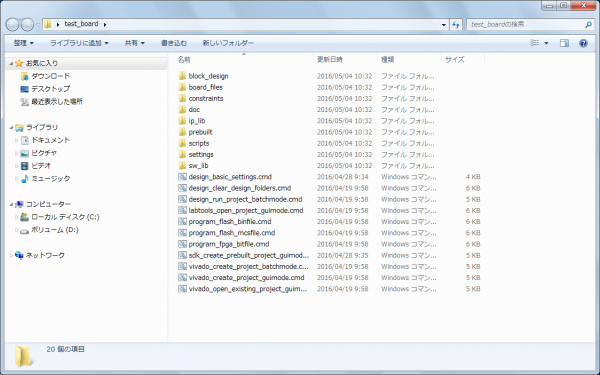
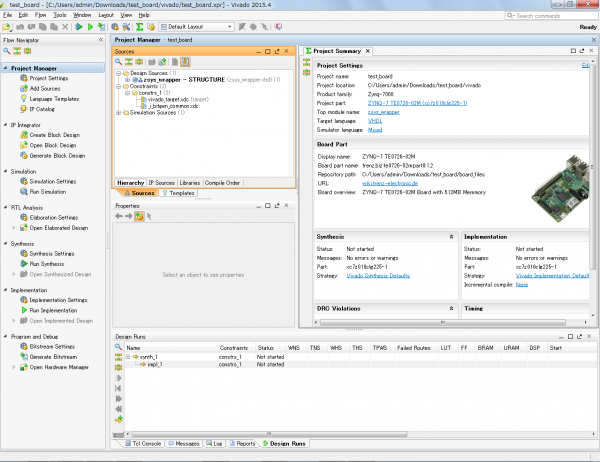
**1.** **Criação do projeto Vivado**

Este projeto de LED tikatika baseia-se no projeto de design de referência. O interior da pasta que descompactou o zip de design de referência é o seguinte.

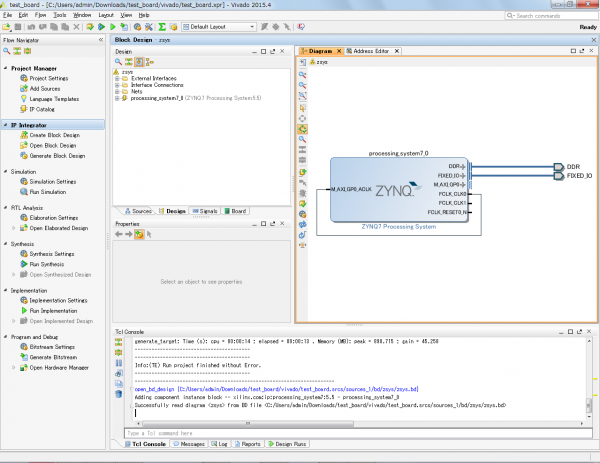
[](http://www.trenz.jp/wp-content/uploads/2016/05/project_directory.png)

Edite design\_basic\_settings.cmd da mesma maneira que no Hello World.  
Quando a edição está acabada vivado\_create\_project\_batchmode.cmd → vivado\_open\_existing\_project\_guimode.cmd   
Ou vivado\_create\_project\_guimode.cmd

Execute o comando para criar um projeto Vivado e abra-o no modo GUI da Vivado.

Quando o projeto é aberto, a seguinte tela será exibida.  
[](http://www.trenz.jp/wp-content/uploads/2016/05/1-1.png)

Clique no Flow Navigator -> Integrator -> Open Block Design no lado esquerdo da tela de Vivado para exibir o diagrama de blocos.

[](http://www.trenz.jp/wp-content/uploads/2016/05/1-2.png)

Atualmente, apenas os módulos necessários para o Hello World são usados.  
De agora em diante, vou adicionar os módulos necessários para iluminação LED aqui.

**2. Crie seu próprio IP**

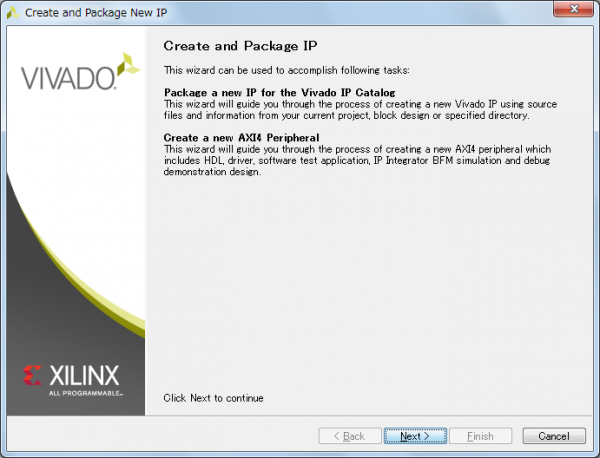
Em seguida, crie uma parte que controle a saída para o LED.  
De qualquer forma, deixe Vivado como está.  
Vá para test\_board / ip\_lib na pasta  
Em seguida, crie um arquivo vhdl chamado led\_ctrl.vhd com um editor apropriado neste,  
Edite-o da seguinte maneira.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | -- led\_ctrl.vhd  library IEEE;  use IEEE.STD\_LOGIC\_1164.ALL;    use IEEE.STD\_LOGIC\_ARITH.ALL;  use IEEE.STD\_LOGIC\_UNSIGNED.ALL;    entity led\_ctrl is      Port ( clk\_ip      : in   std\_logic;             led\_op      : out  std\_logic;             interval\_ip : in   std\_logic\_vector( 31 downto 0)             );  end led\_ctrl;    architecture Behavioral of led\_ctrl is  signal clk         : std\_logic;  signal led\_toggle  : std\_logic;  signal count :  std\_logic\_vector( 31 downto 0);  signal interval :  std\_logic\_vector( 31 downto 0);  begin  clk <= clk\_ip;  led\_op <= led\_toggle;  process(clk) begin  if(clk'event and clk='1') then      if(interval /= interval\_ip) then  count <= X"00000000";  elsif(count = interval) then      if( count /= 0 ) then      count <= X"00000000";      led\_toggle <= not led\_toggle;  end if;  else  count <= count + 1;  end if;  end if;  end process;    process(clk) begin  if(clk'event and clk='1') then     interval <= interval\_ip;  end if;  end process;  end Behavioral; |

