UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

COORDENAÇÃO DE ENGENHARIA ELETRÔNICA

CURSO DE ENGENHARIA ELETRÔNICA

ALMIR AUGUSTO BRAGGIO

FPGA: PROCESSO DE CONFIGURAÇÃO COM DESENVOLVIMENTO DE PROTÓTIPO EM PLACA DE CIRCUITO IMPRESSO

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO

2014

ALMIR AUGUSTO BRAGGIO

FPGA: PROCESSO DE CONFIGURAÇÃO COM DESENVOLVIMENTO DE PROTÓTIPO EM PLACA DE CIRCUITO IMPRESSO

Trabalho de Conclusão de Curso de Graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2 (TCC2), como requisito parcial para obtenção do título de Engenheiro. Eletrônico, da Coordenação do Curso de Engenharia Eletrônica, da Universidade Tecnológica Federal do Paraná – UTFPR, Campus Toledo.

Orientador: Prof. Dr. Paulo de Tarso Neves Júnior

TOLEDO

2014

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Toledo
Coordenação do Curso de Engenharia Eletrônica

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nº 003

FPGA: Processo de configuração com desenvolvimento de protótipo em placa de circuito impresso

por

Almir Augusto Braggio

Esse Trabalho de Conclusão de Curso foi apresentado às 10:00h do dia **21 de fevereiro de 2014** como requisito parcial para a obtenção do título **Bacharel em Engenharia Eletrônica**. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado **APROVADO**.

Prof. Dr. Alberto Yoshihiro Nakano
(UTFPR-TD)

Prof. Dr. Felipe Walter Dafico Pfrimer
(UTFPR-TD)

Prof. Dr. Paulo de Tarso Neves Junior
(UTFPR-TD)
Orientador

Visto da Coordenação

Prof. M. Rodrigo da Ponte Caun
Coordenador da COELE

O Termo de Aprovação assinado encontra-se na Coordenação do Curso

AGRADECIMENTOS

Agradeço a quem ficaria feliz a me ver escrever isso. Aqui há dedicação somada à motivação.

Agradeço meu pai, Almir Ricardo, e à minha mãe, Ana Lúcia. Às minhas irmãs, Ana Karine e Ana Larisse. Ao meu primeiro sobrinho, Pedro.

A minha namorada, Gabriela.

Aos meus amigos. Àqueles que me cederam um tempo para conversar, trocar ideias e me estimular de alguma maneira.

A todos com quem convivi durante todos os anos de faculdade.

Ao meu orientador Prof. Dr. Paulo de Tarso Neves Júnior e a todos que me influenciaram e ajudaram.

Também agradeço a você, que está lendo isso. Os agradecimentos de um trabalho são sempre sinceros.

“Agora é tarde demais para ser reprovado...”

Almir Augusto Braggio

.

“Earn this. Earn it...” *(Faça por merecer)*
(John Miller – Saving Private Ryan, 1998)

RESUMO

BRAGGIO, Almir A. **FPGA: Processo de Configuração com Desenvolvimento de Protótipo em Placa de Circuito Impresso.** 2014. 63f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica) - Universidade Tecnológica Federal do Paraná. Toledo, 2014

Um FPGA (*Field Programmable Gate Array*) é um dispositivo lógico programável geralmente baseado em memória volátil que necessita de uma configuração basicamente descrita por um editor HDL (*Hardware Description Language*) e que garantirá o seu correto funcionamento. Neste trabalho, objetiva-se responder questões básicas relacionadas a esta configuração, na perspectiva de hardware, como modos e esquemas de configuração (ativo e passivo), pinos de configuração e soluções existentes (memórias), de forma a gerar um conhecimento direcionado para o desenvolvimento de uma aplicação simples e funcional de uma PCB (*Printed Circuit Board*). Prevê-se a construção de uma placa com FPGA *Altera*® contendo duas formas de configuração (JTAG e modo ativo serial) e funcionalidades simples, semelhantes a kits básicos de desenvolvimento existentes.

Palavras-chave: FPGA, PLD, PCB, Configuração, Altera.

ABSTRACT

BRAGGIO, Almir A. **FPGA: Configuration Process with Prototype Development in Printed Circuit Board.** 2014. 63f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica) - Federal University of Technology - Paraná. Toledo, 2014.

A FPGA (*Field Programmable Gate Array*) is a programmable logic device usually based on volatile memory that requires a configuration basically described by an HDL editor (*Hardware Description Language*) and ensure its correct operation. In this study, the objective is to answer basic questions related to the configuration, from the perspective of hardware, such as schemas and configuration modes (active and passive), pins configuration and solutions (memories) in order to generate knowledge directed to the development of a simple and practical application of a PCB (*Printed Circuit Board*). It's expected to build a board with *Altera® FPGA* containing two forms of configuration (JTAG and Active Serial mode) and simple features, similar to basic kits of development already existed.

Keywords: FPGA, PLD, PCB, Configuration, Altera.

LISTA DE FIGURAS E ILUSTRAÇÕES

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

AP	*Active Parallel*
AS	*Active Serial*
ARM	*Advanced RISC Machine*
ASSP	*Application Specific Standard Product*
ASIC	*Application-Specific Integrated Circuit*
CI	*Circuito integrado*
CPLD	*Complex PLD (Programmable Logic Device)*
CAD	*Computer Aided Design*
CLB	*Configurable Logic Blocks*
DSP	*Digital Signal Processor*
FPGA	*Field Programmable Gate Array*
GLS	*Gate Level Simulation*
GPIO	*General Purpose Input/Output*
HDL	*Hardware Description Language*
ICSP	*In-Circuit Serial Programming*
ISP	*In-System-Programming*
JTAG	*Joint Test Action Group*
LE	*Logic Elements*
MCU	*Micro Controller Unit*
NRE	*Non-recurring engineering*
PP	*Passive Parallel*
PS	*Passive Serial*
PCB	*Printed Circuit Board*
PLD	*Programmable Logic Device*
PoR	*Power On Reset*
RAM	*Random Access Memory*
RTL	*Register Transfer Level*
SPI	*Serial Peripheral Interface*
SPLD	*Simple PLD*
SRAM	*Static Random Access Memory*
SoC	*System-on-Chip*
VHDL	*Very High Speed Integrated Circuit Hardware Description Language*

SUMÁRIO

1 INTRODUÇÃO

Um FPGA, do inglês *Field Programmable Gate Array*, é um dispositivo lógico programável, geralmente baseado em memória volátil onde sua configuração deve ser carregada a cada uso do dispositivo.

Esta configuração serve para inicializar e controlar corretamente o dispositivo, como conexões de roteamento e outras funções programáveis, conforme fora feito na descrição do projeto implementado, como no uso de um editor HDL, do inglês *Hardware Description Language*.

A configuração pode ser ativa, onde o FPGA controla o processo obtendo os dados necessários através de um dispositivo de memória externa, ou passiva, onde um dispositivo externo controla todo o processo fornecendo estes dados.

A imagem de configuração a ser carregada na RAM (*Random Access Memory*) pode ser inicializada a partir de uma conexão JTAG (*Joint Test Action Group*) ou de uma memória não volátil, como a FLASH[1], por exemplo. Estas são as configurações mais usadas, mas também é possível implementar uma configuração por TCP/IP ou Ethernet, por exemplo, ou configurar o dispositivo através de um sistema embarcado e/ou microcontrolado.

Em todos os casos, na construção de uma placa de circuito impresso (PCB - *Printed Circuit Board*), o hardware com o FPGA deve ser estudado de forma que a sua configuração funcione corretamente de acordo com as ferramentas de desenvolvimento e programação existentes.

1.1 MOTIVAÇÃO E JUSTIFICATIVA

O alto e crescente desempenho destes dispositivos relacionado à queda dos custos e as amplas vantagens de uso somadas a facilidade de se implementar qualquer coisa, de complexos dispositivos de comunicação à computação

[1] Flash Memory: memória não-volátil de acesso rápido do tipo EEPROM (*Electrically-Erasable Programmable Read-Only Memory*), comumente utilizada em cartões de memória, por exemplo, e outros dispositivos de armazenamento.

reconfigurável, faz dos FPGAs atrativos dispositivos de desenvolvimento. Segundo MILLER:

> Os FPGAs estão encontrando mercado para uma agressiva expansão. Eles estão se movendo gradualmente em direção a dominar toda a PCB. [...] Você poderia ter um único dispositivo que pode substituir uma série de combinações e periféricos de MCUs[2]. Você poderia até mesmo acelerar uma parte de seu algoritmo usando lógica combinacional. (MILLER, 2013, Tradução)

Assim, tomando como base as estimadas perspectivas de crescimento, mesmo com um relativo número de ferramentas e kits de desenvolvimento disponíveis no mercado, parece inevitável a importância à informação de como proceder para configurar um FPGA, das técnicas e esquemas à aplicação real.

Este trabalho, então, também é voltado ao desenvolvimento de projetos dedicados com FPGAs e aos projetistas que trabalham com kits de desenvolvimento sem o conhecimento do fundamento para a construção de suas próprias PCBs.

1.2 OBJETIVOS

Por que configurar? Como configurar? Quais os passos? Responder perguntas básicas relacionadas à configuração de FPGAs é o grande objetivo deste trabalho.

Além disso, construir um *know-how* aberto e funcional não apenas aos que estão iniciando na área de dispositivos lógicos programáveis como também aos projetistas para a construção de suas próprias PCBs, é um estímulo impactante e motivador.

Objetiva-se, então, reunir um conhecimento processual para desenvolver uma aplicação com um FPGA, de forma a introduzir os modos e esquemas de configuração possíveis, pinos de configuração, sequência de configuração, soluções

---

[2] MCU, do inglês *Micro Controller Unit*, por vezes referido como *Microprocessor Controller Unit*, é um circuito integrado que pode ser programado para funções específicas, contendo memória, processador e periféricos de entrada e saída.

para configuração, para, enfim, desenvolver um protótipo como exemplo de aplicação, apresentando uma PCB simples e usual, que funcione com aplicações de software disponíveis (destacar-se-á o estudo e aplicação de um dispositivo da marca *Altera*® e software relacionado, *Altera*® *Quartus II*).

2 DESENVOLVIMENTO

Um FPGA é um dispositivo lógico programável, apresentado como circuito integrado, ou CI[3], que contém matrizes de blocos lógicos, com interconexões configuráveis, chamados CLBs (*Configurable Logic Blocks)*, organizados de tal maneira que um desenvolvedor possa programá-los para executar uma ampla gama de tarefas (MAXFIELD, 2004).

Existem diversos tipos de CIs digitais, que são basicamente constituídos de portas lógicas (AND, OR, NOT, entre outras), e é a estruturação dessas portas que constitui a eletrônica digital. Estas portas podem ser construídas de diversas maneiras, dado seu processo de fabricação e diferentes arquiteturas, como a lógica TTL (*Transistor-Transistor Logic*) ou CMOS (*Complementary Metal-Oxide Semiconductor*) (COSTA, 2009).

De modo geral, podem-se caracterizar os CIs digitais em dispositivos de lógica fixa e de lógica programável (sem entrar no mérito de escala de integração ou encapsulamento).

Os dispositivos de lógica fixa, como o nome sugere, são os que têm função lógica definida em sua fabricação, sem possibilidade de alteração. São os ASICs (*Application-Specific Integrated Circuits*) e os ASSPs (*Application Specific Standard Products*). Os de lógica programável, os PLDs (*Programmable Logic Devices*), permitem um desenvolvimento dinâmico (MÉLO; DIAS; STEINBACH, 2011). Os PLD's, podem se dividir em SPLD's (*Simple PLD*), CPLDs (*Complex PLD*) e FPGAs.

Aqui, é necessário saber que a arquitetura interna dos PLDs é pré-determinada pelo fabricante, mas é disposta de tal modo que estes podem ser configurados. E, tratando-se dos ASICs e ASSPs, estes são construídos para executar funções específicas, mas não podem ser configurados. Um ASIC é projetado sob encomenda, e um ASSP é comercializado para quaisquer clientes interessados em sua função.

Embora os dispositivos de aplicação específica possam oferecer o máximo desempenho e grande complexidade, projetar e construir um circuito integrado deste

---

[3] Circuito integrado, ou CI, é todo circuito eletrônico miniaturizado em substrato semicondutor.

tipo é um processo demorado e caro, com a desvantagem de que o projeto final não pode ser modificado sem a criação de uma nova versão. Estes são “concebidos e construídos, desde a sua máscara de silício, com uma finalidade pré-definida, imutável (...) têm um altíssimo custo de desenvolvimento, com utilização justificada apenas para uma produção em larga escala.” (MÉLO; DIAS; STEINBACH, 2011)

Desta maneira, os PLDs, que podem ser personalizados, têm seu diferencial: a execução e construção do design da aplicação são mais fáceis e baratos.

Assim, é perceptível que CIs configuráveis, como os FPGAs, permitem que desenvolvedores individuais ou pequenas empresas realizem projetos de hardware de alto nível sem ter de arcar com grandes investimentos (NRE[4]), ou comprar ferramentas caras e complexas associadas a projetos ASIC.

Historicamente, quando os FPGAs chegaram ao mercado, em meados de 1980, as implementações exigiam uma média complexidade e o processamento era relativamente limitado. Já na década de 1990, com a sofisticação destes dispositivos, seu uso teve um crescimento enorme, principalmente na área automotiva e industrial, mesmo que, geralmente, utilizados para prototipagem de circuitos ASIC.

No entanto, seu baixo custo de desenvolvimento e diminuição de *time-to-market*[5], permitiu uma competição direta com outros dispositivos. No início da década de 2000, os FPGAs já possuíam alto desempenho, inclusive com outras funcionalidades embutidas, como núcleos de microprocessadores e alto número de interfaces de I/O (*input/output*), por exemplo, principalmente se comparado à microcontroladores.

O resultado final é que os FPGAs existentes hoje, podem ser usados para implementar qualquer coisa, inclusive complexos dispositivos de comunicação,

---

[4] *NRE: Non-Recurring Engineering*, refere-se ao custo de uma unidade de tempo de pesquisa, desenvolvimento, design e teste de um novo produto. É diferente dos custos de produção (*COP: Cost Of Production*) e deve ser considerado na análise da rentabilidade de um produto.

[5] *Time-to-Market*: tempo de lançamento de um produto, do desenvolvimento à disponibilidade para venda.

dispositivos para processamento de sinais e imagens, e até mesmo SoCs (*System-on-Chip*[6]).

Os FPGAs são usados, por exemplo, para construir DSPs (*Digital Signal Processor*) de forma barata, a ponto de superar os mais velozes processadores de sinais digitais existentes (devido à fácil implementação de circuitos multiplicadores e de aritmética dedicada, dado que o desenvolvedor o faz voltado para sua aplicação), além da lógica combinacional (MAXFIELD, 2004).

Outro uso recorrente destes dispositivos é na incorporação de microcontroladores, adaptando seu circuito programável com funções de controle de portas I/O, núcleos de processadores, temporizadores e periféricos, associando a queda nos preços dos FPGAs a aplicações robustas de controle embarcado, com a atração da personalização pelo desenvolvedor.

Além disso, os FPGAs têm criado uma nova área: computação reconfigurável. Trata-se da ideia de combinar o desempenho do hardware com a flexibilidade do software. Com ela, é possível explorar o paralelismo inerente do hardware, executando algoritmos muito mais rapidamente que os processadores que obedecem ao modelo de *von Neumann*[7].

2.1 CONFIGURAÇÃO

A grande questão de se definir um processo de configuração de FPGAs é que cada fabricante tem sua própria terminologia (nome e número de pinos, por exemplo) e suas próprias técnicas ou protocolos para o processo de configuração. E para dificultar a vida dos desenvolvedores, os FPGAs podem variar de família ou série, dentro de uma única marca.

A seguir, a título de informação, um levantamento de algumas das maiores marcas de dispositivos atualmente disponíveis no mercado:

---

[6] *SoC: System-on-Chip*, refere-se a todos os componentes de um sistema eletrônico em um circuito integrado, como microprocessador, memória e interfaces de entrada e saída, por exemplo. Típica aplicação na área de sistemas embarcados.

[7] *John Von Neumann* propôs a arquitetura de computador mais conhecida, onde um programa utiliza uma unidade central de processamento (CPU) e uma de armazenamento (memória) para comportar instruções e dados, executando-o de forma sequencial.

- *Achronix™ (achronix.com), 2*004, Santa Clara, California, United States;
- *Altera™ (altera.com),* 1983, San Jose, California, United States;
- *Atmel™ (atmel.com), 1*984, San Jose, California, United States;
- *Lattice™ (latticesemi.com),* 1983, Hillsboro, Oregon, United States;
- *Microsemi™ (microsemi.com),* 1960, Aliso Viejo, California, United States;
- *QuickLogic™ (microsemi.com),* 1988, Sunnyvale, California, United States;
- *Tabula™ (tabula.com),* 2003, Santa Clara, California, United States;
- *Xilinx™ (xilinx.com),* 1984, San Jose, California, United States.

Por estas razões, levantada o número de fabricantes e considerando um relativo número de dispositivos existentes, a discussão da configuração de FPGAs neste trabalho privar-se-á por uma introdução genérica, porém, focando nos dispositivos da empresa *Altera*®. Motivo pelo qual já se trabalha com as ferramentas de desenvolvimento (software *Altera Quartus® II*) dessa fabricante.

Mas, o que é configuração e por que configurar FPGAs? A maior parte dos dispositivos é baseada em SRAM (*Static Random Access Memory*) e durante seu funcionamento os dados de configuração, que servem para inicializar e controlar corretamente o dispositivo, como conexões de roteamento e outras funções programáveis, são armazenados nessa memória. Porém, a memória SRAM é volátil, ou seja, a cada novo uso do dispositivo, os dados devem ser recarregados.

É importante ressaltar que são diversas as etapas anteriores à configuração do FPGA. Da descrição do projeto à sua configuração, as etapas, de modo geral, podem ser listadas como:

1) Especificação ou descrição do projeto;
2) Simulação em nível RTL (*Register Transfer Level*) e *test benches*[8];
3) *Netlisting*[9] (síntese e mapeamento);
4) *Place & route* (posicionamento e roteamento);
5) Verificação e testes;

---

[8] *Test bench*, ou teste de bancada, descreve um processo de testes usado para validar um modelo ou um software.
[9] *Netlist* pode ser uma conexão física ou lógica que descreva uma ligação eletrônica.

6) Simulação em nível de portas GLS (*Gate Level Simulation*);
7) Programação/Configuração da FPGA;

Várias destas etapas são, geralmente, automatizadas com uso de software de design. Estes softwares são chamados de CAD (*Computer Aided Design*), que fazem com que seja fácil programar um circuito lógico usando um dispositivo lógico programável (ALTERA, 2005). Como o software *Altera Quartus® II*.

A descrição do projeto pode ser realizada de duas formas: diagrama lógico, a partir de um editor gráfico com uso de portas lógicas e macro instruções, ou através de um editor HDL (COSTA, 2009). E diversas são as ferramentas; por exemplo, algumas linguagens de programação usuais: ABEL (*Advanced Boolean Equation Language*); VHDL (*Very High Speed Integrated Circuit Hardware Description Language*); Verilog; AHDL (*Altera Hardware Description Language*); JHDL (*Java Hardware Description Language*), entre outras.

As linguagens VHDL e Verilog são as mais utilizadas pela comunidade de desenvolvedores. A Figura 1 exibe o espaço de trabalho de um projeto em VHDL em desenvolvimento com o uso da ferramenta *Quartus® II*.

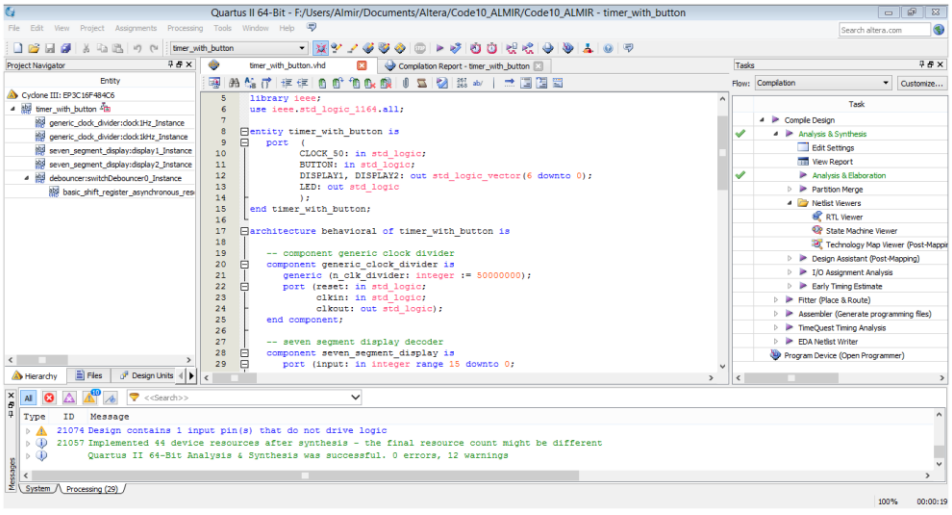


Figura 1 - Workspace do software de design Quartus II

A simulação em nível RTL é a simulação funcional da descrição de hardware, ou seja, sem envolver hardware, desprezando atrasos inerentes de chaveamento, por exemplo. Trata-se de uma verificação lógica do fluxo de dados.

