DebugDemo* para este efeito, embora não seja usual programar uma memória FLASH com uma configuração de depuração de um projeto, e muito menos com um analisador lógico integrado.

1. O primeiro passo a efetuar é a conversão do ficheiro SOF num ficheiro POF para programação da FLASH, através do comando *File->Convert Programming Files... no* "Altera Quartus II" (figura 17).

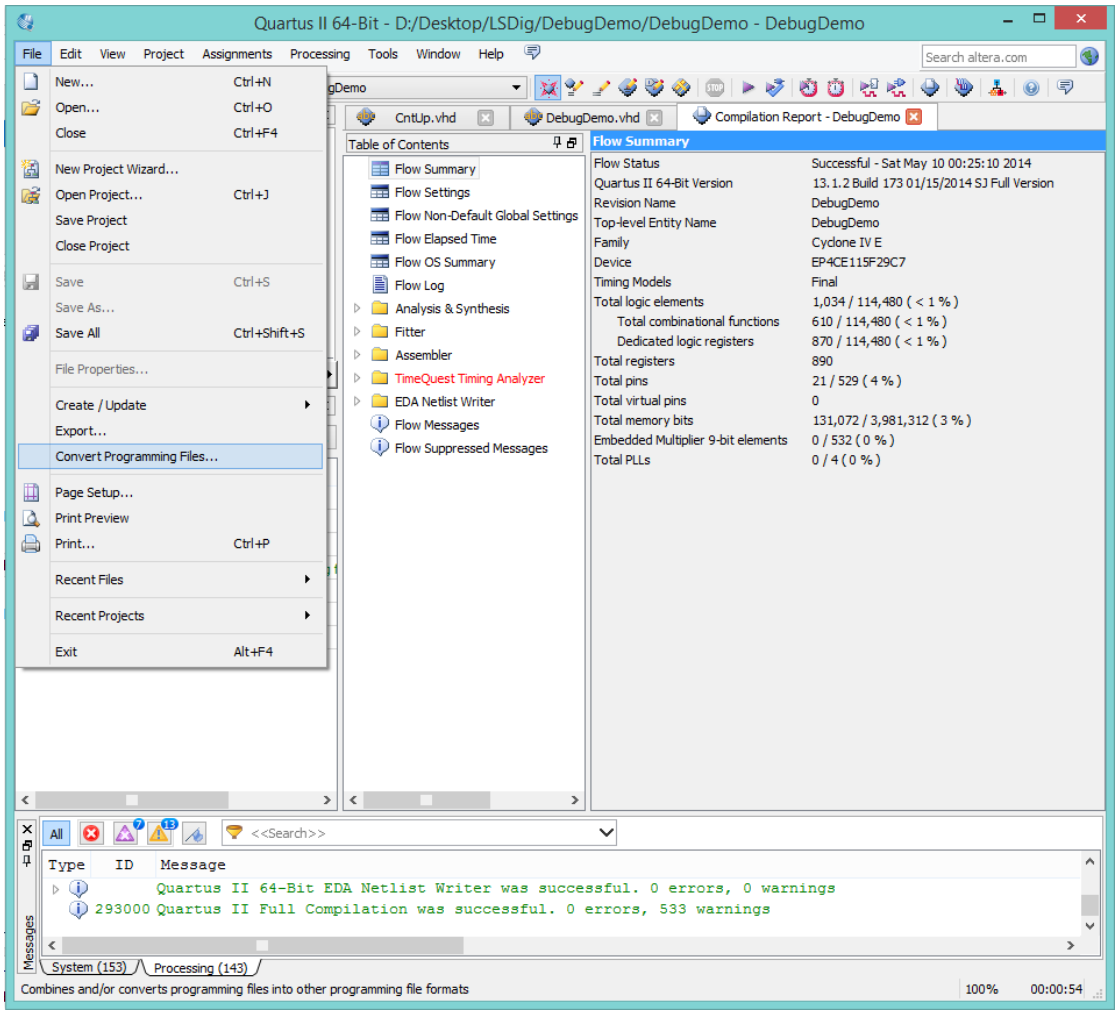

Figura 17 – Execução do comando *Convert Programming Files...*

2. De seguida especifique o nome do ficheiro POF (output_files/DebugDemo.pof) e o tipo de FLASH instalada no kit DE2_115 (EPCS64) – ver figura 18.

3. Adicione o ficheiro SOF (DebugDemo.sof) à página 0 da memória FLASH, selecionando *SOF Data* e premindo *Add File* (figura 19). De seguida prima *Generate* para gerar o ficheiro POF.