Test\_board / ip\_lib / led\_ctrl.vhd foi adicionado até agora. Voltarei a Vivado novamente aqui. Ao clicar em Ferramentas -> Criar e pacote IP na guia Vivado,

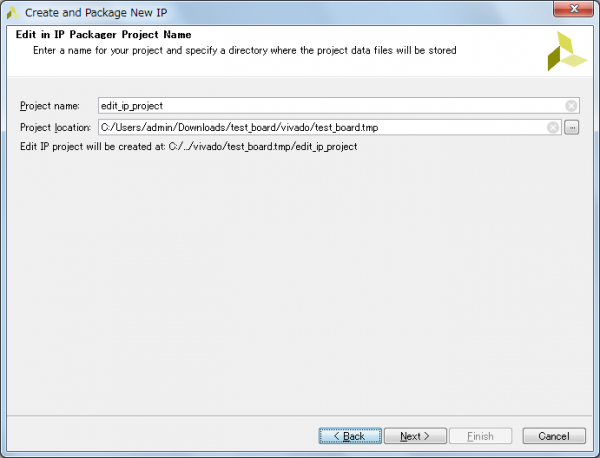
À medida que a tela abaixo aparece, pressione o botão Avançar.

[](http://www.trenz.jp/wp-content/uploads/2016/05/2-2.png)

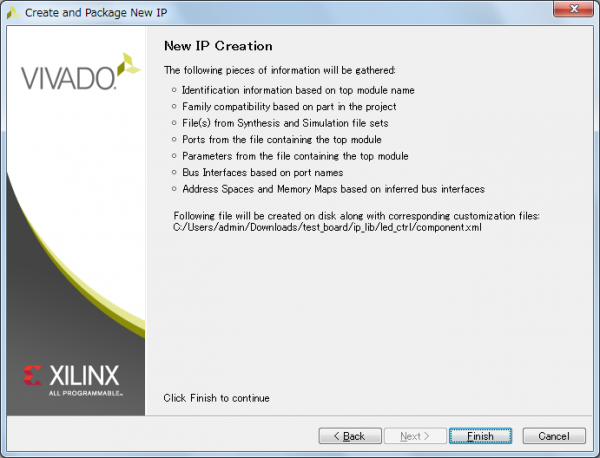
Na tela abaixo, selecione Pacote um diretório especificado e pressione o botão Avançar.

[](http://www.trenz.jp/wp-content/uploads/2016/05/2-3.png)

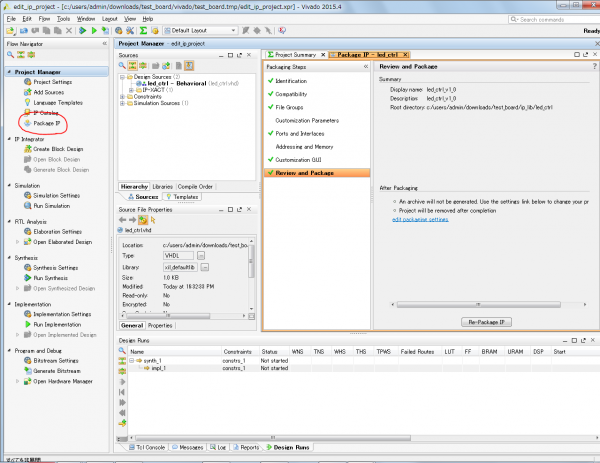
Na tela a seguir, especifique o test\_board / ip\_lib / led\_ctrl criado anteriormente no Diretório e pressione o botão Próximo.

[](http://www.trenz.jp/wp-content/uploads/2016/05/2-5.png)

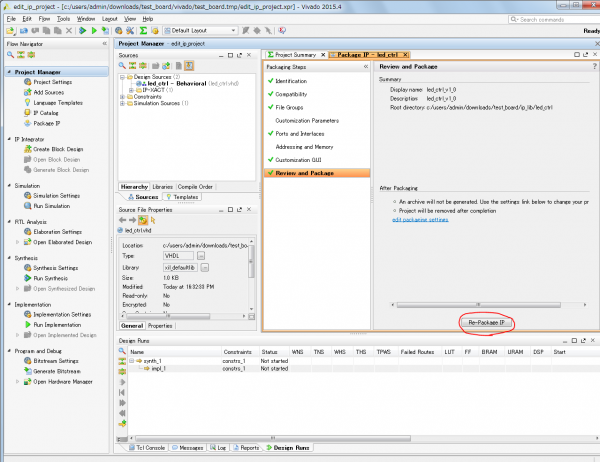
Quando você pressiona o botão Finalizar na tela a seguir, o projeto para o seu IP residencial será iniciado em uma janela separada.

[](http://www.trenz.jp/wp-content/uploads/2016/05/2-6.png)

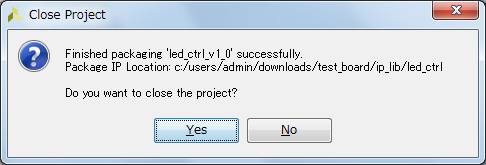
Clique no Flow Navigator -> Project Manager -> Package IP no lado esquerdo da tela para abrir a tela Package IP.

[](http://www.trenz.jp/wp-content/uploads/2016/05/2-7.png)

Clique no botão Pacote IP (ou Re-Package IP) na guia Revisão e Pacote da tela IP do Pacote.