As figuras 2 e 3 exibem o resultado de uma simulação e a visualização do modelo RTL do projeto.

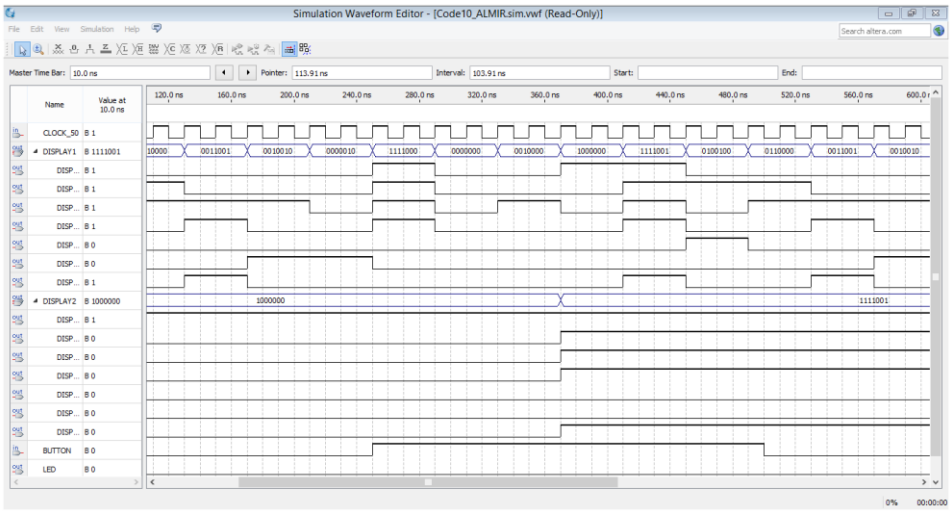


Figura 2 - Resultado da simulação

É importante saber que o modelo RTL é uma descrição síncrona do circuito, usada para demonstrar o fluxo do sinal, funcionando como uma abstração da descrição VHDL na representação do circuito.

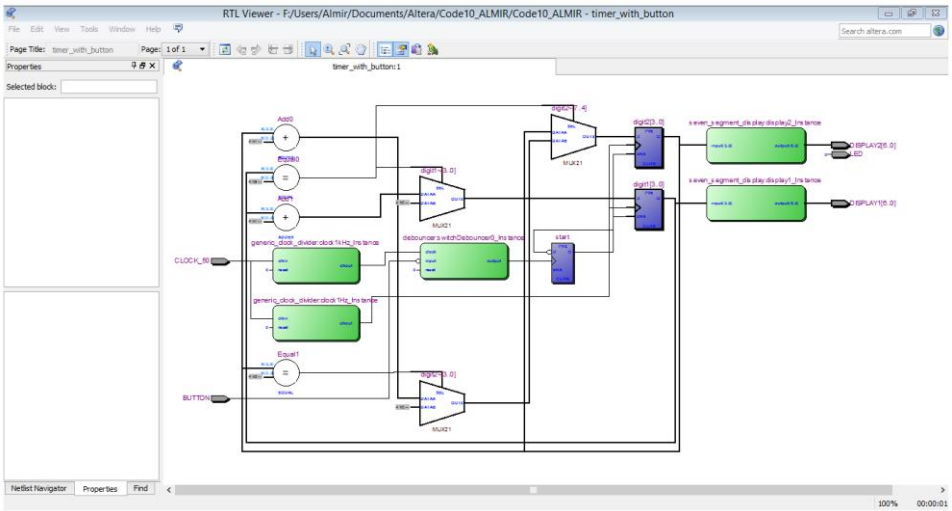


Figura 3 - Visualização em nível RTL

A síntese lógica consiste na otimização, minimizando as expressões booleanas e otimizando o circuito, para o mapeamento posterior, que seleciona um conjunto de portas para reduzir a área ocupada no dispositivo, levando em conta as

restrições arquiteturais. O resultado do *netlisting* pós-mapeamento pode ser visto na Figura 4.

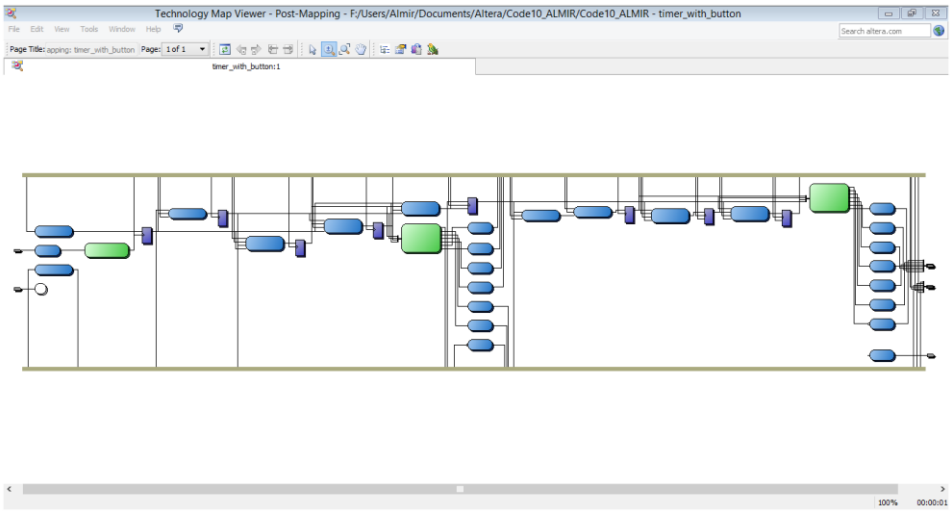


Figura 4 - Visualização *netlist post-mapping*

Após a minimização lógica e o mapeamento, a descrição do projeto se torna uma representação da designação dos componentes lógicos aos componentes físicos, e este é o processo de posicionamento e roteamento (*place & route*).

O posicionamento atribui os componentes do hardware aos blocos lógicos funcionais do sistema e o roteamento atribui às trilhas e elementos programáveis, a interconexão dos blocos lógicos, de forma a maximizar a velocidade e diminuir o custo do sistema.

A visualização da área em uso do dispositivo, chamada *Chip Planner*, pode ser entendida na Figura 5. A visualização dos pinos em uso, conhecida como *Pin Planner*, Figura 6, também é uma maneira interessante de se verificar o comportamento físico no dispositivo.

A etapa de verificação e teste consiste na análise de tempos, de potência e ruídos (SSN – *Simultaneous Switching Noise).* A etapa de simulação em nível de portas (GLS) é opcional, mas é nela que são levadas em conta as possíveis restrições de temporização, por exemplo, de forma específica para cada projeto. São nestas duas etapas que atrasos de propagação dos vários circuitos e trajetos são analisados, de modo a obter uma indicação de desempenho do circuito.

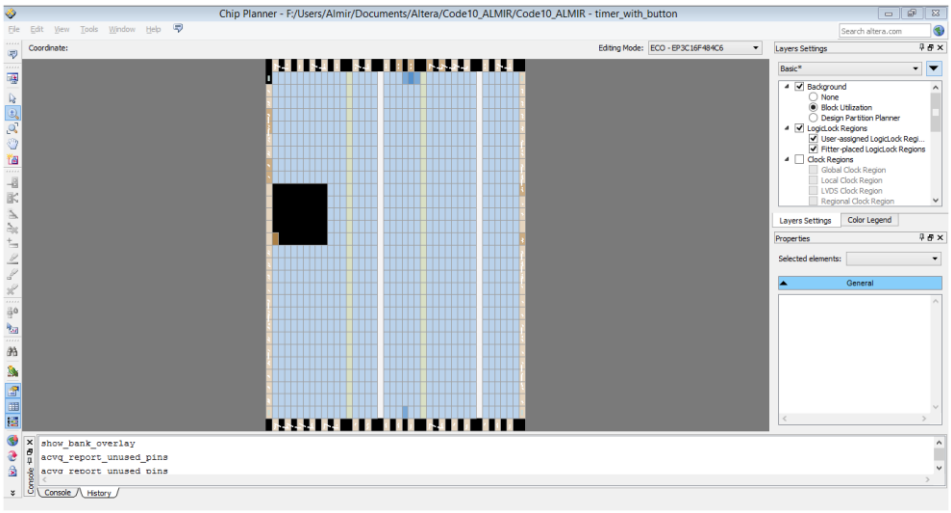


Figura 5 - Chip Planner: área em uso no dispositivo

Figura 6 - Pin planner: disposição dos pinos no dispositivo (encapsulamento BGA)

As etapas de projeto podem seguir um fluxograma, como o descrito na Figura 7. O resultado final é um arquivo de configuração que contém todas as informações que serão enviadas para o FPGA, a fim de configurá-lo para executar o que foi descrito durante as etapas discutidas (ALTERA, 2005).

O arquivo contém um conjunto de dados (bits que são usados para definir o estado dos elementos lógicos) e comandos de configuração (instruções ao dispositivo).

Diversos são os formatos usados (2.1.5 Arquivos de configuração), por exemplo: SOF (*SRAM Object File*), POF (*Programmer Object File*), RBF (*Raw Binary File*), RPD (*Raw Programming Data File*), HEX ou HEXOUT (*Hexadecimal Format File*), TTF (*Tabular Text File*), JAM (*Jam File*), JBC (*Jam Byte-Code File*). Quando o

arquivo está pronto para ser transferido, a informação é referida como fluxo de dados de configuração (ALTERA, 2005).

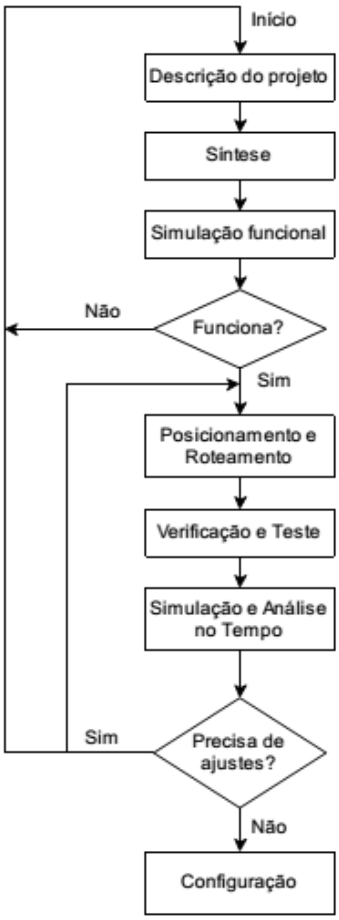


Figura 7 - Fluxograma das etapas de projeto

Os dispositivos baseados em memória estática, como memória Flash, são programados de forma semelhante aos baseados em SRAM e o conceito é simples. Aqui, não será detalhado como é o processo interno ao dispositivo, mas, genericamente, supondo que o dispositivo possua uma única matriz de blocos lógicos programáveis, como na Figura 8, com suas interconexões - contendo um grande número de células de memória associadas, cada célula de memória pode ser vista como uma abstração da configuração do dispositivo. Por exemplo, supondo que cada bloco lógico necessite de 16 bits de configuração, seria necessário uma célula de memória de 16 *latches* (MAXFIELD, 2004).

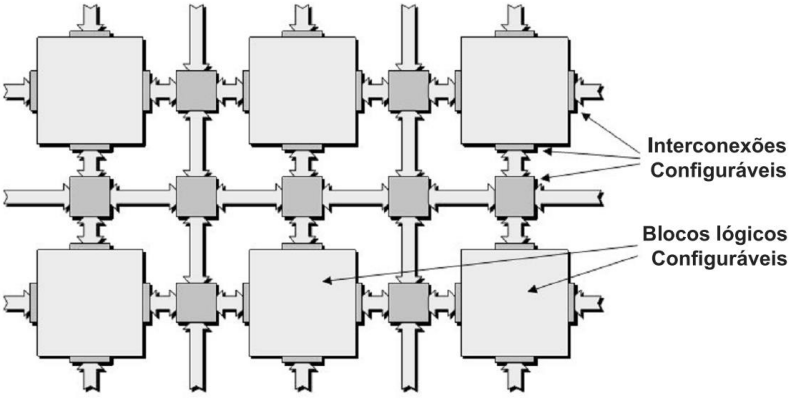


Figura 8 - Vista de uma arquitetura FPGA simples
Fonte: The Design Warrior's Guide to FPGA – MAXFIELD

Como um FPGA contém milhões de células, é interessante saber que os fabricantes dividem os dispositivos, internamente, em grupos de *flip-flops*, denominados *frames*, que compartilham um mesmo *clock* e fazem a configuração das células de memória como um *shift-register*, onde cada *flip-flop* configura grupos de células também dispostas em *frames* (MAXFIELD, 2004).

A partir desta introdução genérica referente à programação de FPGAs, podem-se detalhar os pinos e modos de configuração dos dispositivos.

2.1.1 Pinos de configuração

Tratando-se dos pinos, ou portas, de configuração, os nomes e números de pinos variam com relação à técnica empregada (serial, paralela, entre outras) e de fabricante para fabricante. Entretanto, a maioria apresenta pinos para seleção do modo de configuração, pinos para indicação de início de fim do processo, indicador de erro, e entrada e saída serial de dados.

A *Xilinx*®, por exemplo, apresenta os seguintes pinos de configuração: PROGRAM_B, CCLK, INIT_B, DONE, DIN, DOUT, entre outros pinos para modos específicos, além de pinos para seleção do modo de configuração.

Neste caso, numa configuração qualquer, além de utilizar os pinos de modo de configuração, um pino adicional é utilizado para instruir o dispositivo para iniciar a configuração, enquanto outro indica quando o processo termina. A configuração também pode ser reiniciada quando há erros, por exemplo.

Já a *Altera*® apresenta seus pinos com a seguinte nomenclatura: MSEL, nCONFIG, nSTATUS, CONF_DONE, DCLK, nCE, e outros pinos usados em modos específicos (como os pinos JTAG - item 2.1.2.3). A abordagem destes pinos no circuito será discutida adiante, concomitante com os modos ou esquemas de configuração. A partir deste ponto, só serão discutidos pinos com modos de configuração do fabricante *Altera*®.

2.1.2 Modos e esquemas de configuração

De modo geral, a configuração de FPGAs pode ser ativa, onde o FPGA controla todo o processo obtendo os dados necessários através de um dispositivo de memória externa, ou passiva, onde um dispositivo externo controla todo o processo fornecendo estes dados. Muitas vezes, o modo JTAG (*Joint Test Action Group*), modo passivo de configuração, é referido como uma terceira forma de configuração.

O fluxo de dados de configuração a ser carregado na RAM pode ser inicializado a partir de uma conexão JTAG ou de uma memória FLASH, por exemplo. Estas são as configurações mais indicadas e usuais, mas também é possível programar uma configuração por TCP/IP, por exemplo, ou configurar o dispositivo através de um sistema embarcado (MAXFIELD, 2004).

São várias as possibilidades, e dependendo da estratégia abordada, a configuração pode ser completa (*full-chip configuration*) ou parcial (*partial configuration*).

Na configuração completa, o dispositivo é configurado durante a inicialização (*power-up*), para então começar a funcionar, mas também é possível reconfigurar o dispositivo a qualquer momento, como no caso de uma atualização, por exemplo. Durante esse tipo de configuração, no *power-up* ou durante uma atualização, o sistema não funciona, passando a funcionar apenas após completar o processo (2.1.3 Sequência de configuração).

Durante a configuração parcial, ou reconfiguração parcial, a configuração completa é realizada em *power-up*, mas, depois, o dispositivo permite ser reconfigurado, em algumas áreas do núcleo, enquanto outras áreas continuam a funcionar. Essa forma permite, por exemplo, atualizar o sistema sem interromper a

operação de outros componentes, mas, logicamente, trata-se de um processo mais complexo (MAXFIELD, 2004).

Como citado, o fluxo de dados de configuração pode ser carregado em FPGAs utilizando um esquema ativo ou passivo. Os FPGAs que permitem o modo ativo vêm com uma máquina de estados pré-programada, não volátil, que controla a configuração. No esquema passivo, esta máquina de estados é desativada e um dispositivo externo deve controlar o processo de configuração. Além disso, quando se utiliza um esquema passivo, a reconfiguração é possível, mesmo quando o sistema estiver em funcionamento (ALTERA, 2013).

Os dispositivos FPGAs *Altera*® possibilitam uma série de esquemas de configuração. Depois de se definir um esquema de configuração apropriado para sua aplicação, o desenvolvedor deve configurar o modo escolhido através dos pinos de controle (*MSEL*), para definir o modo de configuração, e muitos são os esquemas:

a) *Active Serial* (AS) (ou "*Master Serial*");
b) *Active Parallel* (AP) (ou "*Master Parallel*");
c) *Passive Serial* (PS) (ou "*Slave Serial*");
d) *Passive Parallel* (PP) (ou "*Slave Parallel*"):
    a. *Fast Passive Parallel* (FPP)
    b. *Passive Parallel Synchronous* (PPS)
    c. *Passive Parallel Assynchronous* (PPA)
e) JTAG (*Joint Test Action Group*);
f) *Passive Configuration via Protocol* (CvP).

2.1.2.1 Esquema de configuração ativo

O modo ativo pode ser serial (*AS*, *Active Serial*) ou paralelo (*AP, Active Parallel*). No modo AS, os dados podem ser carregados de 1 bit ou 4 bits por vez. A *Altera* fornece um dispositivo de memória avançado (EPC[10]), EPCS (S: *single output bit*) ou EPCQ (Q: *quad output bit*), do tipo FLASH, de interface SPI (*Serial Peripheral Interface*) (ALTERA, 2013).

No modo ativo, a máquina de estados do FPGA usa bits para controlar o dispositivo de memória externo, como quando no começo do processo, fornecendo, inclusive o sinal de *clock* durante a leitura. A ideia deste modo é que o FPGA não precise fornecer ao dispositivo de memória externa os endereços de leitura, e sim, simplesmente indicar o momento do início de leitura e obter os dados com os pulsos de *clock*. A Figura 9 expressa como é feita a conexão com uma memória neste modo.

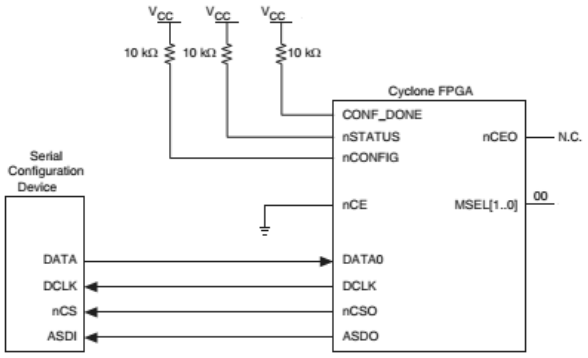


Figura 9 - FPGA Altera em configuração AS com memória flash SPI
Fonte: Altera Serial Configuration Devices Datasheet

A leitura do sinal de dados de configuração para o FPGA só é feita na inicialização do dispositivo. Em alguns casos, quando necessário mais de um FPGA, existe uma memória externa para cada dispositivo. Alternativamente, há a possibilidade de se configurar múltiplos dispositivos (*daisy chain*[11]) com a mesma memória, como na Figura 10 (ALTERA, 2012).

---

[10] *EPC (Enhanced Programmable Configuration) Serial (EPCS)/Quad-Serial (EPCQ) Devices*.
[11] *Daisy-Chain*, é uma forma de conexão onde múltiplos dispositivos são conectados em cadeia.