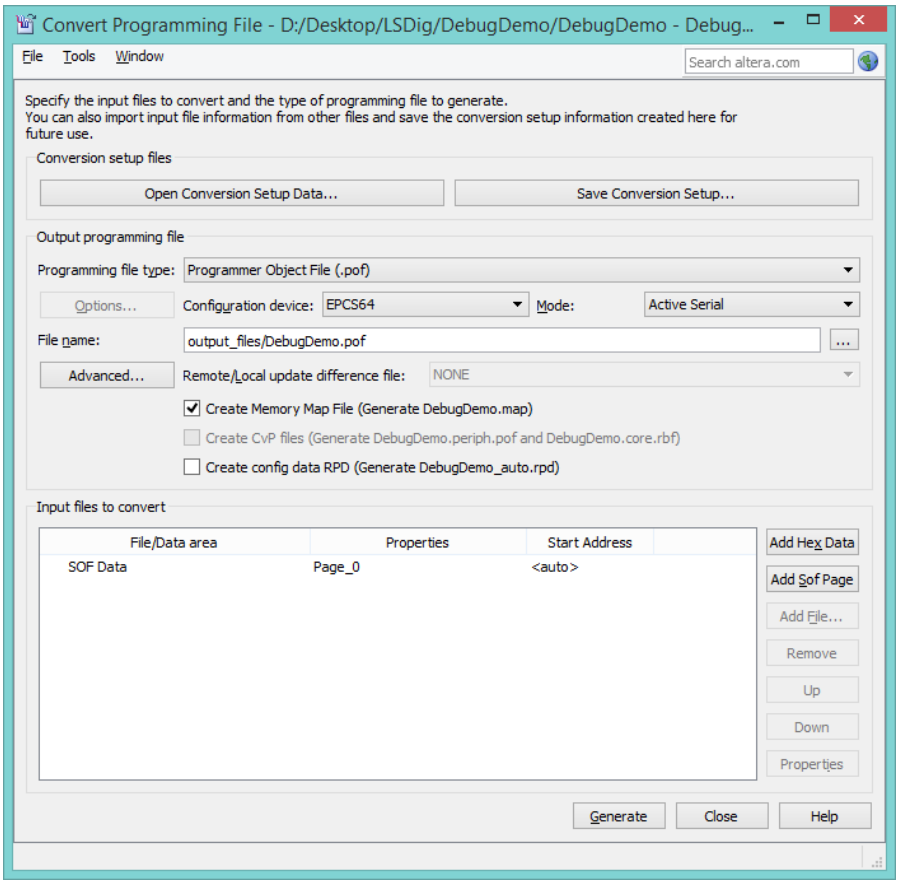

Figura 18 – Configuração do nome do ficheiro POF e do tipo de FLASH.

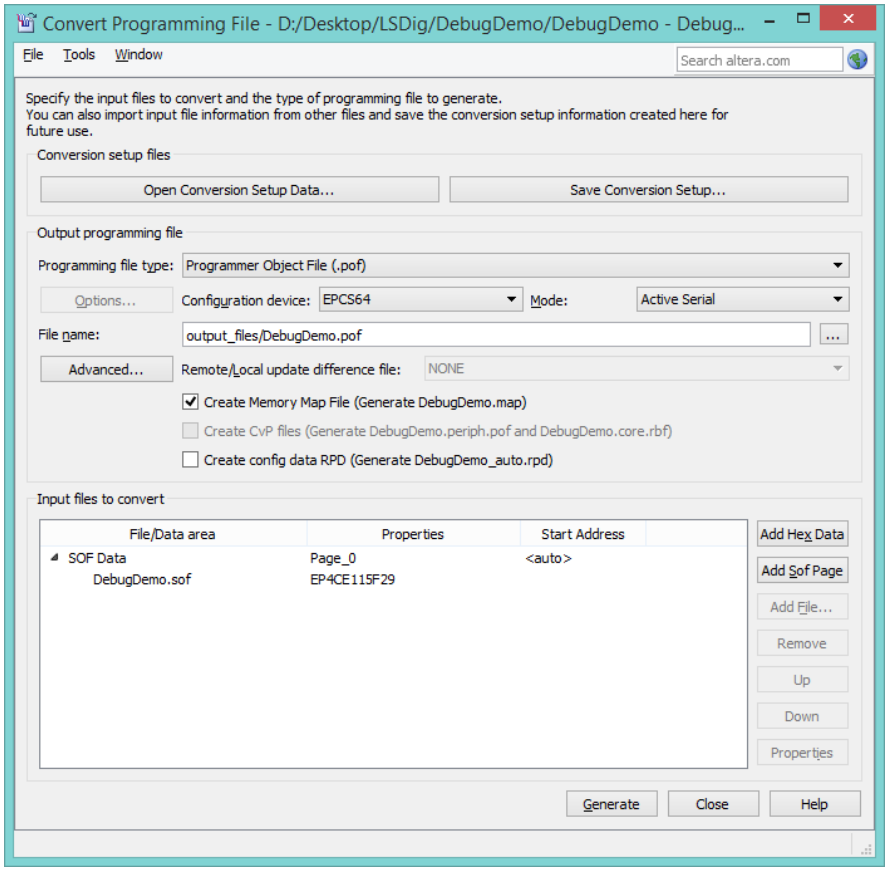

Figura 19 – Adição do ficheiro SOF (DebugDemo.sof).