[](http://www.trenz.jp/wp-content/uploads/2016/05/2-8.png)

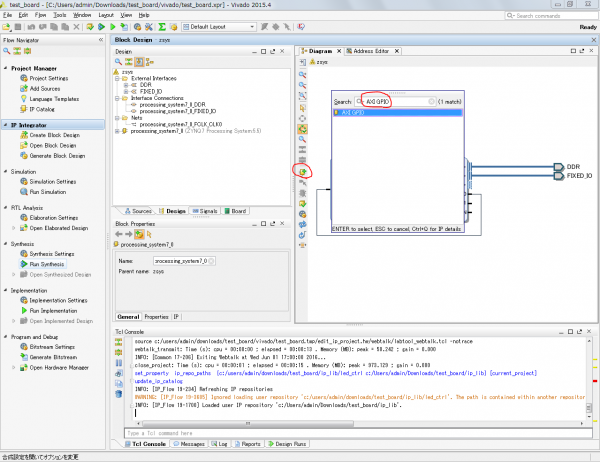
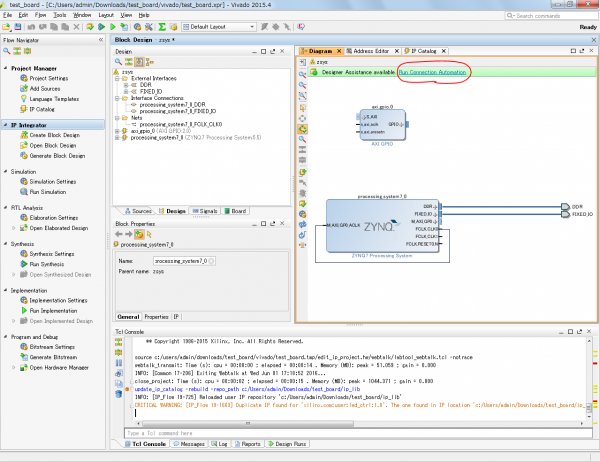
Depois que o IP for criado, a seguinte janela será exibida. Pressione o botão sim,  
Saia do projeto para criação de IP.

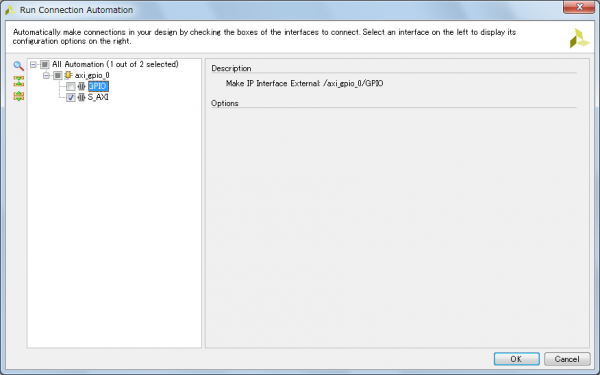
[](http://www.trenz.jp/wp-content/uploads/2016/05/2-9.png)

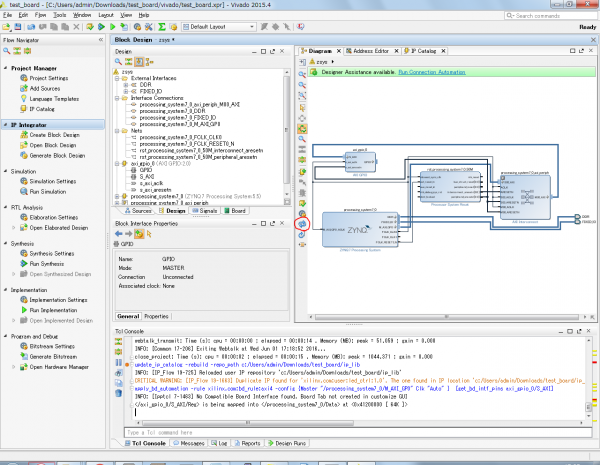
Criar o seu próprio IP para controlar a saída para o LED é completado pelo procedimento acima.

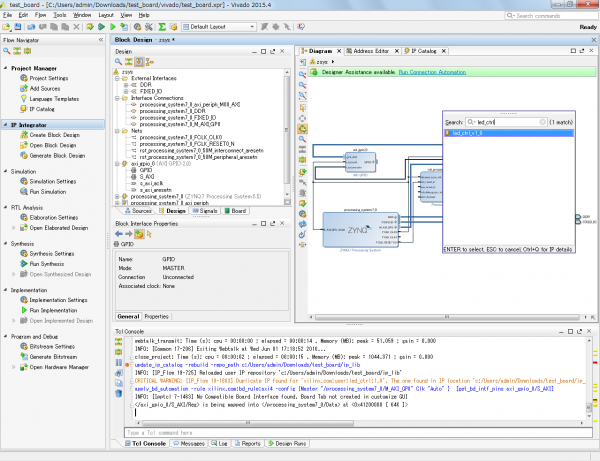
### 3. Edite o diagrama de blocos

Você retornará à tela do diagrama de blocos de Vivado.  
Vou conectar meu IP inicial criado no procedimento anterior da seção PS.  
Primeiro, selecione AXI GPIO na tela Adicionar IP e adicione-o.