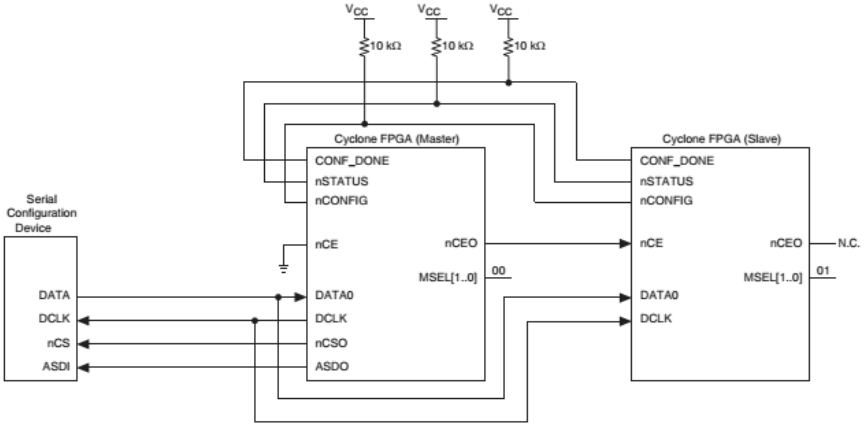


Figura 10 - FPGA Altera em configuração múltipla AS com memória flash SPI
Fonte: Altera Serial Configuration Devices Datasheet

O modo AP é muito semelhante ao modo AS, exceto pela largura do barramento. Além de fornecer bits de controle de início de leitura e sinal de clock, os FPGAs também fornecem ao dispositivo de memória o endereço do byte que será obtido.

O modo paralelo só é aplicável em alguns dispositivos. O FPGA *Altera® Cyclone III* aceita essa configuração, e a fabricante aconselha o uso de memórias *Numonyx StrataFlash P30* e *P33*, dado que suportam modo de leitura síncrono de 40MHz (velocidade do clock para *Cyclone® III* em modo ativo paralelo). Essas memórias são projetadas para fazer interface com microcontroladores, assim, permitem também, combinar dados de configuração com dados de usuário, no mesmo dispositivo, economizando espaço nas PCBs. A Figura 11 representa como é feita a conexão com um FPGA *Cyclone® III*. Neste modo, também é possível a configuração de múltiplos dispositivos (ALTERA, 2008).

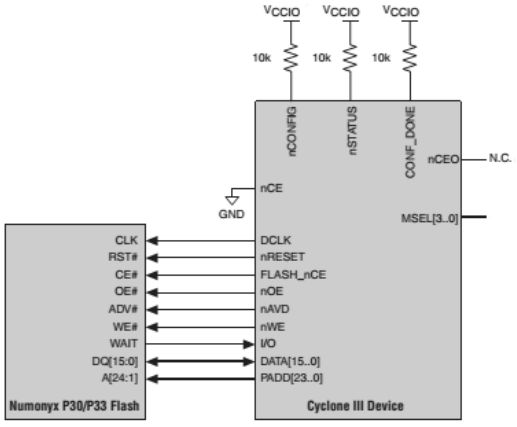


Figura 11 - FPGA Altera em configuração AP com memória flash paralela
Fonte: Altera Configuration Handbook

2.1.2.2 Modo de configuração passivo

No modo passivo, um controlador externo controla todo o processo de configuração. Similar ao modo ativo, dependendo da conexão, os dados podem ser transferidos de forma serial ou paralela.

Na configuração serial, os dados são carregados 1 bit por vez, enquanto que na configuração paralela, há a possibilidade de se carregar 8, 16 ou 32 bits por vez (ALTERA, 2013).

O modo anterior, ativo (2.1.2.1), é atraente pela simplicidade do circuito além de requerer poucos dispositivos na PCB. Entretanto, o modo passivo tem características que agradam qualquer desenvolvedor. Com um simples microcontrolador é possível selecionar dados específicos para a leitura, executar tarefas específicas e de interfaceamento, como no caso de uma conexão TCP/IP, por exemplo.

Aqui, vale ressaltar que se está tratando do processo de configuração do dispositivo FPGA, pois, depois de configurado, o próprio dispositivo pode funcionar como um microcontrolador, por exemplo, como já mencionado. A Figura 12 representa um esquema de ligação para configuração passiva:

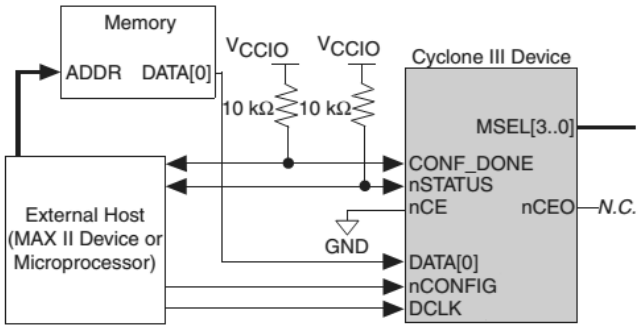


Figura 12 - FPGA Altera em configuração PS
Fonte: Altera Configuration Handbook

O esquema passivo também pode ser: *Fast Passive Parallel* (FPP); *Passive Parallel Synchronous* (PPS) e *Passive Parallel Assynchronous* (PPA). Além destes esquemas, o modo passivo também pode ser JTAG (2.1.2.3), ou CvP (*Configuration via Protocol*) (apenas alguns dispositivos suportam este modo, possuindo *transceivers* específicos).

O modo CvP, usando *PCI-Express*, divide o processo de configuração em duas etapas: na primeira, as entradas e saídas são configuradas, incluindo os *transceivers*, usando dados armazenados em um dispositivo externo, e, na segunda etapa, a configuração é concluída de maneira mais rápida, de forma que uma conexão com a *PCI-Express* pode ser feita para configurar o FPGA.

Ou seja, o modo CvP permite que grandes imagens de configuração sejam transferidas rapidamente para o dispositivo lógico, e ainda, permite reconfigurações rápidas. Este modo, voltado para aplicações de desempenho, tem o objetivo de reduzir custos, proteger o projeto e atualizar o sistema sem pará-lo (ALTERA, 2013).

2.1.2.3 JTAG

JTAG é um padrão IEEE (*Institute of Electrical and Electronics Engineers*) (1149.1 - *Standard Test Access Port and Boundary-Scan Architecture*) que foi desenvolvido para resolver problemas na fabricação de PCBs, mas hoje é usado na programação de dispositivos, depuração e testes. Em seu uso original, JTAG é responsável pelo chamado *Boundary Testing*, ou *Boundary Scan*, usado para testar conexões nos circuitos. O padrão requer o uso de pelo menos quatro pinos especiais dos CIs (TMS, TCK, TDI e TDO) (ALTERA, 2005).

Como muitos outros CIs, os FPGAs são preparados para JTAG. Aqui, não será detalhado o funcionamento do padrão, que está além do propósito deste trabalho. É suficiente entender que o FPGA possui os pinos para JTAG e estes podem ser usados para sua configuração, dado que a ideia por trás do *Boundary Scan* é que, por meio do JTAG, é possível enviar dados de forma serial ao dispositivo.

Com JTAG é também possível enviar comandos especiais que são carregados em registradores específicos por meio da entrada de dados. Um comando específico, por exemplo, instrui o FPGA para sua configuração (MAXFIELD, 2004).

A porta JTAG está sempre disponível, de modo que o dispositivo possa ser configurado tradicionalmente, mesmo quando conectado a uma memória serial, por exemplo, no modo ativo ou passivo.

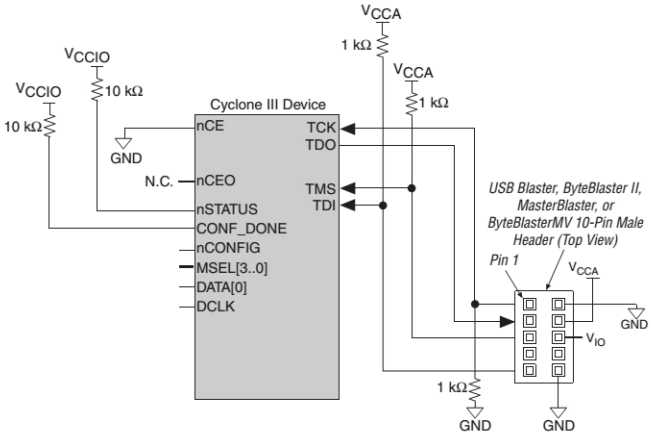


Figura 13 - Exemplo de configuração via JTAG com uso de cabo de download
Fonte: Altera Configuration Handbook

Ressalta-se que o modo JTAG pode ser combinado com o modo ativo serial ou passivo serial, inclusive em *daisy*-chain, como for conveniente ao desenvolvedor. Ademais, sabe-se também que a interface JTAG tem prioridade em relação a outros modos de configuração.

2.1.3 Sequência de configuração

De modo geral, para os modos de configuração descritos, a sequência ou processo de configuração do dispositivo FPGA pode ser dividido em cinco etapas ou fases: *power-up*, *reset*, configuração, inicialização e modo usuário.

A primeira fase, *power-up*, equivale à alimentação correta do dispositivo. Durante a energização, toda fonte de alimentação necessita de um tempo para estabilizar os valores de tensão. Por isso, circuitos integrados como FPGAs são construídos com um circuito PoR (*power-on-reset*) que é disparado quando o valor de tensão atinge valores aceitáveis para o correto funcionamento do circuito (*trip point*, geralmente utiliza-se um circuito comparador como *Schmitt Trigger*), assim, o PoR mantém o restante do circuito em *reset* até que os níveis de tensão se estabilizem.

No estado de *power-up*, os pinos de acesso ao usuário são colocados em modo *tri-state*, ou seja, em alta impedância e, portanto, considerados desconectados do restante do circuito, até a finalização da configuração. Para projetos complexos,

para determinados dispositivos, o atraso de *power-up* pode ser programado através de um pino dedicado.

Na documentação dos dispositivos lógicos, além das corretas tensões de alimentação (*VCCIO, VCCINT, VCCA, VCCD_PLL*), geralmente se indicam também, os tempos de *power-up* e atrasos na inicialização (*PoR delay* típico de no máximo 100 ms para *Altera Cyclone* ®).

Uma vez corretamente alimentado, o dispositivo entra na etapa de *reset*, automaticamente. Neste segundo estado, os pinos usados *nSTATUS* e *CONF_DONE* são colocados em baixa impedância e todos os outros pinos de I/O e a memória RAM são limpos. Na sequência os pinos de seleção da configuração (*MSEL*) são amostrados. Aqui, os pinos I/O usuários ainda estão em *tri-state.*

O FPGA permanecerá em reset enquanto o pino *nCONFIG* for mantido em nível baixo. Uma transição neste pino habilita a etapa de configuração, e quando *nCONFIG* for levado a nível alto, o pino *nSTATUS*, dreno aberto, passa a nível alto, também, com uso de uma alta resistência de *pull-up* externa. Assim o processo de configuração pode começar. Ou seja, o pino *nCONFIG* permite o controle do processo de configuração. Quando este pino for levado a nível baixo, *nSTATUS* e *CONF_DONE* também serão nível baixo. A configuração recomeça se o pino *nCONFIG* for levado a nível alto novamente. (ALTERA, 2013).

Na terceira etapa, de configuração, o dispositivo recebe os dados de configuração em seus pinos de dados (*DATA*) e, para esquemas síncronos, sinal de clock no pino *DCLK*. Os dados de configuração são carregados na borda de subida de *DCLK.*

Após receber o fluxo de dados de configuração completamente, o FPGA libera o pino *CONF_DONE* (dreno aberto), colocado em nível alto por meio de uma resistência de *pull-up* externa. Assim, quando há borda de subida em *CONF_DONE*, a etapa de inicialização do dispositivo pode começar.

Na etapa de inicialização, todos os registradores e buffers internos são ativados. O pino *INIT_DONE* pode ser usado para sinalizar o final da inicialização do dispositivo e o início do modo usuário. Se o pino não for ativado mesmo após o final da inicialização, então *DCLK* deve continuar a chavear até a conclusão da inicialização. Cada dispositivo necessita de um número adicional de ciclos de clock, especificado em sua documentação, para completar a inicialização (ALTERA, 2013).

Uma vez inicializado, o FPGA está no modo usuário, assim os pinos de I/O não estão mais em *tri-state*. No modo usuário, os pinos responsáveis pela configuração não devem flutuar, para evitar reinicializações ou configuração não pertinentes ao seu correto funcionamento. Por isso, estes pinos devem ficar em nível alto ou baixo, o que for conveniente (ALTERA, 2013). Em qualquer momento do processo de configuração, uma reconfiguração pode ser iniciada, alternando o pino *nCONFIG* em um ciclo e reiniciando o processo.

O modo JTAG tem prioridade no processo, ou seja, a configuração pode ser reiniciada por meio dos pinos específicos para JTAG.

A Figura 14 expressa essas etapas descritas de configuração em um diagrama de tempo, com relação à tensão de alimentação.

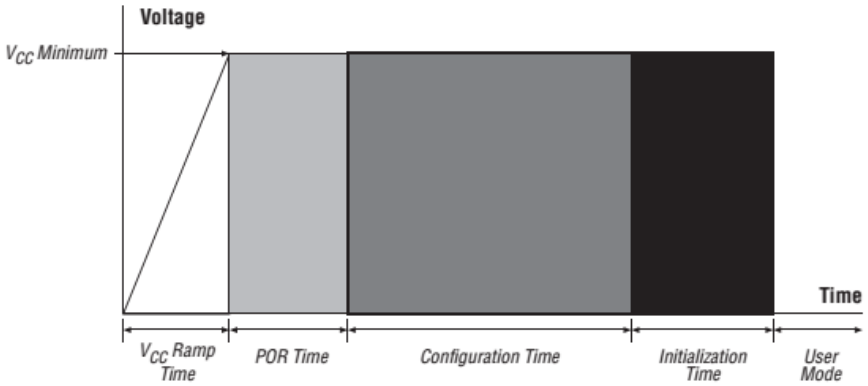


Figura 14 - Estágio de Power Up (etapas de tempo com relação a alimentação)
Fonte: Altera Cyclone III Design Guidelines

2.1.4 Soluções de configuração

Os fabricantes de FPGAs também disponibilizam ferramentas que facilitam a configuração dos dispositivos, como memórias não voláteis e cabos de download. A fabricante *Atmel*®, por exemplo, possui uma linha de produtos específicos para configuração de dispositivos lógicos programáveis, como os circuitos da família *AT18F*®, com capacidade de armazenamento de 1 Mb a 8 Mb, para configuração de FPGAs baseados em SRAM, ou a família de memórias EEPROM *AT17LV*®, de até 32 Mb, programáveis em 5 e 3,3 Volts, para dispositivos de terceiros, como FPGAs *Lattice*®, *Xilinx*® e *Altera*®.

A fabricante *Altera*® tem em sua linha de produtos memórias serial do tipo FLASH, construídas para operar em projetos com FPGAs Altera®, que são os CIs EPCS e EPCQ, já citados no tópico 2.1.2.1 (Esquema de configuração ativo). Estas memórias suportam o esquema de configuração ativo, apenas, e são programadas *in-system* (ISP[12]) (ALTERA, 2012). A fabricante deixa claro que outras memórias seriais não voláteis também podem ser utilizadas como solução de configuração.

Como estas memórias não possuem interface JTAG, a programação pode ser feita via dispositivo secundário, como um microcontrolador ou um cabo de download. Essa seria uma solução trivial, simples e rápida, mas requer conectores distintos na PCB (um para configurar o FPGA e outro para programar o EPCS ou EPCQ).

Mas, é possível configurar o FPGA e programar a memória serial pela mesma interface JTAG, pelo próprio conector de configuração JTAG da FPGA, com uso de um *megafunction*[13] denominado SFL – *Serial Flash Loader.* Neste caso, uma imagem prévia (como um *boot loader*[14]) é carregada na FPGA a fim de criar uma ponte entre a memória e a interface JTAG, como pode ser visto na Figura 15 (ALTERA, 2012).

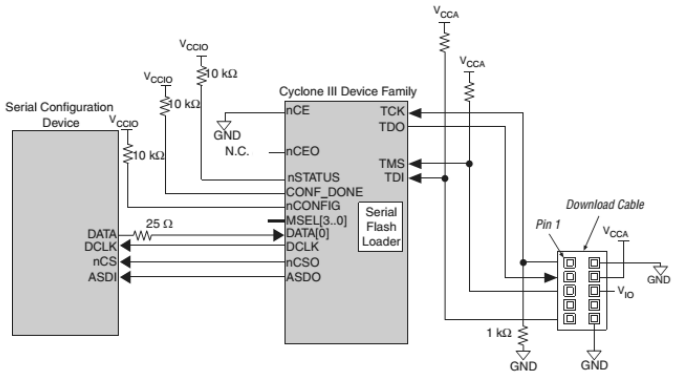


Figura 15 - Esquema para programação de dispositivo serial via cabo ou interface JTAG
Fonte: Cyclone III Device Handbook

[12] ISP, do inglês, *in-system-programmable,* que se refere à capacidade de circuitos integrados serem programados mesmo após sua instalação.
[13] *Megafunctions,* são subsistemas parametrizados (no caso, pela *Altera*®) criados para otimizar e tornar eficiente o trabalho de desenvolvimento de sistemas em linguagem de hardware (HDL).
[14] *Boot Loader* ou *Bootstrap,* é um pequeno programa utilizado para carregar um segundo sistema ante ao carregamento do sistema principal.

Outra solução de configuração apresentada para dispositivos Altera®, é um dispositivo de memória FLASH paralelo (PFL – *Parallel Flash Loader*), como o comum CPLD *MAX II*, usado na programação de memórias FLASH, como uma ponte entre sua interface JTAG e a interface de dados, e também pode ser usado para controlar a configuração do FPGA usando os dados armazenados na memória FLASH. Portanto, é uma solução que permite programar o dispositivo FLASH indiretamente. Ao optar por essa solução, o desenvolvedor deve fazer uso do *megafunction* PFL para programar o dispositivo *MAX II* (ALTERA, 2012).

Alguns conhecidos kits de desenvolvimento usam dessa solução, como o kit *Altera® DE0 Board,* da *Terasic®*, permitindo a configuração direta do FPGA (modo '*run*' - JTAG) e a programação da memória EPCS (modo '*prog*' - AS). E o mesmo pode ser feito com o uso de um microcontrolador, pré-programado para a mesma função. A Figura 12 mostra as conexões de interface de configuração entre um dispositivo *Cyclone® III* e um dispositivo ou host externo responsável pela configuração de uma memória.

De modo geral, estas são as soluções de configuração em uma visão de hardware. Outras soluções fundamentam-se nestas para configurar os dispositivos lógicos programáveis. Os cabos de download (*BitBlaster™, ByteBlaster™ II, ByteBlasterMV™, ByteBlaster™ Parallel, Master Blaster™, USB-Blaster™, Ethernet Blaster, OpenJTAG*, entre outros) baseiam-se na teoria já descrita aqui, somando-se a facilidade de interfaceamento *Software <-> JTAG*.

Também devem ser levadas em conta as soluções de configuração baseadas em um *software driver*, onde o desenvolvedor pode usar um processador externo para configurar FPGAs e programar CPLDs ou dispositivos de configuração. Podem-se ressaltar os seguintes: *Jam™Player, JRunner™, MicroBlaster™* e *SRunner™*.

O padrão *Jam™* e a linguagem de programação STAPL (*Jam Standard Test and Programming Language*), posterior padrão JEDEC[15] JESD-71, é um nível de software para *in-system programming and configuration*, onde desenvolvedores

[15] JEDEC: *Joint Electron Device Engineering Council*. É o órgão para padronização de engenharia de semicondutores da EIA – Electronic Industries Alliance, que representa todas as áreas da indústria de eletrônicos dos Estados Unidos.

costumam implementar para flexibilizar o desenvolvimento, qualidade e ciclo de vida dos produtos finais (ALTERA, 2013).

2.1.5 Arquivos de configuração

Como já citado, quando na descrição e compilação de um projeto, o resultado final é um arquivo de configuração que contém todas as informações que serão enviadas para o PLD. No software *Altera Quartus® II*, alguns arquivos de configuração padrão são gerados: SOF ou POF.

O formato SOF (*SRAM Object File*) é o arquivo de dados de configuração utilizado durante uma configuração PS, quando o fluxo de dados é enviado para o FPGA por um cabo de download. O formato POF (*Programmer Object File)* é o arquivo usado para programar CPLDs e outros dispositivos de configuração.

Diversos outros formatos de dados de configuração podem ser gerados na compilação no *Quartus II*. O formato JAM (*JAM File*) é um arquivo ASCii que é usado para configurar ou programar um ou mais FPGAs, CPLDs e dispositivos de configuração em um *JTAG chain*. Similar ao formato JAM, o formato JBC (*Jam Byte-Code File*) é um arquivo binário que também é usado para configurar ou programar todos os dispositivos em um *JTAG chain* (ALTERA, 2008).

O formato JIC (*JTAG Indirect Configuration File*) é utilizado para programar dispositivos em SFL (*megafunction serial flash loader*) através da interface JTAG quando em modo AS (ALTERA, 2008).