4. Uma vez gerado o ficheiro POF, para programar a memória FLASH do kit DE2-115, é necessário executar o comando *Program Device* no "Altera Quartus II", selecionar o modo "Active Serial Programming" (figura 20), premir "Add File" para adicionar o ficheiro "DebugDemo.pof" e selecionar as opções *Program/Configure* e *Verify*. Para efetuar a programação da FLASH propriamente dita, o botão *Run<->Prog* do kit DE2-115 deve ser colocado na posição *Prog* e deve ser premido o botão *Start* na aplicação de programação.

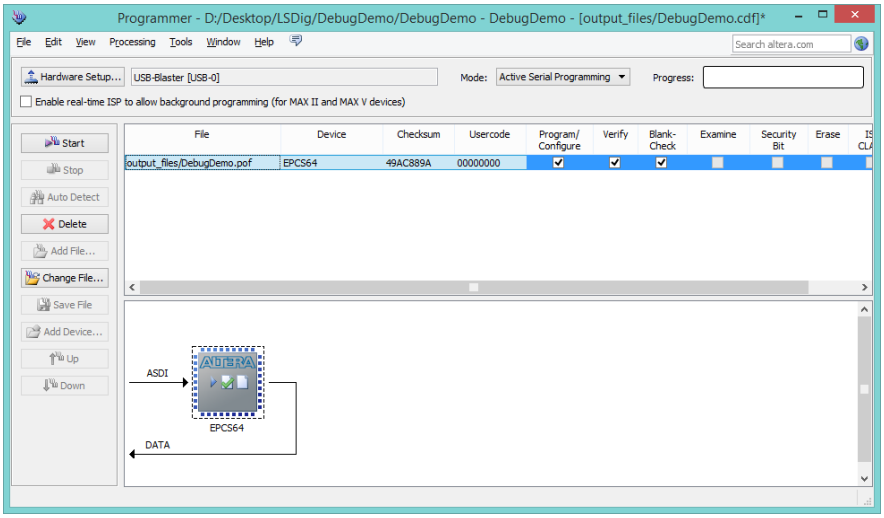

Figura 20 – Seleção do ficheiro e opções para programação da memória FLASH.

5. Uma vez concluída a programação da memória FLASH, a aplicação deve apresentar o aspeto da figura 21. De notar que o processo de programação é relativamente lento.

6. Uma vez programada a FLASH, a placa pode ser desligada e o botão *Run<->Prog* do kit DE2-115 deve ser colocado na posição *Run*. Seguidamente, a placa pode ser ligada, a FPGA é configurada e deve arrancar com a configuração pretendida sem qualquer intervenção do computador.

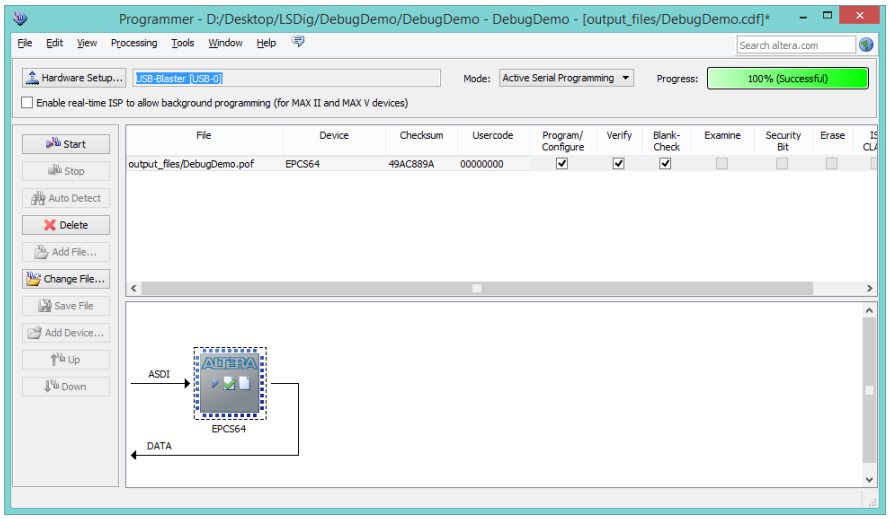

Figura 21 – Aplicação de programação após configuração da memória FLASH.

7. Repita os pontos 4, 5 e 6 usando o ficheiro "DE2_115_Default.pof" (disponível no site de LSDig) de forma a repor a configuração de fábrica (*default*) do kit DE2-115.

PDF criado em 14/05/2015 às 13:30:10