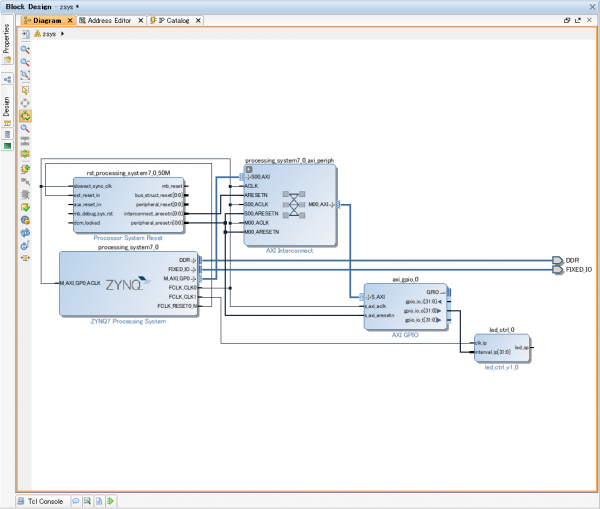
[](http://www.trenz.jp/wp-content/uploads/2016/05/3-11.png)Clique em Run Connection Automation acima do diagrama de bloco,  
Ele conectará automaticamente os dois blocos.[](http://www.trenz.jp/wp-content/uploads/2016/05/3-2.png)

Verifique apenas S\_AXI na seguinte janela.  
[](http://www.trenz.jp/wp-content/uploads/2016/05/3-3.png)

Como resultado da fiação, foram adicionados alguns blocos e tornou-se um layout desarrumado como abaixo.  
Clique em Regentate Layout no círculo vermelho na figura abaixo para organizar o layout.[](http://www.trenz.jp/wp-content/uploads/2016/05/3-4.png)

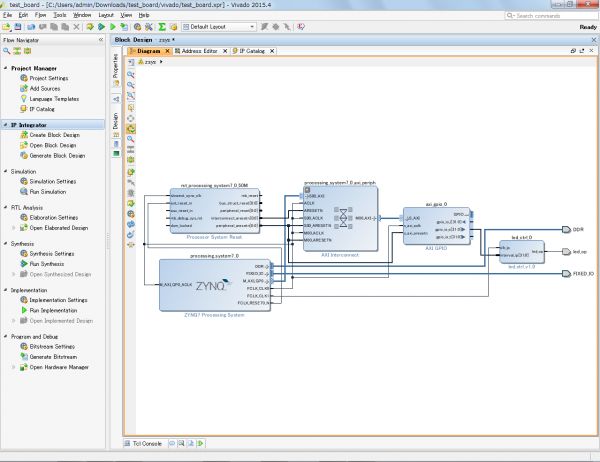
Da mesma forma, adicione seu próprio IP pesquisando led\_ctrl de Adicionar IP.[](http://www.trenz.jp/wp-content/uploads/2016/05/3-5.png)

Conecta o porto de led\_ctrl\_ 0. Você pode conectar as portas arrastando com o mouse.  
Existem dois lugares para se conectar.  
· Clk\_ip de led\_ctrl\_ 0 e FCLK 1 de processing\_system 7 \_ 0  
· Interval\_ip de led\_ctrl\_0 e gpio\_io\_o de axi\_gpio  
axi\_gpio\_o é exibido clicando em "+" ao lado de GPIO.  
Se você executar o Regenerate Layout conectando-se acima, parecerá a figura abaixo.

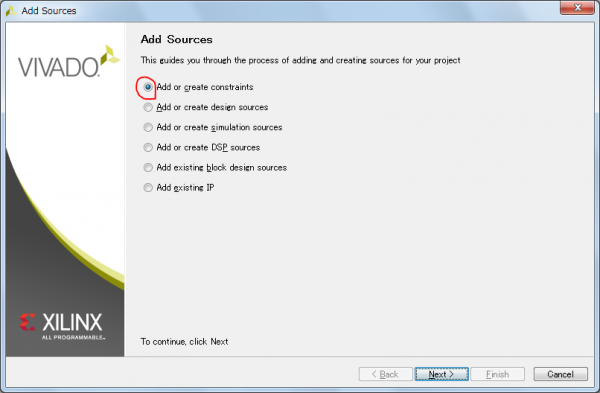
[](http://www.trenz.jp/wp-content/uploads/2016/05/3-6.png)

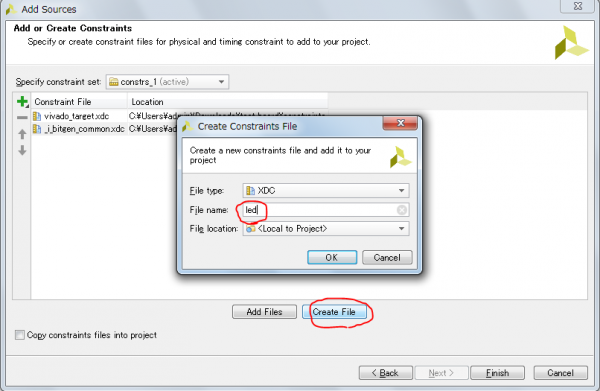
Finalmente, crie uma porta para saída LED.  
Clique com o botão direito do mouse em led\_ctrl\_ 0 e pressione Criar porta.  
Então a seguinte tela aparece, então pressione o botão OK com padrão.

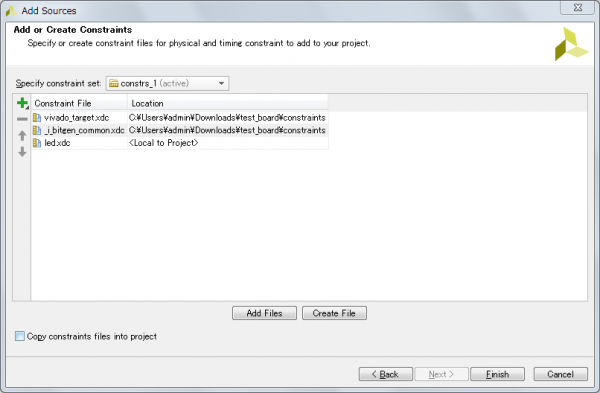
[](http://www.trenz.jp/wp-content/uploads/2016/05/3-7.png)

Quando o Regenerate Layout é executado, ele finalmente se torna o seguinte diagrama de blocos.[](http://www.trenz.jp/wp-content/uploads/2016/05/3-8.png)