O formato CDF (*Chain Description File)* é um arquivo simples que contém dados do projeto no compilador, como modo de configuração, arquivos usados, informação do dispositivo e outras informações específicas (ALTERA, 2008).

Outros arquivos também podem ser gerados no software da *Altera* ®, mas não são suportados pelo programador. São eles: RBF, RPD, TTF e HEXOUT.

O formato RBF (*RAW Binary File*) é um arquivo binário que contém dados de configuração para um ou mais dispositivos FPGAs. Este arquivo pode ser armazenado em uma unidade de armazenamento, por exemplo, e um microcontrolador pode ler os dados do arquivo e carregá-lo no FPGA. O formato

RPD (*RAW Programming Data File*) é um arquivo binário que contém os dados de configuração para um dispositivo de configuração EPCS (ALTERA, 2008).

O formato TTF (*Tabular Text File*) é um arquivo ASCii que contém uma versão dos dados de configuração separados por vírgulas para os modos FPP, PPS, PPA e PS, permitindo que o desenvolvedor inclua os dados de configuração como parte de um código fonte, onde um microcontrolador ou processador possa acessá-lo e carregá-lo em um dispositivo lógico (ALTERA, 2008).

O formato HEX ou HEXOUT é um arquivo ASCii do padrão *Intel HEX*. Microcontroladores ou hosts externos podem usar este tipo de arquivo para armazenar e transmitir dados de configuração. Este formato também pode ser usado em programadores de terceiros para programação de dispositivos da Altera (ALTERA, 2008).

Estes formatos citados são os arquivos mais utilizados em configuração de dispositivos lógicos programáveis e um conhecimento prévio é importante para projetos com FPGAs. Outros formatos de arquivos não listados aqui podem ser utilizados em configurações parciais e/ou reconfigurações online.

2.2 APLICAÇÃO

Com a teoria apresentada até aqui, é possível o projeto inicial de uma PCB como exemplo de protótipo para configuração de um FPGA.

Apresentado os modos e esquemas de configuração e optando-se por manter duas formas de configuração do dispositivo lógico programável, escolhe-se combinar o modo passivo JTAG com o modo ativo serial (AS), utilizando-se de um dispositivo de memória externa (EPCS).

Para a aplicação JTAG, pode-se desenvolver uma interface com o computador, de modo a gerenciar a comunicação com o software de desenvolvimento de conhecimento (*Altera® Quartus II*). Como visto, existem diversas alternativas para essa interface, como uma vasta gama de cabos de download.

Mas, aqui, se escolhe desenvolver a própria interface *USB <-> JTAG* com uso de um microcontrolador (com USB) e bibliotecas já em desenvolvimento e/ou disponíveis na internet, como o projeto *UrJTAG* (*urjtag.org).* O projeto do japonês

Satoshi Ojika foi escolhido como referência, já que emula o driver da ferramenta *USB-Blaster* (facilitando a comunicação com o software da A*ltera*®), por meio de um microcontrolador PIC Microchip® *18F14k50*.

Em termos de escolha do CI FPGA, optou-se por um dispositivo com encapsulamento não-BGA (*Ball Grid Array*) de forma a facilitar a construção do protótipo, escolhendo um encapsulamento QFP (*Quad Flat Package*), com menor número de pinos, dado o processo não industrial de confecção da PCB.

A capacidade física do dispositivo FPGA não foi levada em consideração para este protótipo, como número de elementos lógicos (LEs), número de registradores, elementos multiplicadores, capacidade de memória RAM, número de PLLs (*Phase-Locked Loop*) e grades de velocidade (*speed grade*), embora essas informações sejam de grande importância para a descrição do projeto, servindo como base para a definição do hardware.

Ademais, a série *Altera® Cyclone III* suporta a migração dentro do mesmo encapsulamento, tornando os projetos flexíveis. Por exemplo, para um dado dispositivo, caso o projeto necessite de mais elementos lógicos (ou menos, caso barateamento do projeto), os dispositivos de diferentes densidades têm os mesmos locais de pinos de alimentação, configuração e pinos dedicados, o que permite atualização ou mudanças nos projetos sem necessidade de mudança de layout (ALTERA, 2013).

Como se optou pelos modos de configuração passiva JTAG e ativo serial, precisa-se selecionar uma família de FPGAs que suporte esses dois modos. Os FPGAs *Altera* da família *Cyclone ® III* suportam essas configurações (ALTERA, 2008). O dispositivo FPGA escolhido: *ALTERA CYCLONE III EP3C5E144C8N*.

Para a escolha do dispositivo de memória (EPCS) apropriado, é preciso determinar o espaço de configuração total necessário para o FPGA em uso. Para o caso, o FPGA escolhido necessita de um tamanho máximo de 2.944.088 bits de dados de armazenamento. Logo, segundo (Altera, 2008), um dispositivo EPCS4 ou outros de capacidade superior, como EPCS16, EPCS64 ou EPCS128, podem ser usados. No caso, escolhe-se: *ALTERA EPCS4SI8N*, que permite armazenar até 4.194.304 bits de dados, ou seja, adequa-se com folga ao FPGA escolhido.

Na definição das funcionalidades do protótipo em PCB, opta-se por simples dispositivos de interface, como *push-buttons* e *LEDs*, além de conectores *general*

purpose input/output (GPIO) que possibilitam o acesso a quaisquer outros pinos do FPGA, facilitando a expansão de hardware ao desenvolvedor, para incluir outros atributos, conforme necessidade e equiparando-se a outros kits de desenvolvimento existentes.

Além dos propostos, frisa-se a existência de outros pontos que são levados em consideração, como a necessidade da disponibilidade de um conector JTAG, mesmo com a presença da interface de configuração USB com a finalidade de testes posteriores e/ou acesso aos pinos JTAG no padrão de conexão dos cabos de download (*header* 10 pinos); a atenção ao fornecimento de tensão, dado as diferentes tensões e pinos de alimentação; no projeto de uso de capacitores de desacoplamento e outras considerações.

Com o levantado até aqui, monta-se um esquema do PCB ante ao *design*. O *placement*[16] do projeto pode ser visualizado na Figura 16. Todos os esquemáticos completos podem ser encontrados no APÊNDICE A - Circuito Eletrônico (Esquemáticos).

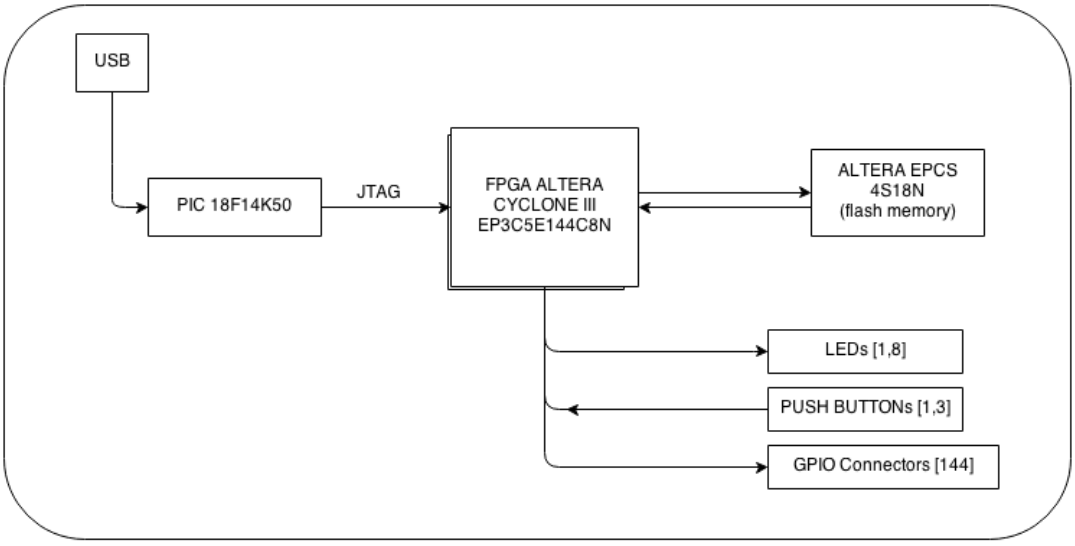


Figura 16 - Placement do projeto em desenvolvimento
Fonte: Autoria própria

[16] *Placement* é uma visualização geral do projeto, que atribui uma localização prévia para os componentes do design. É muito utilizado em projetos EDA (*Electronic Design Automation*).

2.2.1 Recursos

Dado o desenvolvimento do protótipo e testes, até o funcionamento, os recursos utilizados durante todo ou parte do projeto, são:

- Computador pessoal;
- *PCB Design Software* (*Altium™ Designer Trial*)*;*
- *CAD PLD Sofware (Altera™ Quartus® II Web Edition);*
- *Circuit Board Plotter (LPKF™ Rapid PCB Prototyping* - UTFPR, Campus Toledo)*;*
- *In-Circuit Debugger/Programmer* (*PICKit 3 Microchip®* - próprio)*;*
- Equipamentos para solda de eletrônicos (estação de solda, fluxo de solda, etc);
- Equipamentos eletrônicos de bancada (multímetro, osciloscópio, etc);
- Dispositivos e componentes eletrônicos (*EP3C5E144C8N*, *EPCS4SI8N, PIC18F14k50*, *push-buttons*, *LEDs*, e outros componentes à definir, como conectores, capacitores, resistores, cabo USB).

2.2.2 USB/JTAG

Para a aplicação JTAG, como dito, basear-se-á no projeto de OJIKA. Um microcontrolador, com interface USB, programado via ICSP (*In-Circuit Serial Programming*), funcionará como controlador do fluxo de dados USB -> JTAG. A ideia de emular o driver do *USB-Blaster* é facilitar a configuração do FPGA via software de desenvolvimento *Altera Quartus II*.

O circuito é simples e é exibido nas figuras 17 e 18. A função dos divisores resistivos é a de adequar a tensão de 5 V do microcontrolador com a tensão de 3,3 V do FPGA (2.2.4 Alimentação).

Observa-se que a alimentação do circuito será via interface USB (5V – 500mA). Verifica-se a existência dos conectores USB, ICSP (6 pinos) e JTAG (10 pinos).

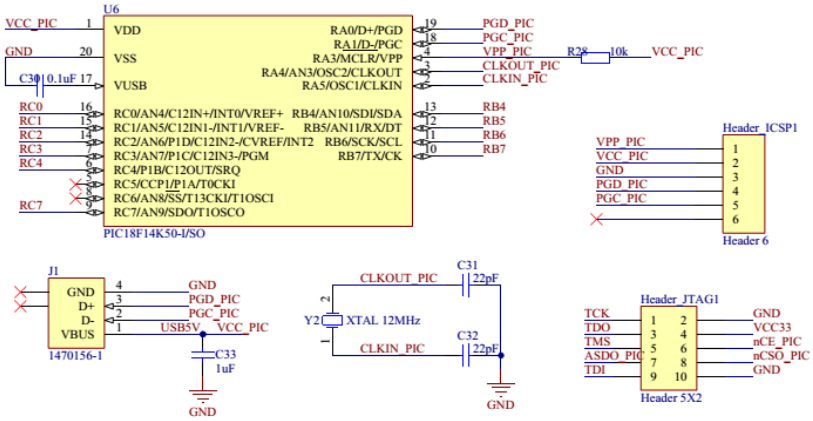


Figura 17 - Circuito PIC 18F14k50 (USB-Blaster)

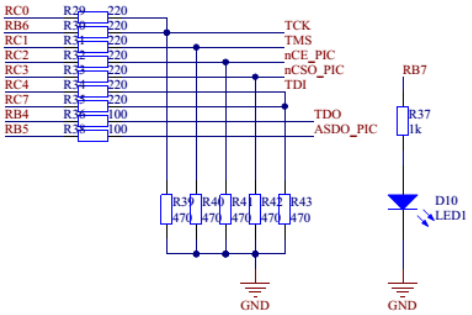


Figura 18 - Circuito PIC 18F14k50 (USB-Blaster) - pinos

Dos 7 pinos de dados disponíveis no microcontrolador, apenas 4 pinos são usados para a configuração do FPGA em modo JTAG (TDI, TODO, TCK e TMS). Os pinos restantes são utilizados em outros esquemas de configuração, como no modo AS e PS, no caso de programação direta, que não é o caso desse protótipo (o uso do modo ativo será via *megafunction Serial Flash Loader*, com programação indireta da memória EPCS).

Por ora, é necessário saber que se utiliza o driver USB-Blaster para configurar o FPGA em modo JTAG, e os pinos estão disponíveis.

2.2.3 FPGA

Tratando-se do FPGA *ALTERA CYCLONE III EP3C5E144C8N,* todas as informações de configuração como pinos e esquemas básicos de conexão são obtidas por meio do *Altera Cyclone III Device Handbook* (ALTERA, 2012).

Como se optou pelo modo JTAG e AS, como forma de utilizar apenas um header/conector ou interface de programação, o esquema da ligação pode ser visto na Figura 15, onde a memória serial é programada pela mesma interface JTAG que o FPGA é configurado, com uso de um *boot loader.*

Assim, com este modo de configuração, o fabricante dá algumas exigências/requisitos para o correto funcionamento do dispositivo. Um deles é conexão dos pinos MSEL (pinos de seleção do modo de configuração), que podem selecionar o modo de configuração em uso; conectados diretamente na tensão VCCA ou GND, conforme tabela pré-determinada pelo fabricante (não se devem deixar os pinos MSEL desconectados).

Outra observação é o uso de resistores de *pull-up (*em relação à VCCIO*)* nos pinos de configuração não usados neste esquema (*nSTATUS, nCONFIG, CONF_DONE*), e referenciar outro(s) pino(s) (*nCE*).

O fabricante também aconselha o uso de resistores de *pull-up* (em relação à VCCA) e *pull-down* entre os pinos de dados do conector JTAG (com valores de 1kΩ a 10kΩ) para garantir a integridade do sinal de dados.

O circuito esquematizado, com os pinos do FPGA divididos em blocos, é exibido na sequência: Figura 19, Figura 20, Figura 21, Figura 22 e Figura 23.

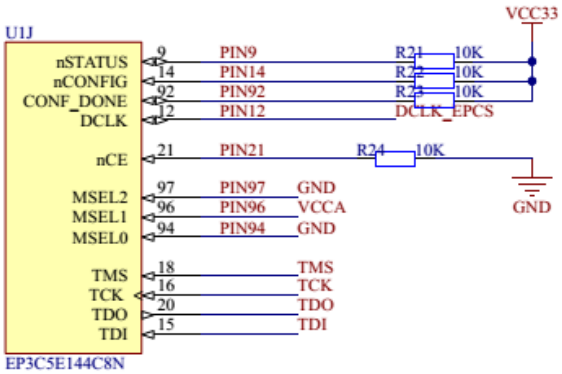


Figura 19 - Circuito FPGA EP3C5E144C8N (bloco de configuração)

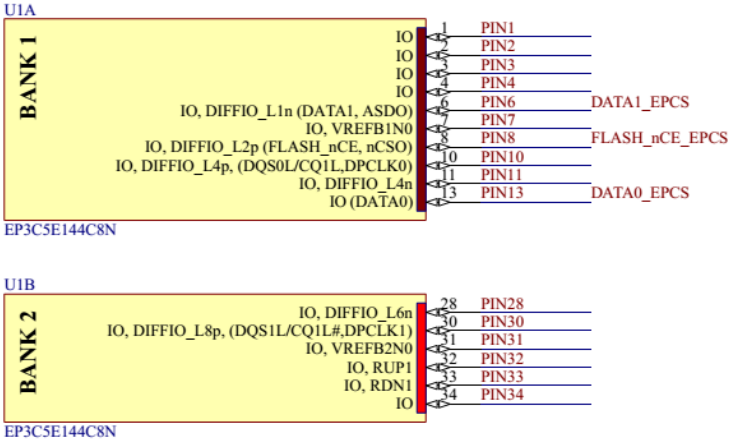


Figura 20 - Circuito FPGA EP3C5E144C8N (bancos 1 e 2)

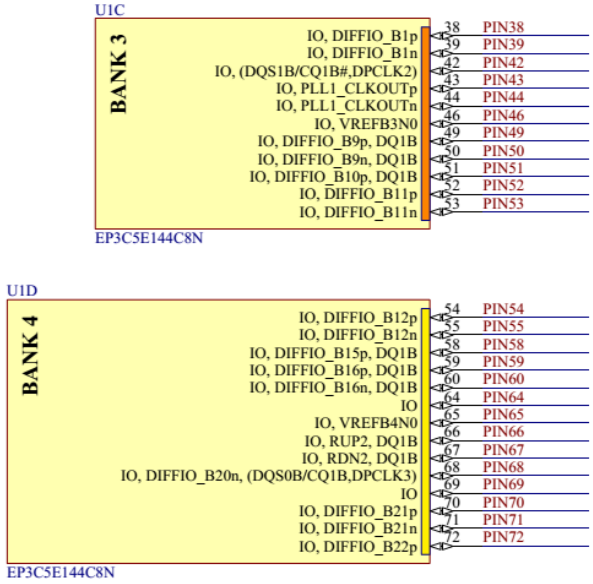


Figura 21 - Circuito FPGA EP3C5E144C8N (bancos 3 e 4)

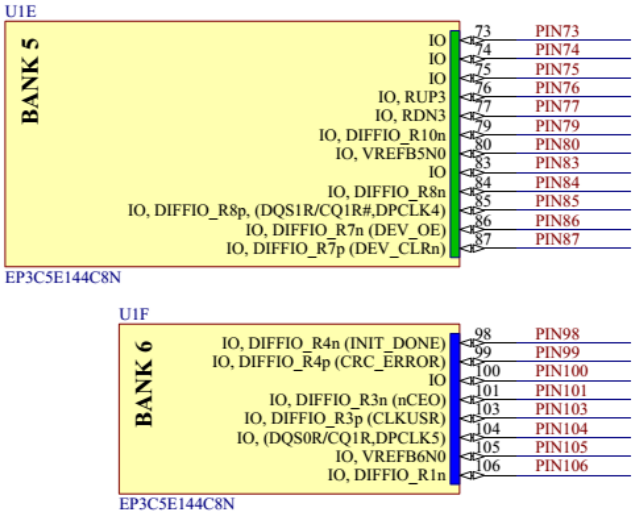


Figura 22 - Circuito FPGA EP3C5E144C8N (bancos 5 e 6)

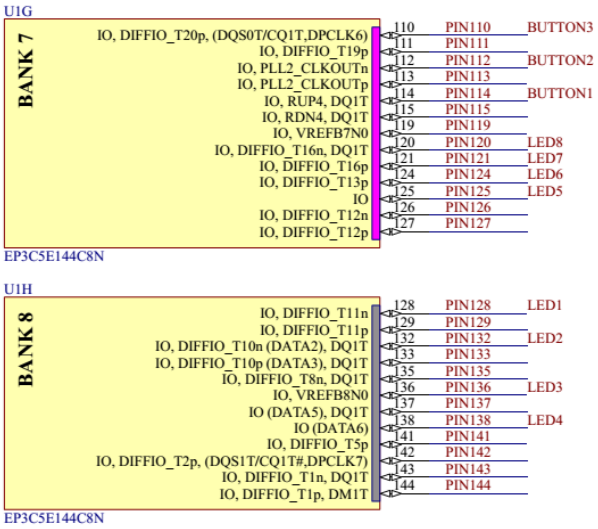


Figura 23 - Circuito FPGA EP3C5E144C8N (bancos 7 e 8)

2.2.4 Alimentação

Neste protótipo, o circuito de alimentação é simples e baseia-se no regulador de tensão LM317. A tensão de entrada é obtida por meio do conector USB em 5 V (500 mA) e, a partir desse valor de tensão, são obtidos os outros valores de tensão para os circuitos e FPGA: 3,3 V, 2,5 V e 1,2 V - Figura 24.