**4. Criar arquivo XDC**

Em seguida, adicione o arquivo XDC para configurar o pino de saída.  
Clique em Flow Navigator -> Adicionar Fontes no lado esquerdo da tela Vivado,  
Selecione Adicionar ou criar restrições na tela abaixo e pressione o botão Avançar.[](http://www.trenz.jp/wp-content/uploads/2016/05/4-1.png)

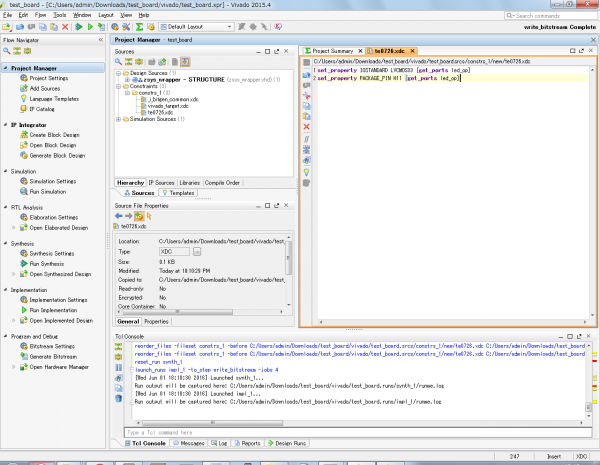
No botão Criar arquivo, crie um arquivo com o nome do arquivo te 0726.[](http://www.trenz.jp/wp-content/uploads/2016/05/4-2.png)

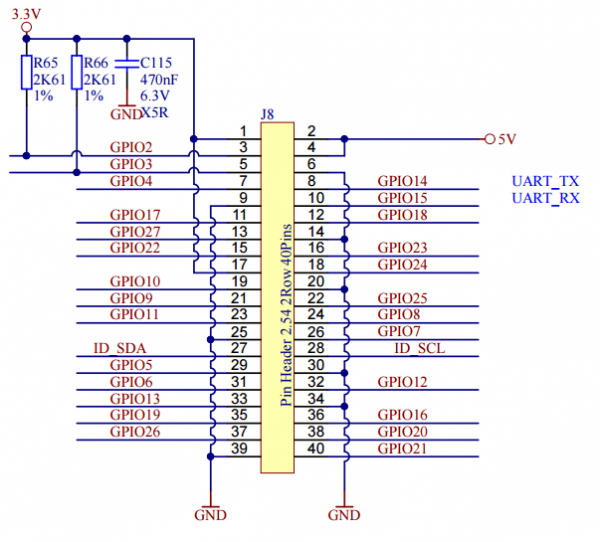
Pressione Terminar.[](http://www.trenz.jp/wp-content/uploads/2016/05/4-3.png)

Clique em Flow Navigator e selecione Restrições -> constrs\_1 -> te0726.xdc na tela Fontes  
Clique duas vezes para abri-lo e edite-o da seguinte maneira.



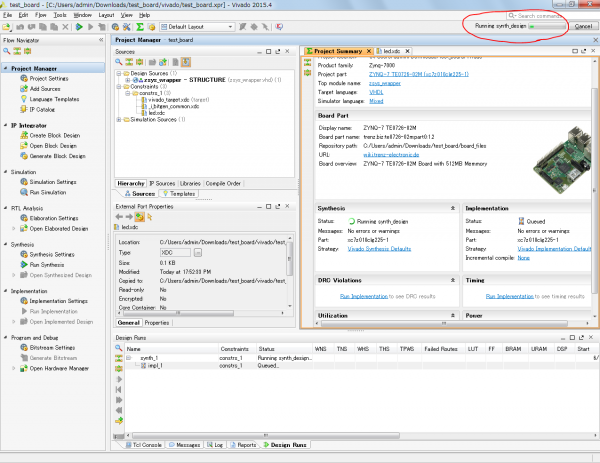
|  |  |
| --- | --- |
| 1  2 | set\_property IOSTANDARD LVCMOS33 [get\_ports led\_op]  set\_property PACKAGE\_PIN H11 [get\_ports led\_op] |

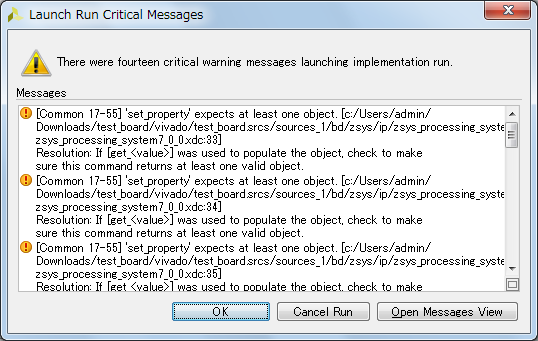
[](http://www.trenz.jp/wp-content/uploads/2016/05/5-11.png)

O pino usado neste momento é H11. Isso corresponde ao GPIO 18 no diagrama de circuito.  
GPIO 18 é o 12º pino IO.  
Para a atribuição de IO, consulte a seguinte parte do diagrama de circuito.  
Além disso, o design de referência do hdmi\_fb possui um arquivo xdc de referência.[](http://www.trenz.jp/wp-content/uploads/2016/05/4-4.png)

Você acabou de editar o arquivo XDC.