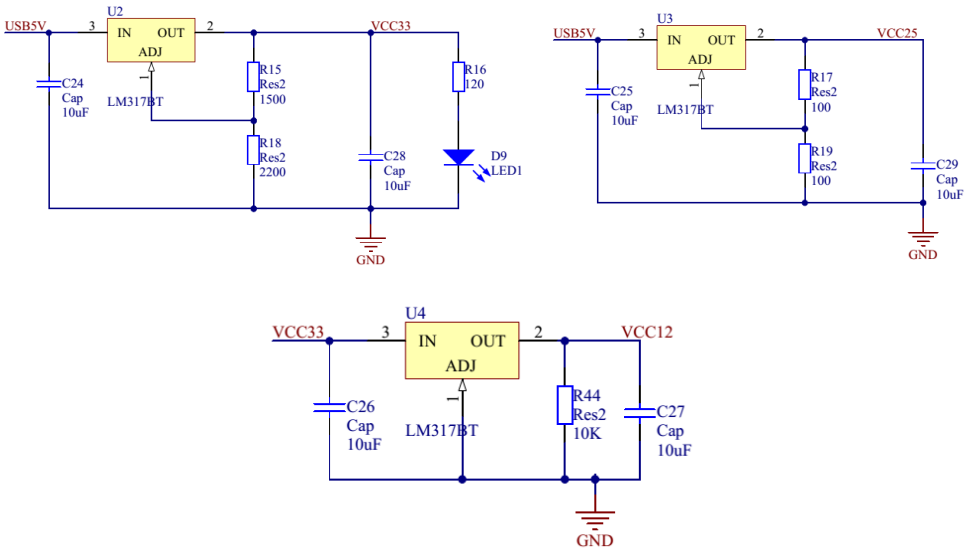


Figura 24 - Circuitos reguladores 3,3 V; 2,5 V e 1,2 V

Para projetos que exigem eficiência, com relação ao fornecimento e consumo de energia, todo projeto exige um planejamento prévio, onde a fonte de

potência deve ser capaz de fornecer corrente elétrica suficiente para o funcionamento do dispositivo.

O que parece ser uma obviedade é motivo de discussões constantes em PCBs com FPGA. Um projeto pode exigir uma potência diferente na etapa de design em comparação à aplicação. O chamado *Early System* Design, é um estudo importante, principalmente em projetos complexos, e, com ele, é possível estimar potências necessárias, com base em fatores de entrada: temperatura ambiente, fluxo de ar, frequência de clock, por exemplo.

Da alimentação, dos pinos de I/O, há a possibilidade de flexibilização, onde cada banco (*banks*), pode ser alimentado com um valor de tensão individual (BANK1 = VCCIO1, ex. Figura 25), pois as tensões de cada banco, internamente ao dispositivo, são independentes das tensões VCCIO dos outros bancos. O fabricante alerta para que todos os bancos sejam alimentados. Aqui, aconselha-se sempre conferir os níveis de tensão suportados, consultando-os na documentação do dispositivo e/ou família.

Além disso, cada banco do FPGA possui um pino de referência de tensão (BANK1 = VREFB1, ex. Figura 20) para adaptação de tensão dentro daquele banco, dado que, cada banco só pode ter um único nível VCCIO e um nível VREF. Essa tensão VREF pode ser usada como uma referência para as entradas de tensão para determinar os limiares da lógica em uso, assim, é importante que estes pinos sejam livres de ruídos. Pinos VREF podem ser usados como pinos de I/O, mas o fabricante aconselha não utilizar para este propósito, dado que é um pino mais lento em relação aos demais por possuir maior capacitância de entrada (por isso, não se deve usar como pino de clock).

Segundo (ALTERA, 2012), as tensões requeridas para funcionamento:

- VCCINT: fornecimento de tensão para a lógica interna do dispositivo (tipicamente 1,2 V);
- VCCIO: fornecimento de tensão para buffers de saída (de 1,2 V a típico 3,3 V, como o desenvolvedor escolher);
- VCCA: tensão analógica para o regulador PLL (tipicamente 2,5 V);
- VCCD_PLL: tensão digital para o regulador PLL (tipicamente 1,2 V);

Assim, para um projeto qualquer, ao menos dois valores de tensão são requeridos. VCCIO é tipicamente 3,3 V, devido à facilidade de acoplamento com outros dispositivos. Desta forma, são três valores de tensão que devem ser fornecidos ao CI. As figuras 25 e 26 exibem os pinos de alimentação conectados no esquemático.

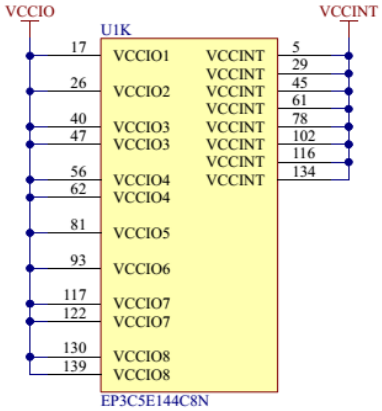


Figura 25 - Circuito FPGA EP3C5E144C8N (alimentação: VCCIO e VCCINT)

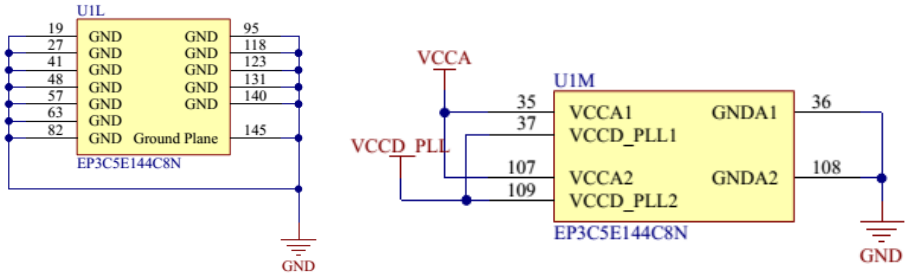


Figura 26 - Circuito FPGA EP3C5E144C8N (referência/ground e alimentação PLL)

O pino 145 (Figura 26) é um plano *ground* localizado na parte inferior do CI (encapsulamento EQFP) que deve ser conectado à referência, caso contrário o FPGA não é inicializado permanecendo em estado de *reset*.

Com esse considerável número de pinos de tensão, espalhados pelas laterais do CI, muitas vezes, a minimização no comprimento das trilhas pode se tornar uma tarefa complicada para o projetista, principalmente em PCBs de dupla face, apenas.

Desta forma, sem entrar no mérito físico da construção de PCBs, aconselha-se abusar, apropriadamente, de capacitores de desacoplamento. Estes são utilizados para manter a tensão constante em transientes, além da capacidade de filtro, diminuindo ruídos de alta frequência (KILTS, 2007).

2.2.5 EPCS

Como já mencionado, a memória serial EPCS4 é conectada diretamente ao FPGA e será programada de maneira indireta, por meio do JTAG. O esquemático pode ser entendido na Figura 27, com pinos conectados no FPGA (vide Figura 19 e Figura 20).

Ao conectar um dispositivo de configuração em série com um dispositivo *Cyclone® IIII,* há necessidade da conexão de um resistor série (~25 Ω) próximo ao dispositivo de configuração série (pino DATA). O resistor tem a função de minimizar a diferença de impedância do dispositivo com a trilha da placa e reduzir *overshoots*[17] no pino de dados (ALTERA, 2012).

Para programação da memória (via JTAG/SFL), um arquivo no formato JIC (*JTAG Indirect Configuration File*) deve ser gerado a partir do arquivo SOF do projeto (*Using the Serial FlashLoader with the Quartus II Software*, ALTERA, 2012). Após programação, quando a PCB for conectada a alimentação, o FPGA em modo ativo serial (vide pinos de seleção *MSEL*), irá carregar os dados de configuração da memória EPCS automaticamente.

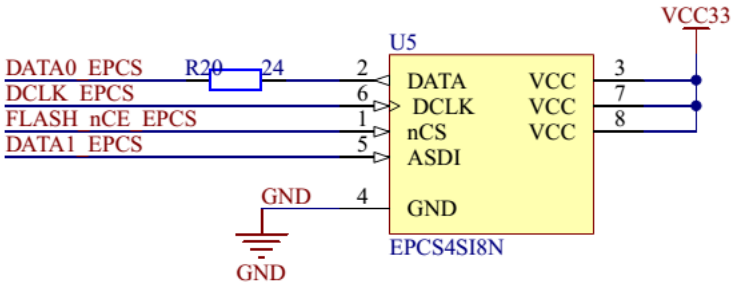


Figura 27 - Circuito *ALTERA EPCS4SI8N* (memória)

2.2.6 Interface

Tratando-se de interface com o usuário, fora colocados 8 LEDs, 3 *push-buttons* e 4 barramentos/*headers* como GPIO, conectados diretamente ao FPGA. Todos os pinos do FPGA podem ser acessados via barramento.

---

[17] *Overshoot:* quando um sinal ou função excede um valor alvo.

3 RESULTADOS

Assim como o ditado nos objetivos (1.2 Objetivos), já se respondeu àquelas perguntas básicas (Por que configurar? Como configurar? Quais os passos?) relacionadas à configuração de FPGAs de forma que, mesmo com a escolha de um único dispositivo da marca *Altera®,* não há nenhum impedimento para replicação da teoria aqui explícita para outros dispositivos e de outros fabricantes, sabendo-se das divergências, quanto à nomenclatura, pinos específicos, ferramentas de desenvolvimento e soluções de configuração, que possam ser encontradas pelo desenvolvedor, de modo que todo o conceito processual é equivalente.

Dado o objetivo de uma aplicação de um simples protótipo, com uso do software *Altium™ Designer,* somado ao teórico já levantado até então, é possível, através dos esquemáticos, o projeto de uma PCB. Os esquemas completos estão no APÊNDICE A - Circuito Eletrônico (Esquemáticos).

A partir dos esquemáticos e de uma prévia análise do encapsulamento dos dispositivos a serem usados, em posse dos *datasheets*[18] fornecidos pelos fabricantes, e das dimensões para *footprints*[19], o layout da PCB começa a tomar forma. Limitado a uma confecção de apenas dois *layers*/camadas (camada *top*/superior e *bottom/*inferior), e como qualquer projeto de PCB, o trabalho da disposição dos componentes, de vias e *pads* (furos e superfícies de contato com dimensões apropriadas), do traçado de trilhas ou roteamento (minimizando a dimensão do comprimento de trilhas), da atenção a dimensões (proximidade ou afastamento de componentes) e de quaisquer outras características físicas e técnicas necessárias (assim como o uso de capacitores de desacoplamento), é uma tarefa que requer atenção e experiência do designer.

Após uma análise para o melhor layout objetivando o fácil acesso a todos os pinos do CI FPGA (144 pinos), facilitando a disposição dos outros componentes e a posterior soldagem, o melhor arranjo encontrado pode ser visto nas figuras 28 e 29,

---

[18] *Datasheet* refere-se a todo documento que apresenta os dados e características técnicas resumidas de um dispositivo/equipamento/produto.
[19] *Footprint* refere-se às dimensões físicas para o componente na placa de circuito impresso, e pode consistir de furos, *pads* (superfície de contato) e contornos, por exemplo.

com utilização de *headers* para acesso a todos os pinos do FPGA. Toda a arte final está no APÊNDICE B - Placa de Circuito Impresso (Arte Final).

Uma visão da face superior após renderização em modelo tridimensional e tamanho real pode ser vista na Figura 30.

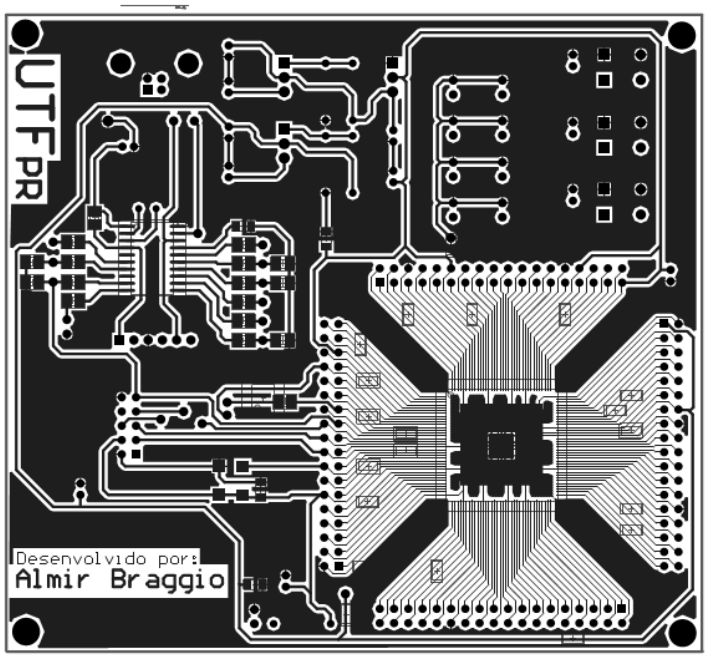


Figura 28 - Vista Superior PCB (camada top) (12.4 cm x 11.35 cm)

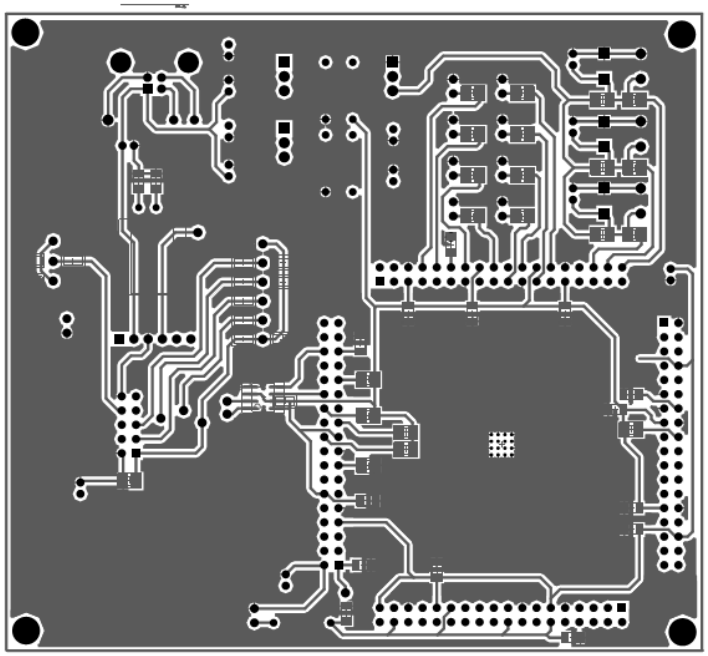


Figura 29 - Vista Inferior PCB (camada bottom) (12.4 cm x 11.35 cm)

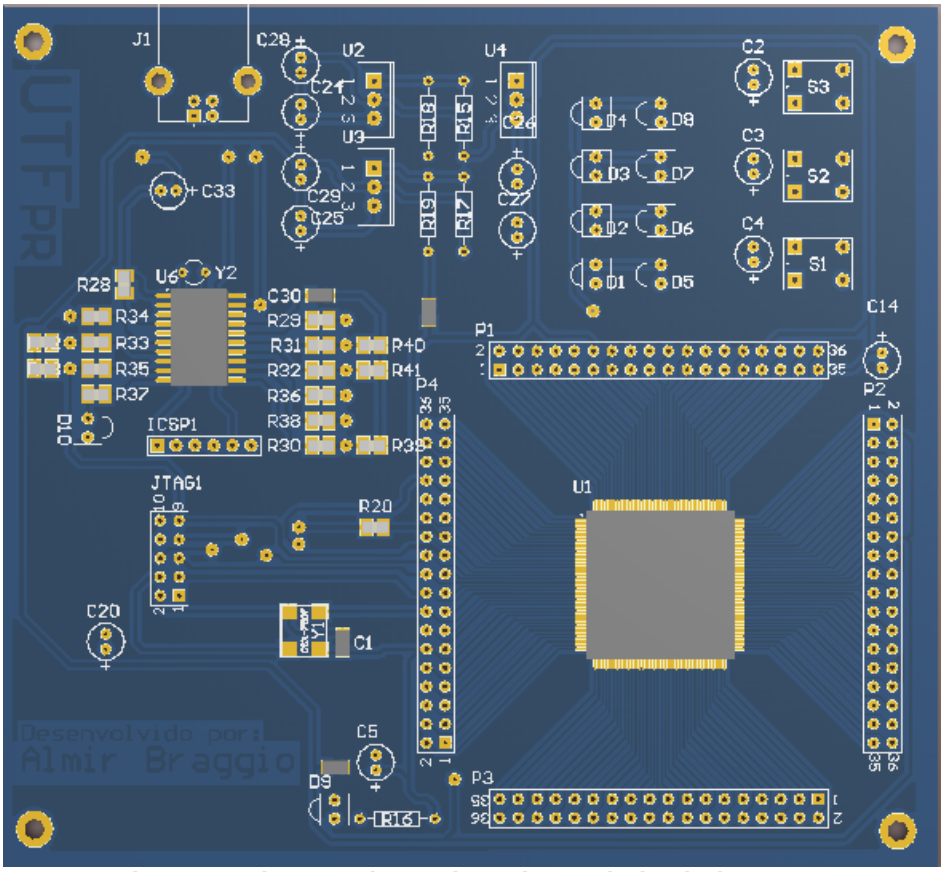


Figura 30 - Vista Superior PCB (camada top) - renderizada (12.4 cm x 11.35 cm)

A geração dos arquivos *Gerber*[20] (*Gerber Files e Drill Files*) foi feita para a impressão da PCB em *Circuit Board Plotter (LPKF™ Rapid PCB Prototyping).* A solda dos componentes é feita com uso de estação de solda, fluxo de solda e solda de estanho-chumbo (0,5mm). Aqui, não houve metalização dos furos, o que dificultou a etapa de solda.

A programação do microcontrolador *PIC 18F14k50* foi feita com uso de um programador ICSP (*PICkit3*®), com firmware compilado em software *Microchip*® *MPLAB IDE* com compilador *C18 Compiler.*

Como resumo, pré-testes, a aplicação final objetiva o funcionamento do protótipo com o software *Altera*® *Quartus II*, de modo a rodar uma aplicação simples descrita em linguagem VHDL, configurando o FPGA em modo JTAG e modo ativo serial, com uso da memória flash EPCS.

---

[20] Formato *Gerber* são arquivos de imagem 2D, utilizado na indústria eletrônica para descrição de placas de circuito impresso.

As imagens seguintes (Figura 31 e Figura 32) são fotografias do protótipo finalizado.

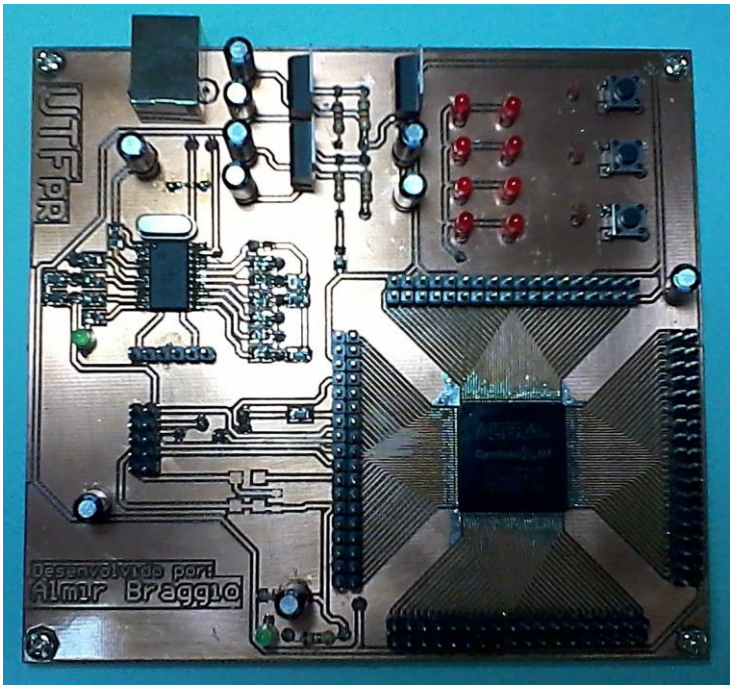


Figura 31 - Vista Superior PCB (12.4 cm x 11.35 cm)

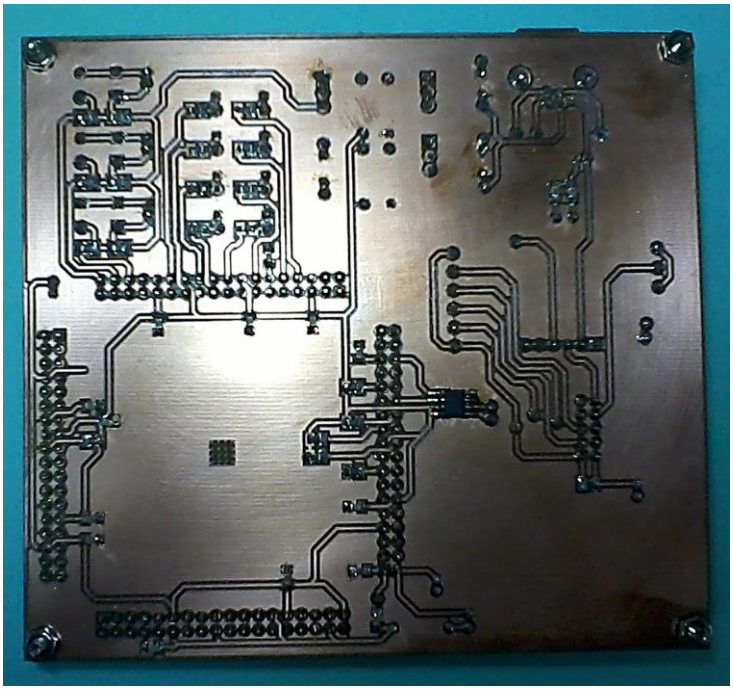


Figura 32 - Vista Superior PCB (12.4 cm x 11.35 cm)

3.1 TESTES

Pós-programação do microcontrolador, com a conexão do cabo USB (PCB/tipo B – Computador/tipo A), o sistema operacional em uso (*Microsoft® Windows 8*) já identifica o driver do USB Blaster emulado (Figura 33) (caso o driver não tenha sido instalado ou identificado, ele pode ser encontrado na pasta de instalação do software *Altera® Quartus II*, diretório *%\altera\13.0\quartus\drivers\usb-blaster*) (Figura 34).

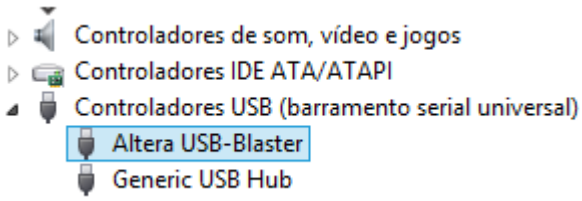


Figura 33 - Dispositivo USB-Blaster

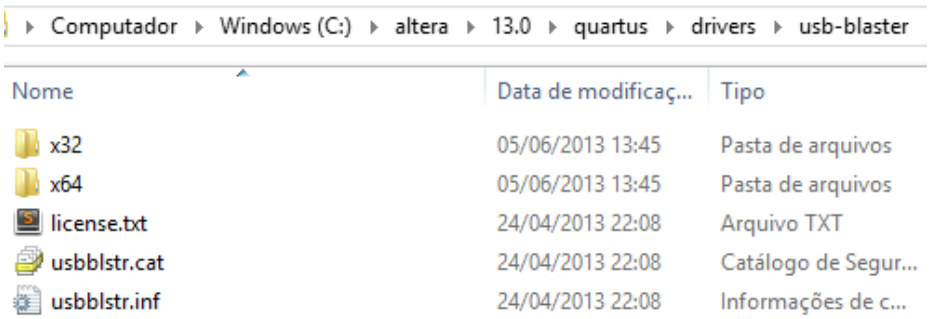


Figura 34 - Diretório do driver para USB-Blaster

Para início do design, em *Altera® Quartus II,* cria-se um novo projeto utilizando o *Project Wizard* no *Altera® Quartus II* (versão em uso: 13.0 – 64bits), escolhendo um novo diretório para o projeto, nome de *top-level,* arquivos de projeto e bibliotecas e dispositivo em uso (*Cyclone III - EP3C5E144C8*) (Figura 35).

Após a análise e síntese (*Analysis & Synthesis)* do projeto descrito (Figura 7 - Fluxograma das etapas de projeto), em VHDL, Verilog ou outra linguagem, faz-se a denominação dos pinos utilizados (*Pin Planner*) (Figura 36) e na sequência, a compilação do projeto (*Compilation)*. A compilação gerará um arquivo SOF que é o formato usado em programação JTAG.

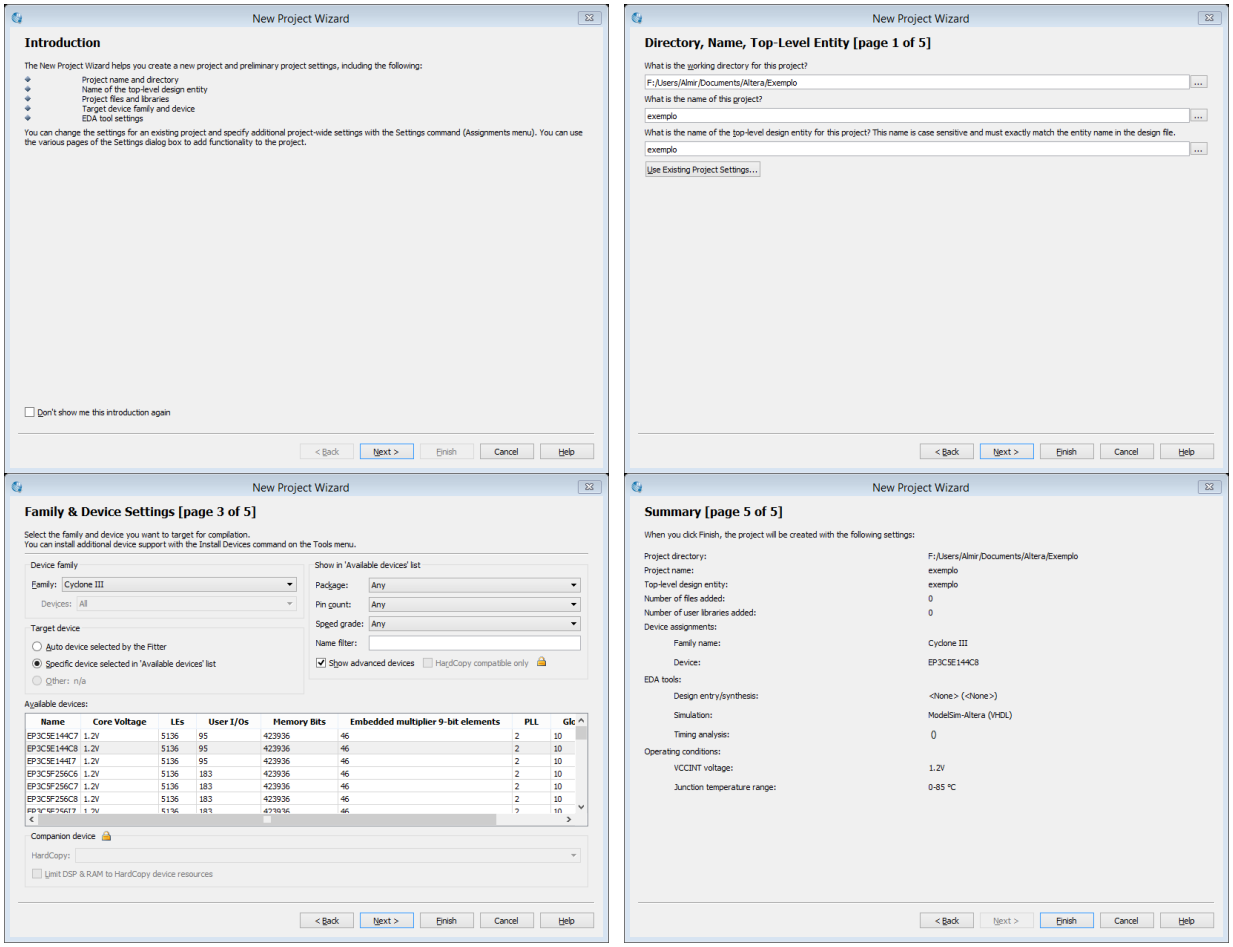


Figura 35 - Screenshots Project Wizard

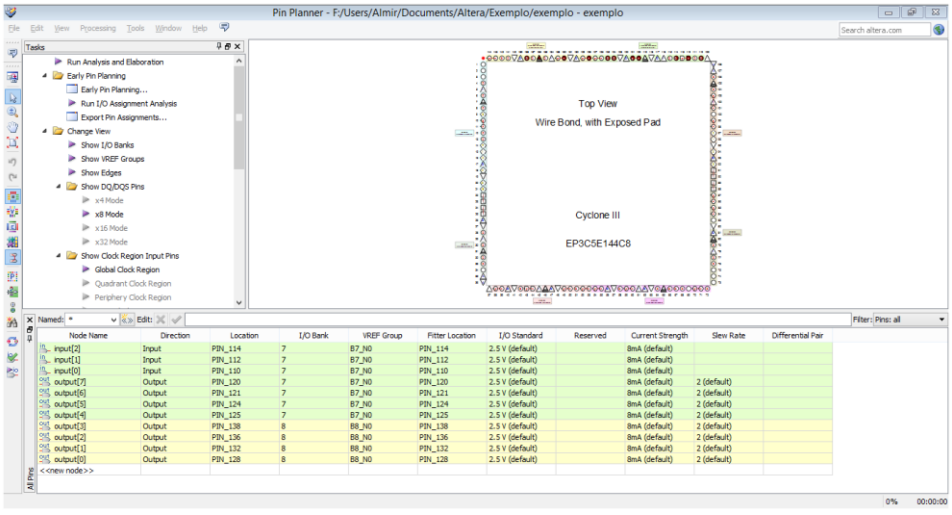


Figura 36 - Pin Planner

Na tela de programação (*Programmer)* (Figura 37) escolhe-se o hardware USB-Blaster (*Hardware Setup*), o arquivo SOF e inicia-se a configuração do FPGA.

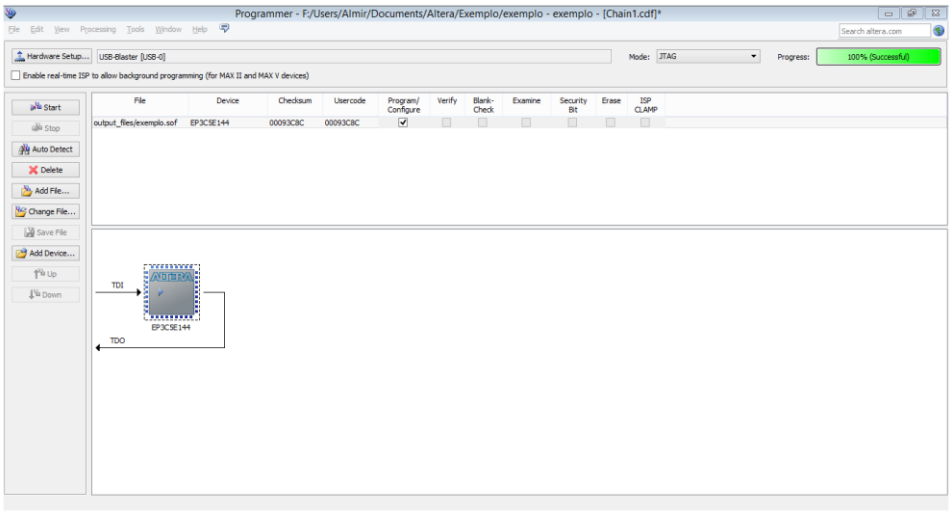


Figura 37 - Programmer

Para informações mais detalhadas referentes ao uso do *Quartus II* usando programação em VHDL, ver fonte *Altera Quartus II Introduction Using VHDL Design.*

Para a programação da memória EPCS, deve-se criar um *megafunction SFL* (*Tools -> MegaWizard Plug-in Manager*) (Figura 38), compilar o projeto e converter o arquivo SOF para o formato JIC (*File -> Convert Programming Files*) (Figura 39).

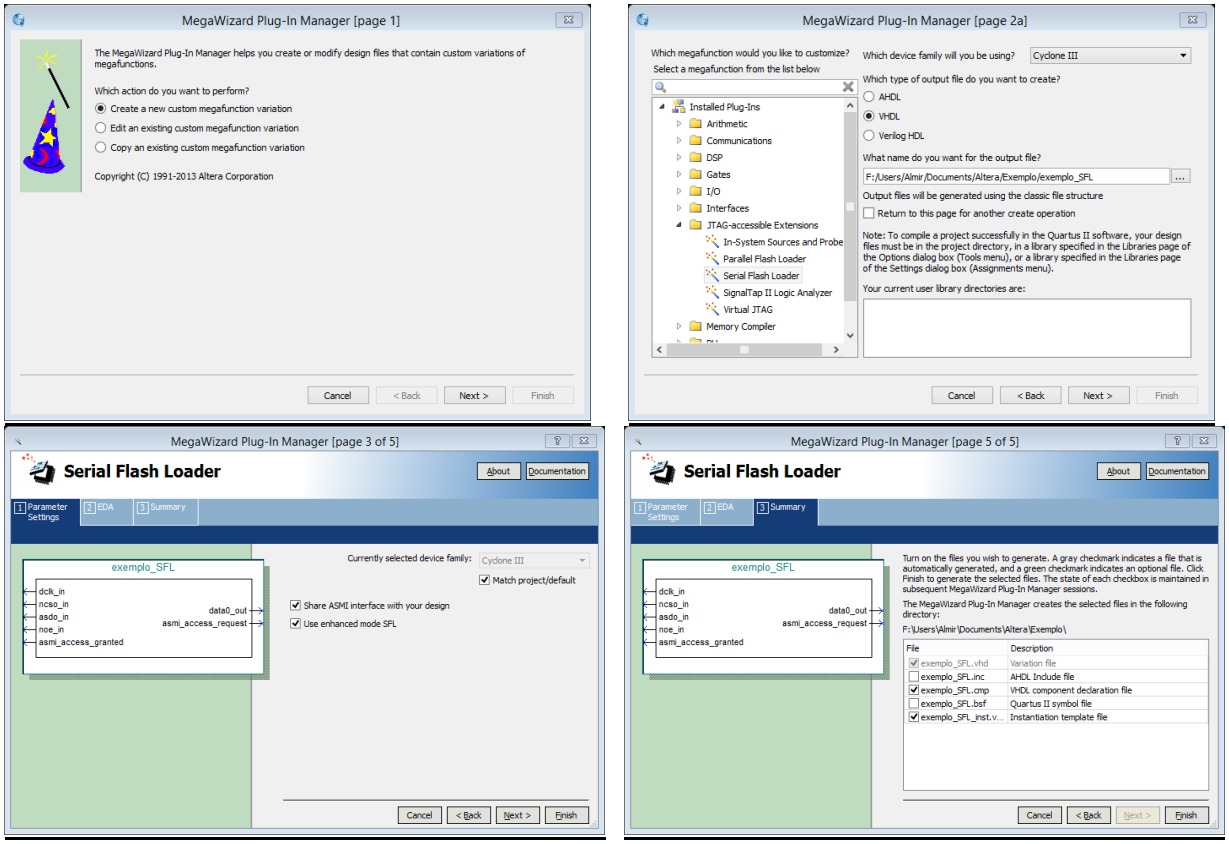


Figura 38 - Megafunction SFL

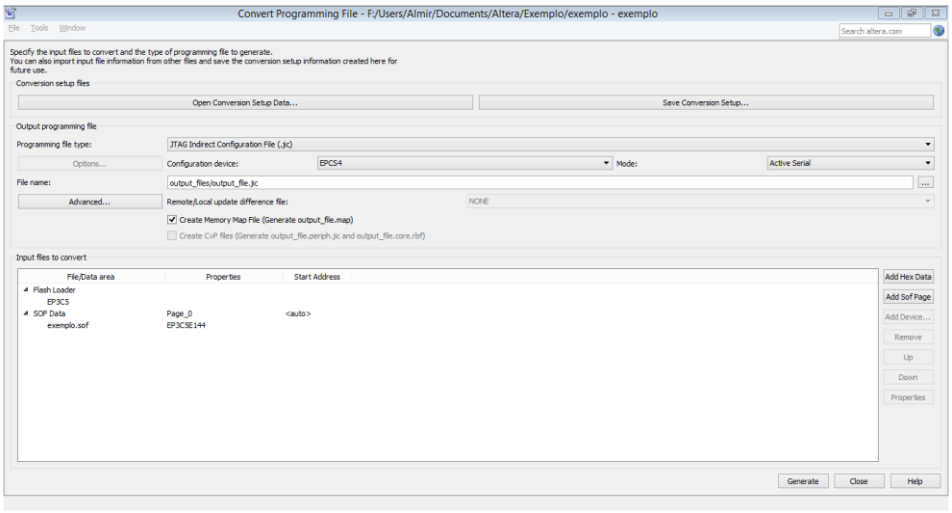


Figura 39 - Converter SOF para JIC

Com a geração do arquivo JIC, para programação do dispositivo de memória serial, na tela de programação, deve-se carregar o arquivo JIC (*Add File*) e marcar a caixa de seleção (*Program/Configure*) (Figura 40). Para outras configurações relacionadas à configuração via SFL, os guias devem ser consultados.

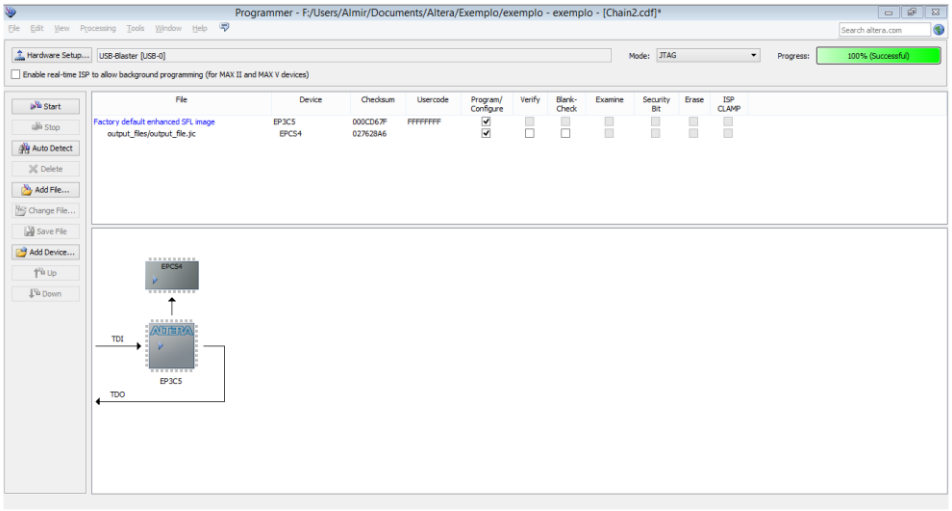


Figura 40 - Programmer SFL

Para informações mais detalhadas referentes à *Serial Flash Loader* com o *Quartus II*, ver fonte *Using the Serial FlashLoader with the Quartus II Software*.

Este teste, apenas como exemplo, pode ser tomado como base para quaisquer outros designs para configuração JTAG ou ativo serial com SFL, mas sempre amparado à documentação fornecida por *Altera*®.

4 CONSIDERAÇÕES FINAIS

Como forma de facilitar trabalhos posteriores, aqui, citam-se alguns pontos importantes em relação ao desenvolvimento de projetos de PCBs com FPGAs, que são apresentados no Quadro 1.

Descrição	Check?
Selecionar um dispositivo com base na densidade de elementos lógicos (LEs, memória, multiplicadores), recursos disponíveis (PLLs, pinos I/O) e encapsulamento. Considerar possibilidade de migração de dispositivo com relação à densidade;	
Caso possível, estimar o consumo de potência para projetar o fornecimento de tensão e solução de resfriamento, e reduzir o consumo total com baixa capacitância de entrada e baixos níveis de tensão;	
Utilizar pinos de I/O adequados para a aplicação e priorizar pinos que compartilham do mesmo banco, tensão de alimentação (VCCIO) e referência (VREF), quando em uso e em tensões distintas entre bancos;	
Selecionar um esquema de configuração suportado pelo dispositivo, verificando os tempos exigidos pelo projeto. Para modos AS, AP e PS, é possível a seleção do tempo de PoR;	
Para modo AS, escolher um EPCS apropriado para o tamanho do fluxo de dados máximo do dispositivo selecionado;	
Optar por SFL para reduzir interfaces separadas de programação;	
Quando usar PLL, certificar-se dos valores suportados no datasheet;	
Verificar e conectar os pinos não utilizados de acordo com o seu comportamento, principalmente tratando-se dos pinos de configuração. Os pinos MSEL, nCONFIG, nSTATUS e CONF_DONE, são utilizados em todos os modos de configuração;	
Não deixar os pinos JTAG desconectados (interface JTAG tem prioridade em relação aos outros modos de configuração);	
Ligar os pinos MSEL diretamente a VCCA ou GND de acordo com o esquema de solução escolhido (não deixar estes pinos flutuando);	
Sempre verificar a estabilidade dos pinos de clock de configuração (TCK e DCLK), eles não podem ter ruídos;	
Quando possível, usar os pinos específicos para controle de design da aplicação: CLKUSR: usado para controlar a inicialização pós-configuração; INIT_DONE: usado para verificar o processo de inicialização; CRC_ERROR: usado para detecção de erros em *cyclical redundancy check*;	
Adicionar corretos valores de resistência série para sinais de configuração. Por exemplo, em modo AS, usar resistor de 25 Ohm;	
Usar resistores de pull-up e pull-down com valores recomendados na documentação;	
Verificar a correta conexão de todos os pinos de tensão e referência (inclusive o plano ground, pino 145, no encapsulamento EQFP), e os pinos de configuração;	
Ser cauteloso com descargas eletrostáticas e operar o dispositivo FPGA sempre em condições recomendadas;	

Quadro 1 - Informações que requerem atenção no desenvolvimento

As expressões aqui selecionadas, Quadro 1, mesmo que possam ser consideradas adicionais em relação ao proposto, foram observadas durante o desenvolvimento do protótipo em questão.