**5. Síntese lógica e criação de arquivos de bits.**

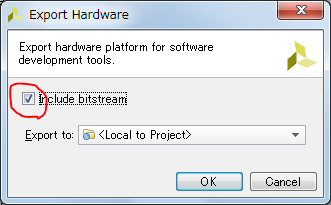
Execute a síntese lógica.  
Basta clicar em Flow Navigator -> Gerar Bitstream no lado esquerdo da tela Vivado.  
O trabalho é feito em segundo plano, mas o status de progresso é exibido no canto superior direito da tela.[](http://www.trenz.jp/wp-content/uploads/2016/05/5-1.png)

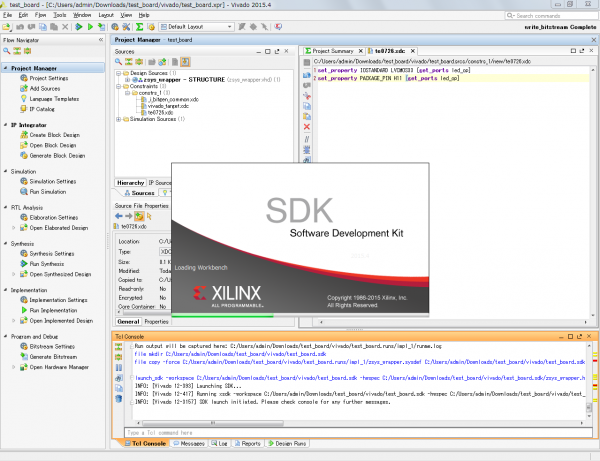
O aviso será exibido, mas não se preocupe e pressione OK.[](http://www.trenz.jp/wp-content/uploads/2016/05/5-2.png)

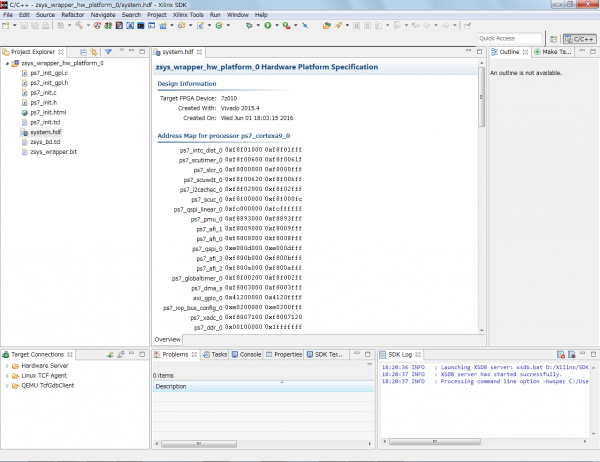
A síntese lógica e a criação de arquivos de bits estão agora concluídas.

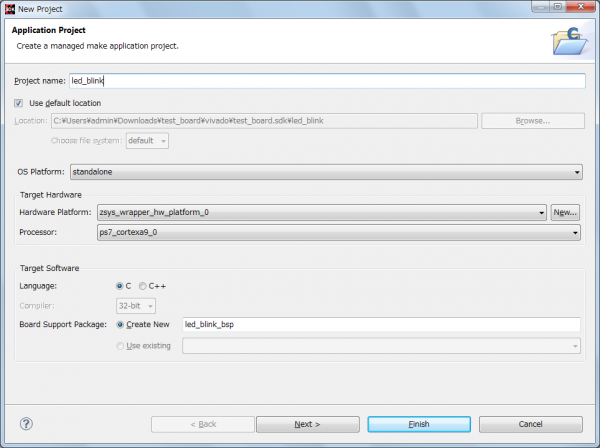
**6. Criação de software**

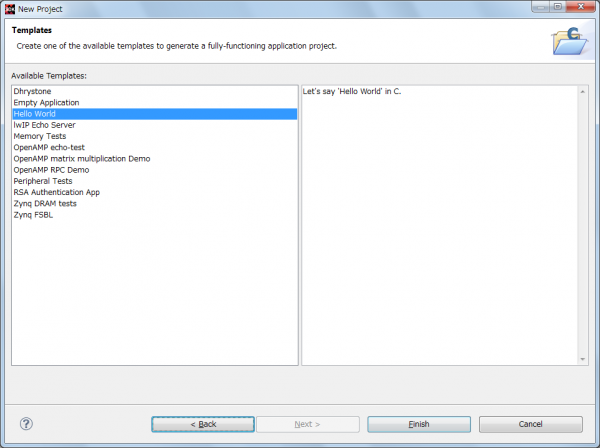
A partir daqui, vou seguir trabalhando com o SDK.  
Neste procedimento, você cria um software que se comunica com o PC e se comunica com a peça PL.  
Em primeiro lugar, execute Export -> Exportar hardware na guia Arquivo do projeto Vivado acima.  
A janela a seguir será exibida. Verifique o fluxo de bits Incluir e pressione o botão OK.

[](http://www.trenz.jp/wp-content/uploads/2016/05/6-1.png)

Execute Iniciar o SDK na guia Arquivo e pressione o botão OK com a janela padrão exibida,  
O SDK será iniciado.[](http://www.trenz.jp/wp-content/uploads/2016/05/6-2.png)

[](http://www.trenz.jp/wp-content/uploads/2016/05/6-3.png)

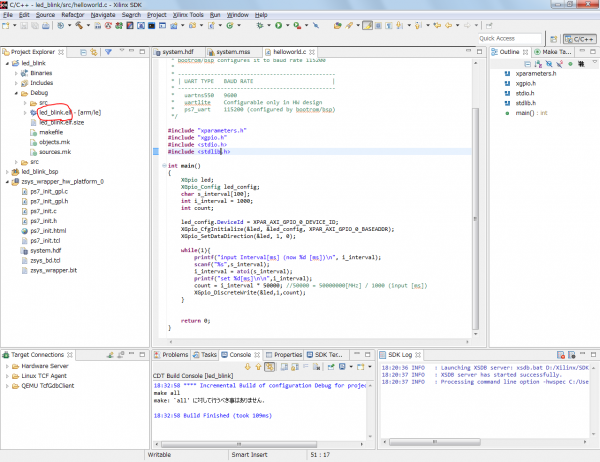
Pressione a guia Arquivo Novo -> Projeto de Aplicação.  
Na seguinte janela a ser exibida, defina Project Name como led\_blink,  
Deixe os outros padrões e pressione o botão Avançar.[](http://www.trenz.jp/wp-content/uploads/2016/05/6-4.png)

Especifique o Olá Mundo em Modelos e pressione o botão Concluir.[](http://www.trenz.jp/wp-content/uploads/2016/05/6-5.png)

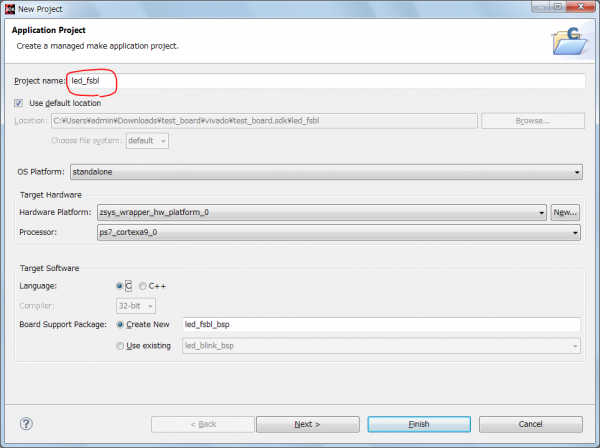
Uma vez que led\_blink e led\_blink\_bsp são adicionados ao Explorador de Projeto esquerdo,  
Clique duas vezes em led\_blink -> src -> helloworld.c e edite seus conteúdos da seguinte forma.

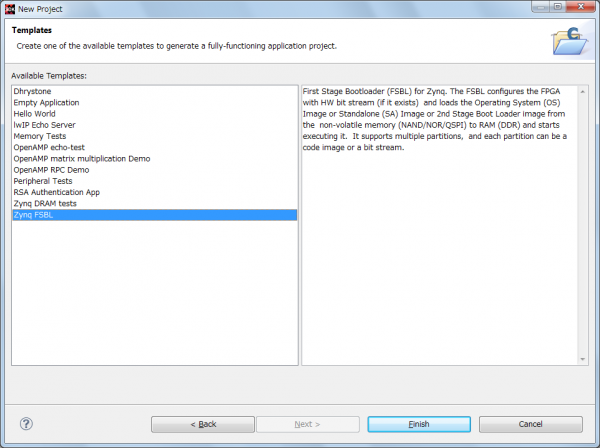


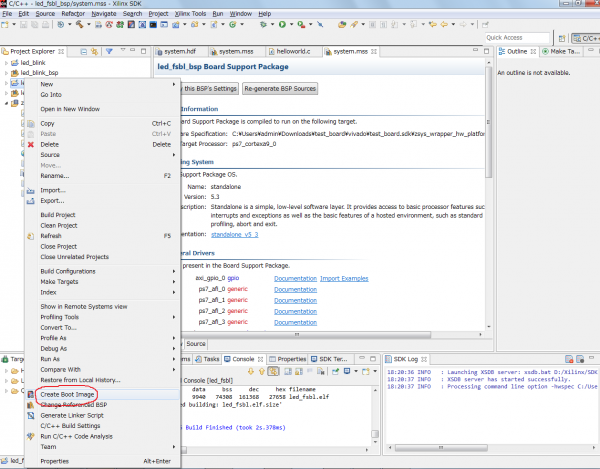
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | #include "xparameters.h"  #include "xgpio.h"  #include <stdio.h>  #include <stdlib.h>    int main()  {      XGpio led;     XGpio\_Config led\_config;     char s\_interval[100];     int i\_interval = 1000;     int count;       led\_config.DeviceId = XPAR\_AXI\_GPIO\_0\_DEVICE\_ID;  XGpio\_CfgInitialize(&led, &led\_config, XPAR\_AXI\_GPIO\_0\_BASEADDR);  XGpio\_SetDataDirection(&led, 1, 0);    while(1){  printf("input Interval[ms] (now %d [ms])\n", i\_interval);  scanf("%s",s\_interval);  i\_interval = atoi(s\_interval);  printf("set %d[ms]\n\n",i\_interval);  count = i\_interval \* 50000; //50000 = 50000000[MHz] / 1000 (input [ms])  XGpio\_DiscreteWrite(&led,1,count);  }          return 0;  } |

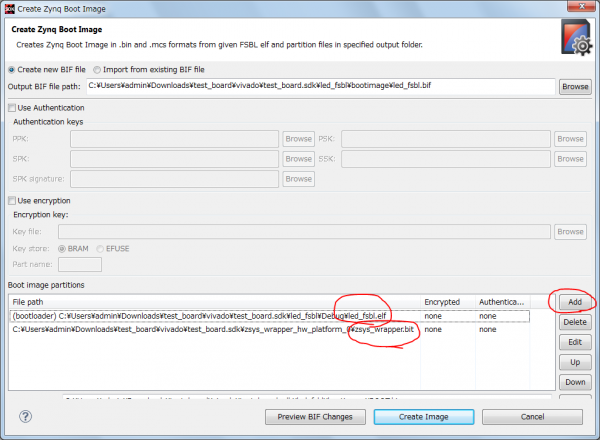
Crie o projeto com ctrl + B.  
O led\_blink criado -> Debug -> led\_blink.elf é um programa para se mover na seção PS.[](http://www.trenz.jp/wp-content/uploads/2016/05/6-6.png)

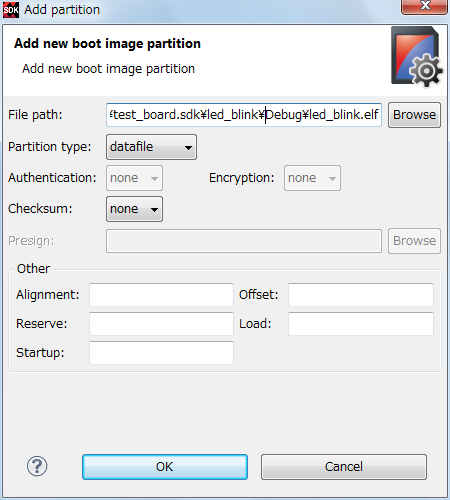
**7. Criar FSBL e Boot.bin**

Em seguida, criarei FSBL e BOOT.bin.  
Na guia Arquivo Novo -> Projeto de Aplicação como na etapa anterior  
Crie um projeto com o nome do Projeto como led\_fsbl e Modelos como Zynq FSBL.[](http://www.trenz.jp/wp-content/uploads/2016/05/7-1.png)