Além disso, aconselha-se a futuros projetistas, quando se baseando neste texto, utilizar materiais teóricos fornecidos pelo fabricante, independente do dispositivo ou marca.

Respeitando essas observações e a teoria aqui descrita, como *savoir-faire*[21], qualquer projeto futuro com FPGAs terá resultado positivo, somando-se às documentações fornecidas pelos fabricantes, e sucesso em replicações e aplicações futuras.

[21] *Savoir-Faire:* expressão francesa para definir o "*know how to do"*, ou o "saber como fazer".

REFERÊNCIAS

Altera Configuration Handbook. Altera Corporation, 2008. Disponível em: <www.altera.com/literature/hb/cfg/config_handbook.pdf>. Acesso em: 25 jun. 2013.

Altera Quartus II Introduction Using VHDL Design. Altera Corporation, 2005. Disponível em: <ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital_Logic/tut_quartus_intro_vhdl.pdf>. Acesso em: 25 jun. 2013.

Altera Serial Configuration (EPCS) Devices Datasheet. Altera Corporation, 2012. Disponível em: <http://www.altera.com/literature/hb/cfg/cyc_c51014.pdf>. Acesso em: 01 ago. 2013.

Configuring Altera FPGAs. Altera Corporation, 2013. *(Online Training)*

Cyclone III Design Guidelines. Altera Corporation, 2013. Disponível em: <http://www.altera.com/literature/an/an466.pdf>. Acesso em: 02 jan. 2014.

Cyclone III Device Handbook. Volume I. Altera Corporation, 2012. Disponível em: <http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf>. Acesso em: 23 jan. 2014.

COSTA, Cesar da. **Projetos de Circuitos Digitais com FPGA**. 1 ed. São Paulo: Érica, 2009.

IEEE 1149.1 JTAG Boundary-Scan Testing in Altera Devices. Altera Corporation, 2005. Disponível em: < http://www.altera.com/literature/an/an039.pdf>. Acesso em: 01 ago. 2013.

KILTS, Steve. **Advanced FPGA Design: Architecture, Implementation and Optimization**. John Wiley & Sons, Inc, 2007.

MAXFIELD, Clive “Max”. **The Design Warrior’s Guide to FPGAs**. Elsevier, 2004.

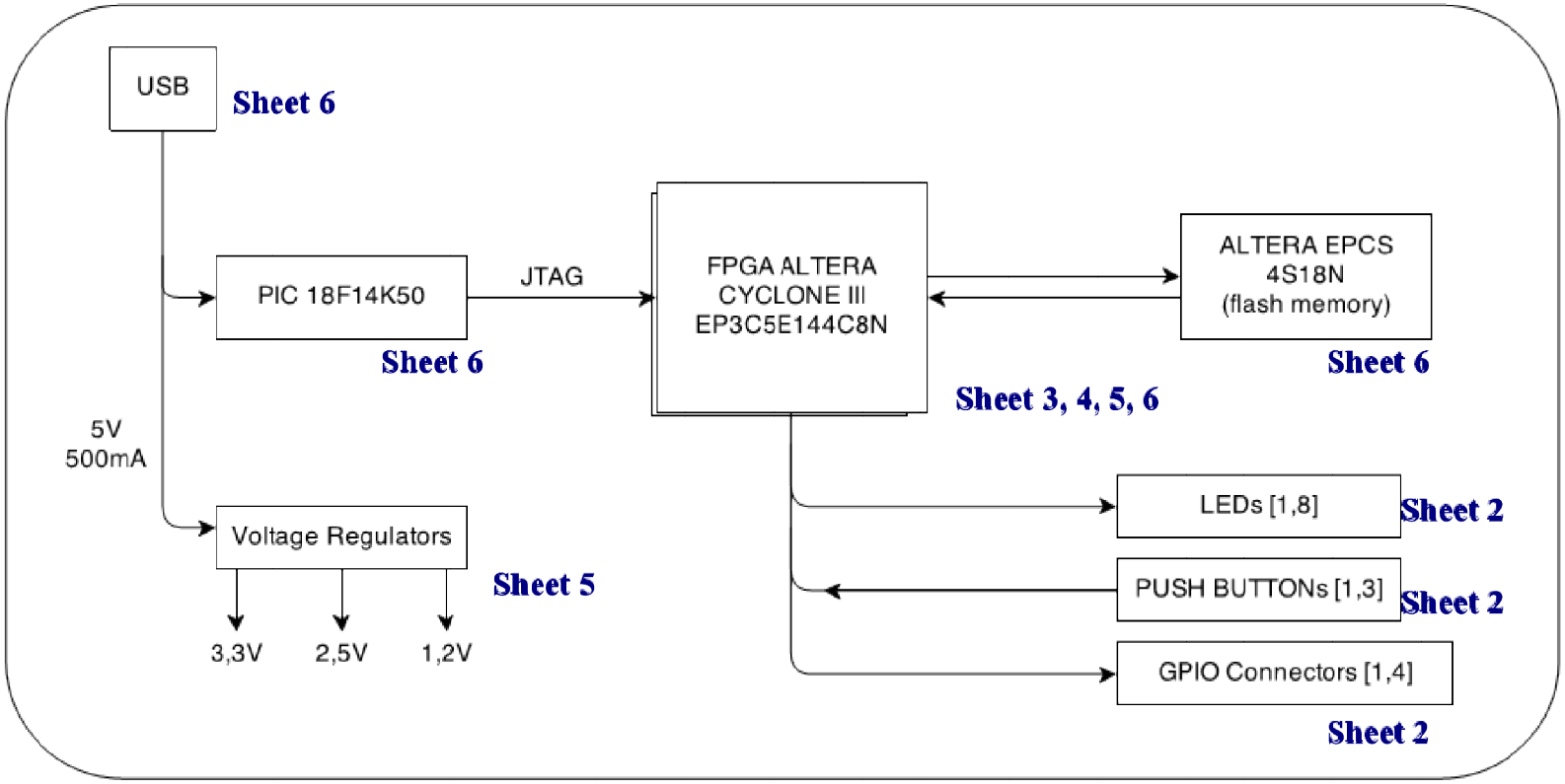
MÉLO, E.; A.DIAS, Roberto; STEINBACH, R. **FPGA PARA TODOS: Um Projeto para a Disseminação da Tecnologia de Lógica Programável**. In: XXXIX Congresso Brasileiro de Educação em Engenharia – COBENGE 2011, 2011.

MILLER, Warren. **The Next Market for FPGAs**. 2013. Disponível em: <http://www.eetimes.com/author.asp?section_id=36&doc_id=1318866>. Acesso em: 11 ago. 2013.

OJIKA, Satoshi. **USB-Blasterもどきの製作 (Desenvolvimento de uma imitação do USB-Blaster)**. 2012. Disponível em: <http://www.sa89a.net/mp.cgi/ele/ub.htm>. Acesso em: 20 ago. 2013.

Using the Serial FlashLoader with the Quartus II Software. Altera Corporation, 2012. Disponível em: <http://www.altera.com/literature/an/an370.pdf>. Acesso em: 20 jan. 2014.

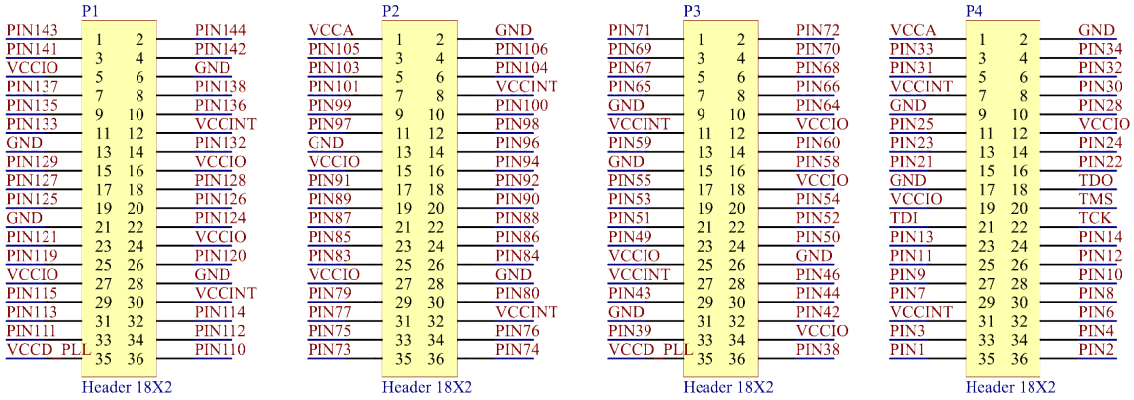
APÊNDICE A - Circuito Eletrônico (Esquemáticos)

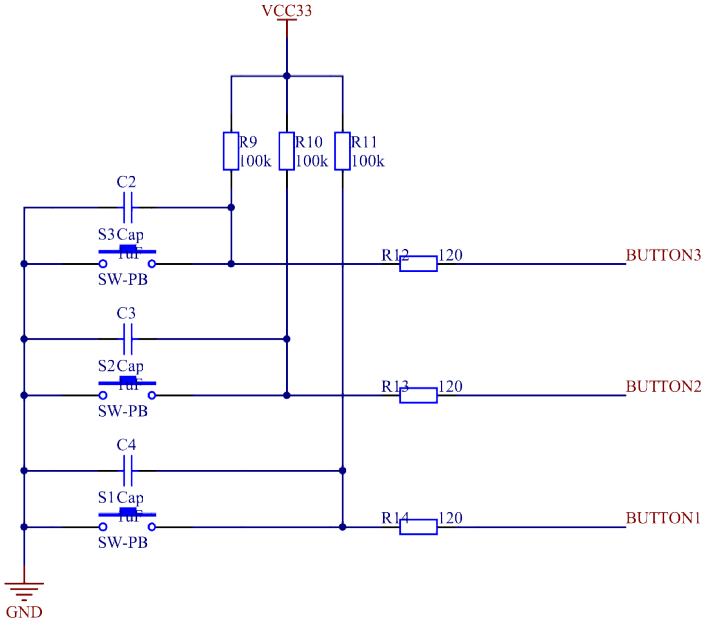


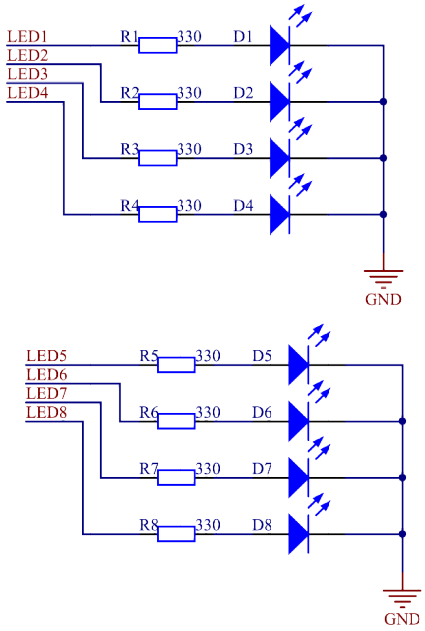
Sheet1.SchDoc

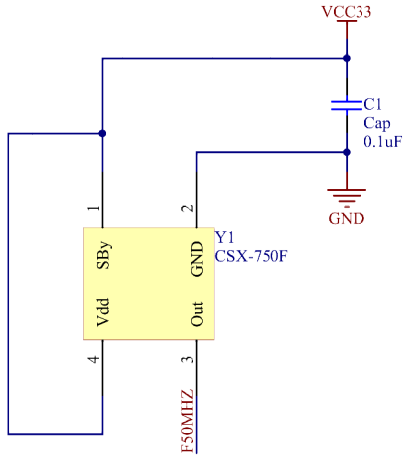
Title:	Placement
Revision:	0.2
Author:	Almir Braggio
Date:	02/02/2014
Sheet:	1 of 6

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ





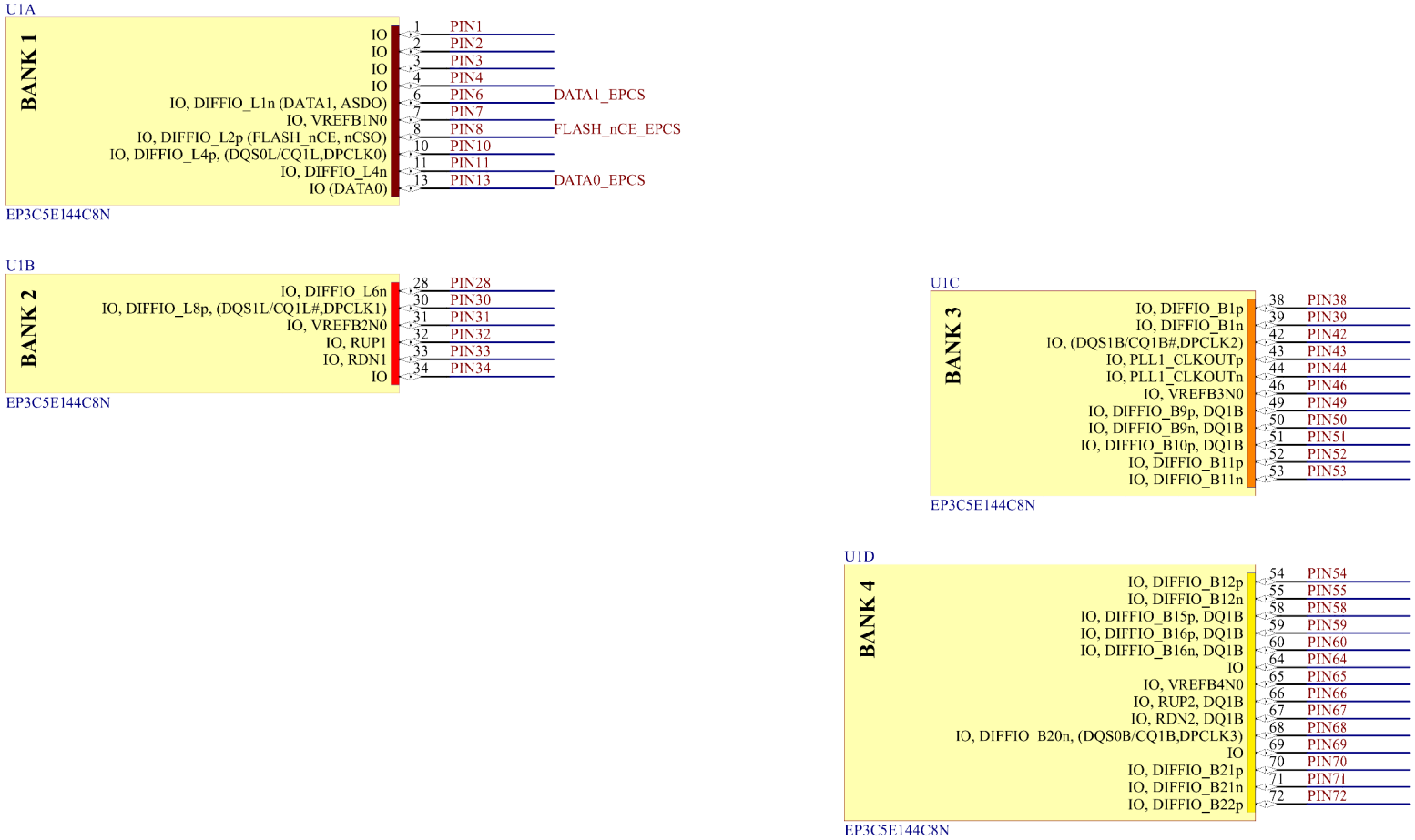




Sheet2.SchDoc

Title:	Headers - Buttons - LEDs - Clock
Revision:	0.2
Author:	Almir Braggio
Date:	02/02/2014
Sheet:	2 of 6

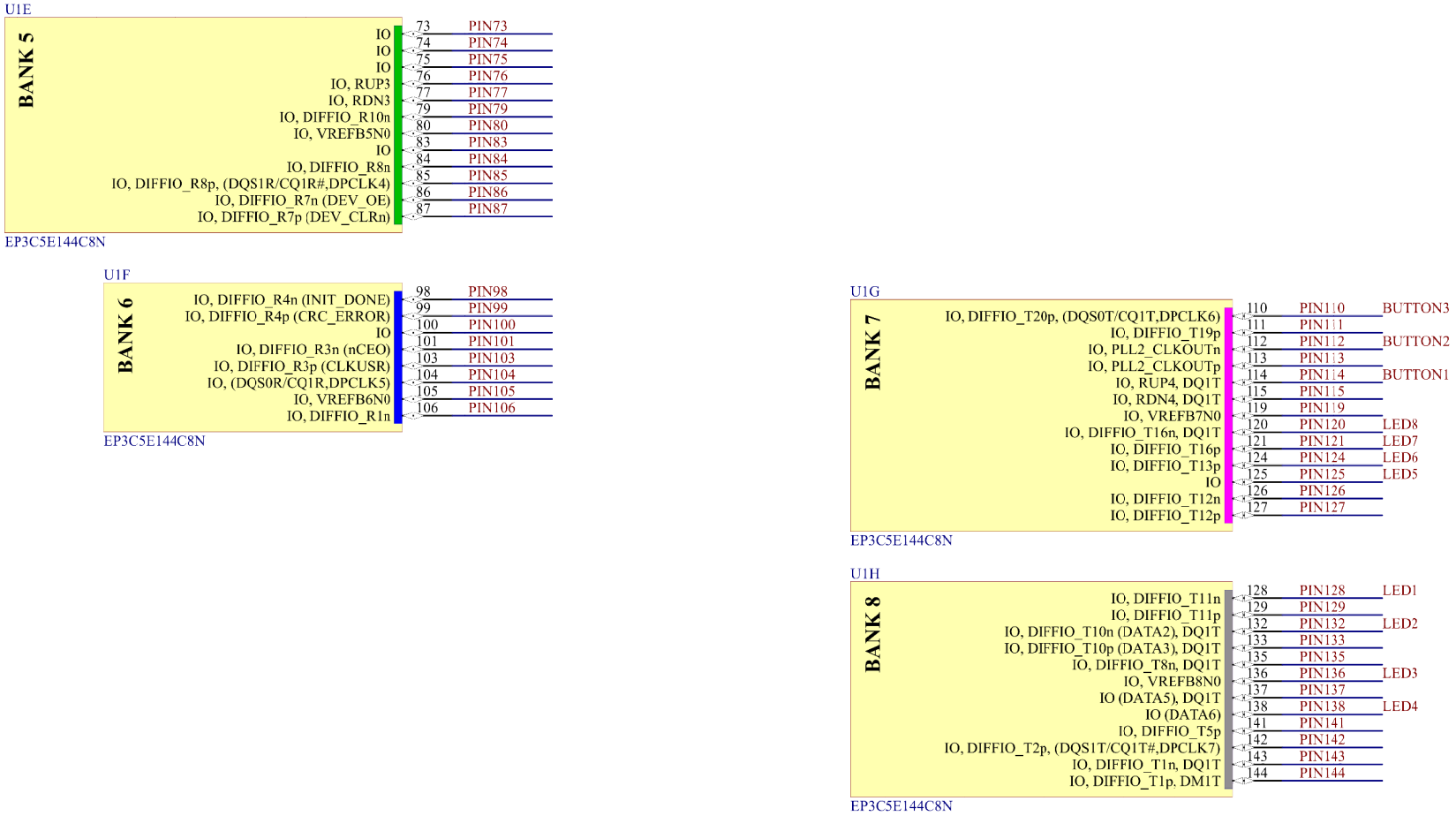
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ



Sheet3.SchDoc

Title:	Bank 1 ... 4
Revision:	0.2
Author:	Almir Braggio
Date:	02/02/2014
Sheet:	3 of 6