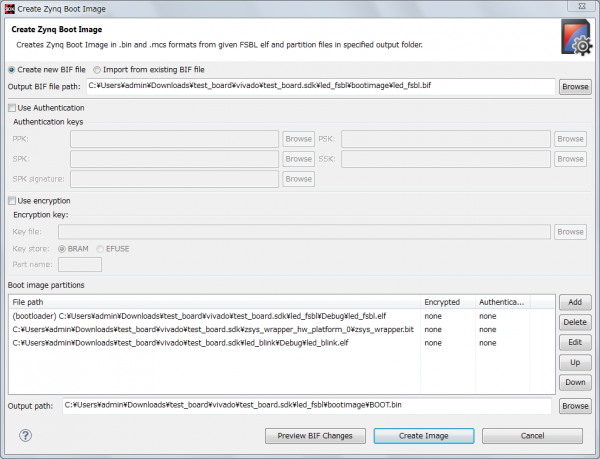
[](http://www.trenz.jp/wp-content/uploads/2016/05/7-2.png)

Aguarde um pouco, clique com o botão direito do mouse no Project Explorer -> led\_fsbl e execute Criar imagem de inicialização.[](http://www.trenz.jp/wp-content/uploads/2016/05/7-3.png)

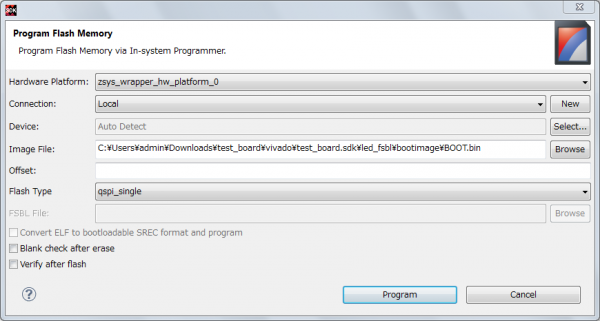
Boot Image contém o arquivo ELS do FSBL e o arquivo de bits criado pelo Vivado no estado inicial.  
Adicione led\_blink.elf que você acabou de editar a partir do botão Adicionar aqui.  
O caminho padrão para led\_blink.elf é  
test\_board / vivado / test\_board.sdk / led\_blink / Debug.[](http://www.trenz.jp/wp-content/uploads/2016/05/7-4.png)

[](http://www.trenz.jp/wp-content/uploads/2016/05/7-5.png)

Pressione o botão Criar inicialização para criar BOOT.bin.

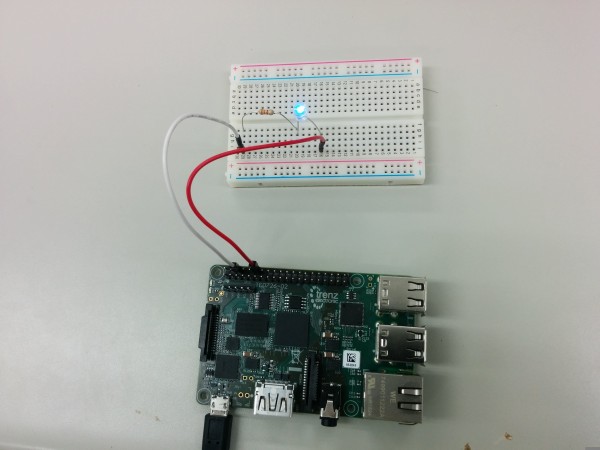
[](http://www.trenz.jp/wp-content/uploads/2016/05/7-6.png)

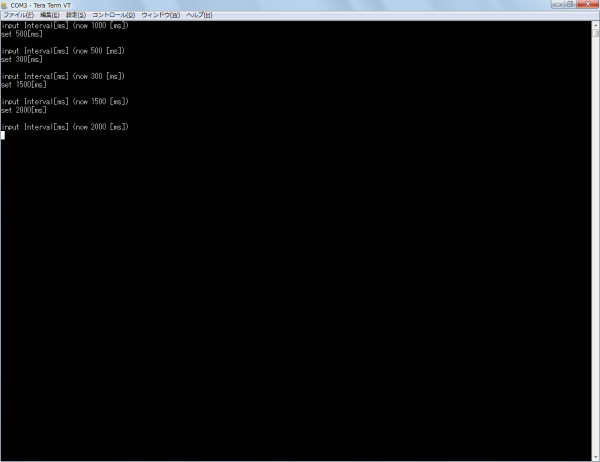
**8. Flash ROM escrever**

Abra a janela do Programa Flash na guia Xilinx Tools do SDK,  
No arquivo de imagem, especifique BOOT.bin criado anteriormente.  
O caminho padrão é test\_board / vivado / test\_board.sdk / led\_fsbl / bootimage /.  
Quando você pressiona o botão Programa, BOOT.bin é gravado na ROM flash.[](http://www.trenz.jp/wp-content/uploads/2016/05/8-1.png)

**9. O LED acende-se!**

Conecte o LED e o resistor.  
Uma vez que a saída foi ajustada para o pino 12, o lado ânodo do LED foi conectado ao pino 12,

O lado do catodo está conectado ao pino 6 conectado ao GND.  
Conecte e desconecte o USB, ou como o Hello World  
Quando a reinicialização do LED XMD se acenderá](http://www.trenz.jp/wp-content/uploads/2016/05/8-2.jpg)

Além disso, é possível definir o ciclo no qual a comunicação serial é realizada a partir de Teraterm etc. para piscar.[](http://www.trenz.jp/wp-content/uploads/2016/05/9-1.png)