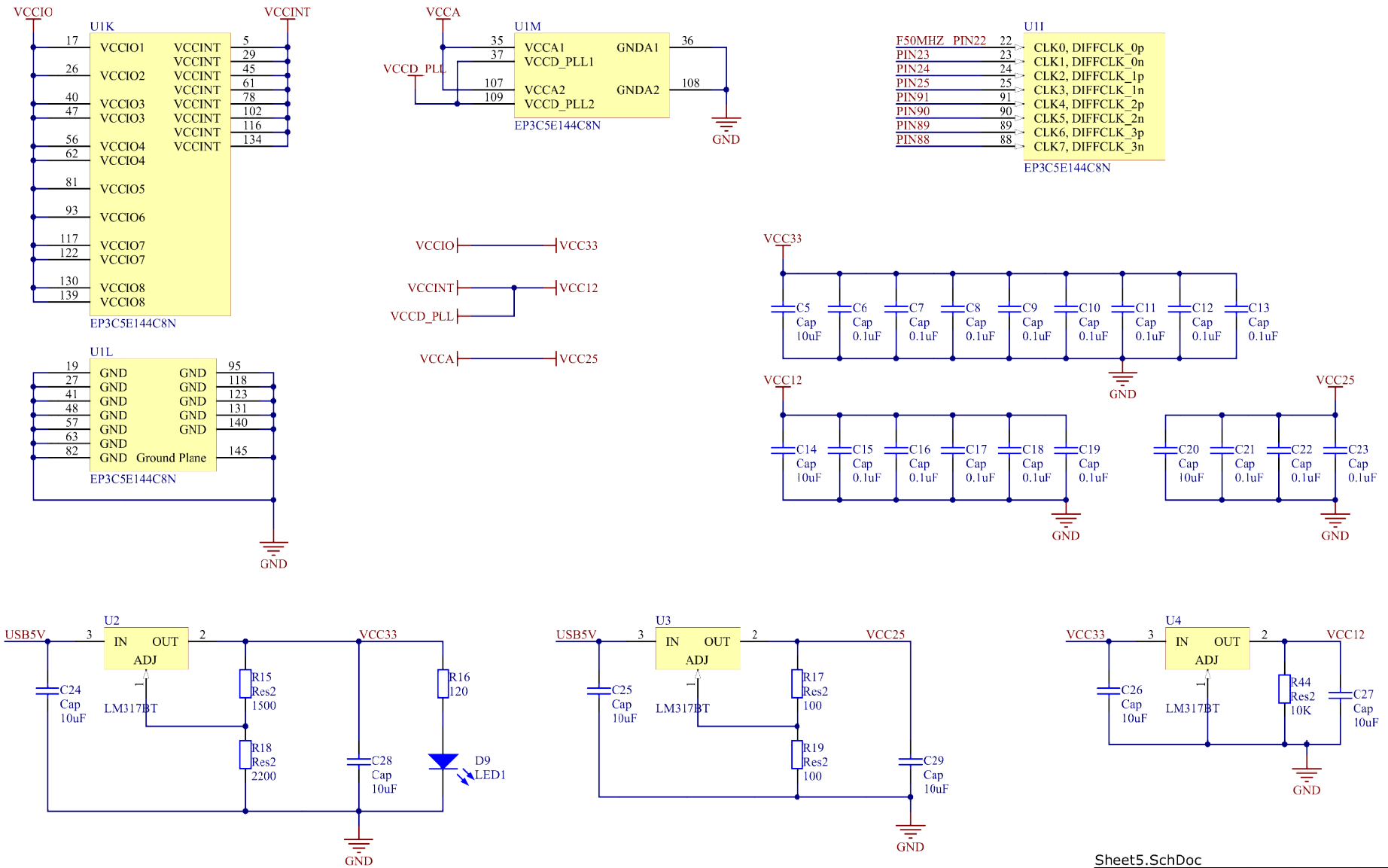
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ



Sheet4.SchDoc

Title:	Bank 5 ... 8
Revision:	0.2
Author:	Almir Braggio
Date:	02/02/2014
Sheet:	4 of 6

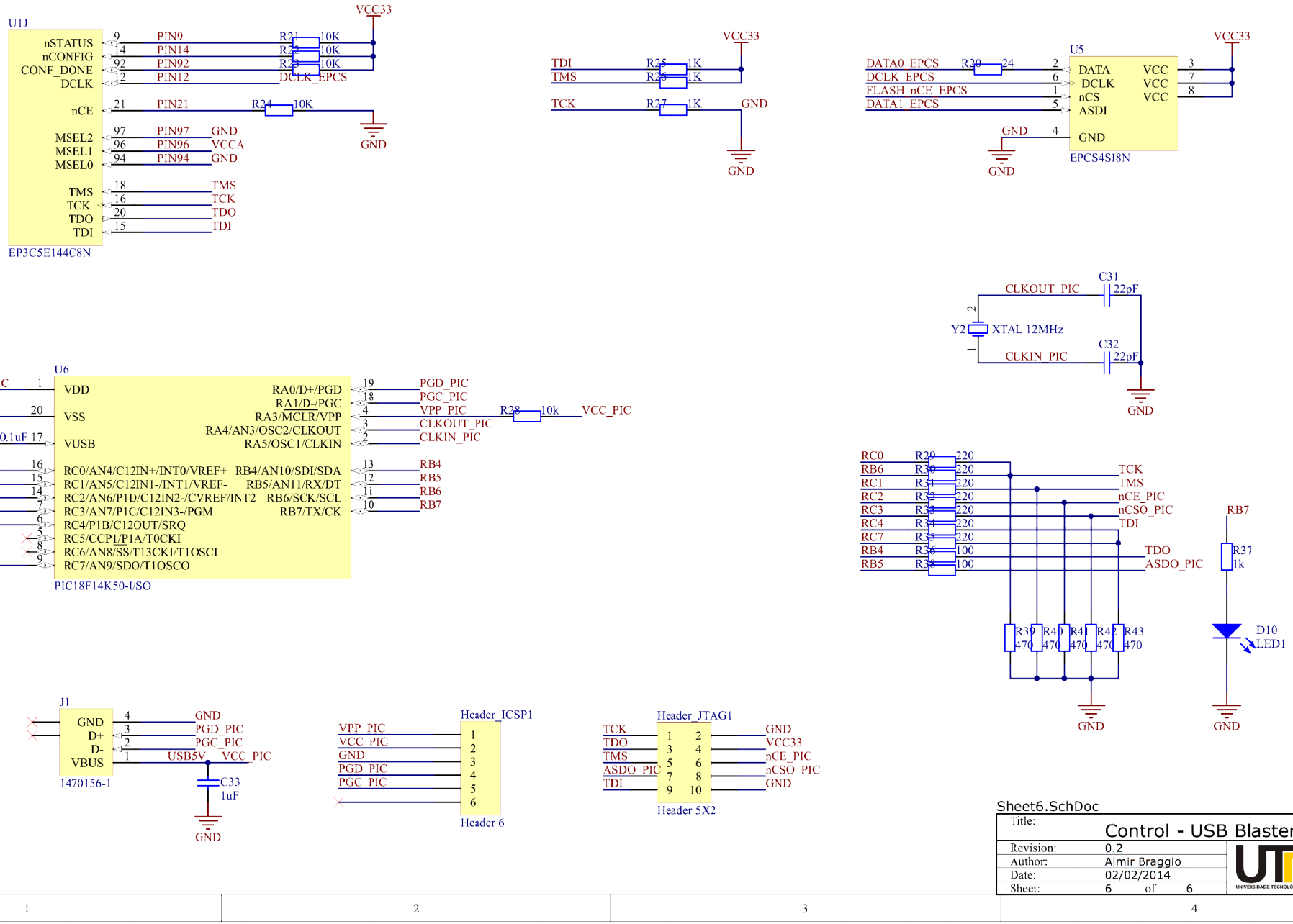
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ



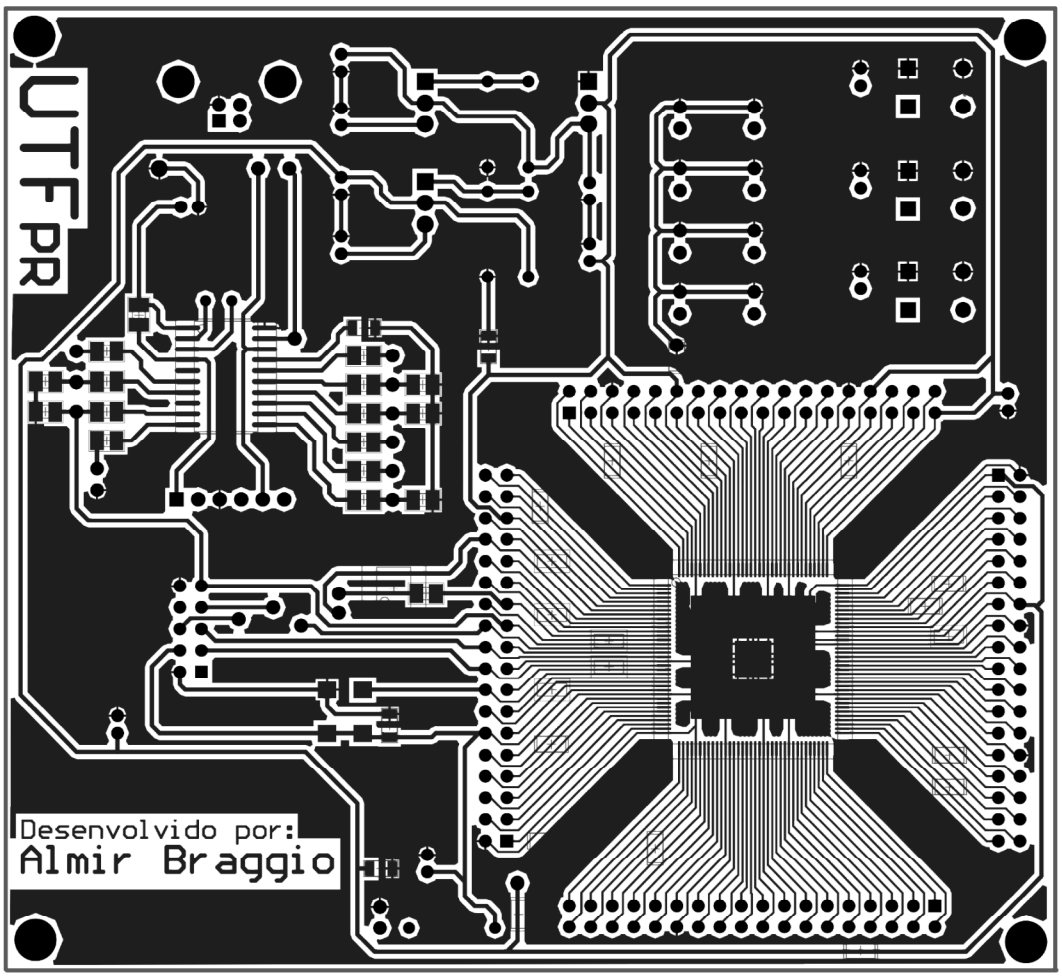
Sheet5.SchDoc

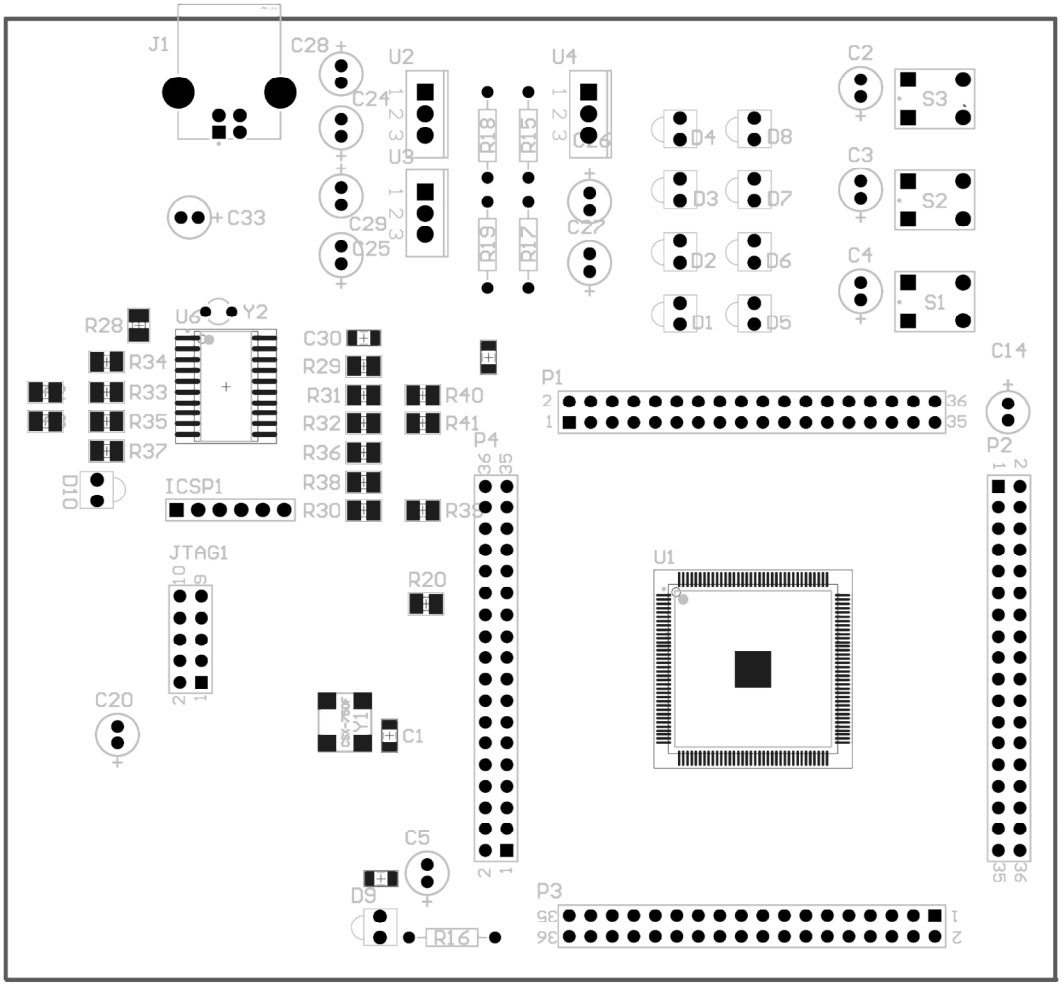
Title:	Power
Revision:	0.2
Author:	Almir Braggio
Date:	02/02/2014
Sheet:	5 of 6

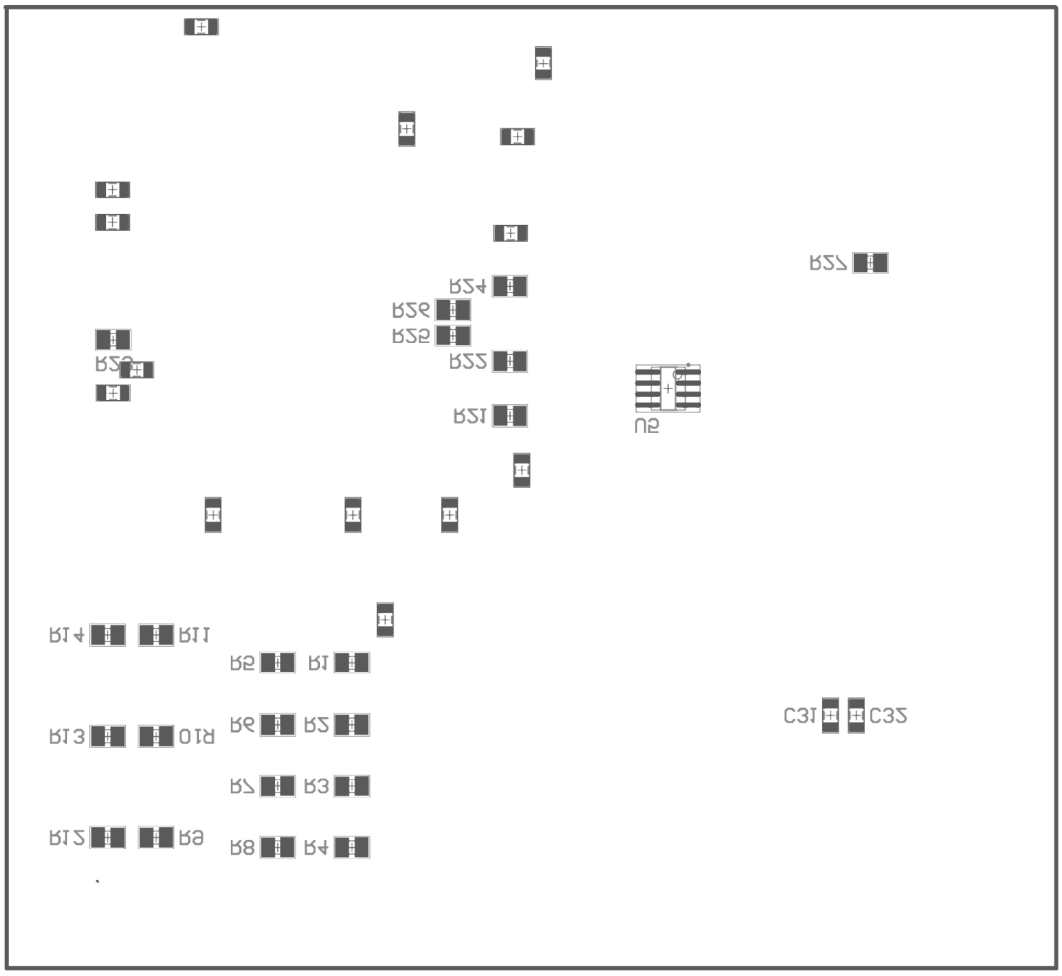
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ


U1J
nSTATUS
nCONFIG
CONF_DONE
DCLK
nCE
MSEL2
MSEL1
MSEL0
TMS
TCK
TDO
TDI
PIN9
PIN14
PIN92
PIN12
PIN21
PIN97
PIN96
PIN94
R21 10K
R22 10K
R23 10K
DCLK_EPCS
R24 10K
GND
VCCA
VCC33
EP3C5E144C8N
TDI R25 1K
TMS R26 1K
TCK R27 1K
U5
DATA0_EPCS R20 24
DCLK_EPCS
FLASH_nCE_EPCS
DATA1_EPCS
DATA
DCLK
nCS
ASDI
VCC
EPCS4SI8N
C31 22pF
CLKOUT_PIC
Y2 XTAL 12MHz
C32 22pF
CLKIN_PIC
U6
VCC_PIC
VDD
VSS
VUSB
C30 0.1uF
RA0/D+/PGD
RA1/D-/PGC
RA3/MCLR/VPP
RA4/AN3/OSC2/CLKOUT
RA5/OSC1/CLKIN
PGD_PIC
PGC_PIC
VPP_PIC R28 10k VCC_PIC
CLKOUT_PIC
CLKIN_PIC
RC0/AN4/C12IN+/INT0/VREF+
RC1/AN5/C12IN1-/INT1/VREF-
RC2/AN6/P1D/C12IN2-/CVREF/INT2
RC3/AN7/P1C/C12IN3-/PGM
RC4/P1B/C12OUT/SRQ
RC5/CCP1/P1A/T0CKI
RC6/AN8/SS/T13CKI/T1OSCI
RC7/AN9/SDO/T1OSCO
RB4/AN10/SDI/SDA
RB5/AN11/RX/DT
RB6/SCK/SCL
RB7/TX/CK
RB4
RB5
RB6
RB7
PIC18F14K50-I/SO
RC0 R29 220
RB6 R30 220
RC1 R31 220
RC2 R32 220
RC3 R33 220
RC4 R34 220
RC7 R35 220
RB4 R36 100
RB5 R37 100
TCK
TMS
nCE_PIC
nCSO_PIC
TDI
TDO
ASDO_PIC
R39 470
R40 470
R41 470
R42 470
R43 470
RB7
R37 1k
D10 LED1
GND
J1
GND
D+
D-
VBUS
1470156-1
USB5V
VCC_PIC
C33 1uF
Header_ICSP1
VPP_PIC
VCC_PIC
PGD_PIC
PGC_PIC
Header 6
Header_JTAG1
TCK
TDO
TMS
ASDO_PIC
TDI
VCC33
nCE_PIC
nCSO_PIC
Header 5X2
Sheet6.SchDoc
Title: Control - USB Blaster (PIC)
Revision: 0.2
Author: Almir Braggio
Date: 02/02/2014
Sheet: 6 of 6
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

APÊNDICE B - Placa de Circuito Impresso (Arte Final)


UTFPR
Desenvolvido por:
Almir Braggio


J1
C28
U2
U4
C2
C24
U3
C33
C29
C25
R18
R15
R19
R17
C22
D4
D8
D3
D7
D2
D6
D1
D5
S3
S2
S1
C3
C4
R28
U6
Y2
C30
R29
R31
R32
R36
R38
R30
R34
R33
R35
R37
R40
R41
R39
P1
P4
C14
P2
ICSP1
JTAG1
R20
U1
C20
Y1
C1
C5
D9
R16
P3


R22
R24
R25
R23
R21
U5
C31
C32
R14
R11
R12
R9
R13
R10
R8
R7
R6
R5
R4
R3
R2
R1