

# Tutorial QFLOW

**Ruhan Conceição<sup>1</sup>**  
**Lisandro Luiz da Silva<sup>1</sup>**

<sup>1</sup>Centro de Desenvolvimento Tecnológico – Universidade Federal de Pelotas (UFPel)  
Caixa Postal 354 – 96.010-900 – Pelotas – RS – Brasil  
{radconceicao, lldsilva@inf}.ufpel.edu.br

## 1. Introdução

Um fluxo de síntese digital é um conjunto de ferramentas e métodos utilizados para transformar um projeto de circuito descrito em uma linguagem de alto nível como Verilog ou VHDL em um circuito físico. Qflow é um conjunto de diversas ferramentas que possui essa transformação, ela possui ferramentas para a síntese de circuitos digitais a partir de arquivo Verilog que acaba gerando o layout físico. Este tutorial explica o processo de instalação juntamente com a demonstração de um exemplo.

## 2. Conjunto de Ferramentas Qflow

**Yosys** = esta é uma ferramenta de síntese capaz de lidar com todo o tipo de sintaxe Verilog, e pode sintetizar grandes projetos devido à sua versatilidade. É uma ferramenta que irá sintetizar tudo, desde decodificadores até microprocessadores com facilidade e precisão.

**GrayWolf** = ferramenta desenvolvida na Universidade de Yale, foi distribuída com código aberto por um tempo. Ferramenta que determina uma estimativa aproximada do roteamento necessário, e tenta colocar todas as células em um bloco, ordenando-as para minimizar a quantidade total de fiação que liga todos os pinos.

**Qrouter** = A etapa final na criação de um layout de células é o detalhe da rota, onde descreve-se exatamente como a fiação física deve ser gerada para conectar todos os pinos no design.

A cadeia de ferramentas qflow é perfeitamente capaz de lidar com os subsistemas digitais que possuem muitos chips, processamento de sinal, unidades de lógica e aritmética. As primeiras versões do fluxo digital de qflow foram usadas para criar circuitos digitais utilizados em circuitos integrados comerciais de alto desempenho. Qflow é um fluxo de projeto de síntese digital completa, usa software open-source e bibliotecas de células padrão de código aberto.

O fluxo de síntese do Qflow é demonstrado na Figura 1. Para maiores informações sobre seu fluxo acesse o link <http://opencircuitdesign.com/qflow/>.

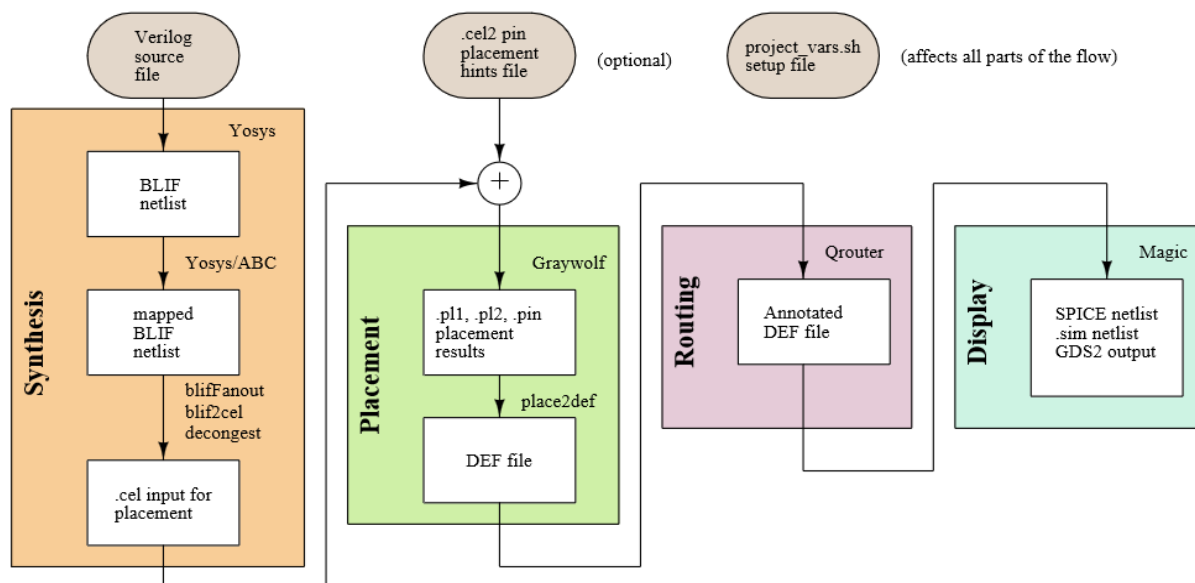


Figura 1 – Fluxo do Qflow

### 3. Instalação Qflow

Para a instalação do Qflow o usuário deverá baixar todos os arquivos no link <https://github.com/radc/qflow>. Os arquivos baixados são:

- qflow-org (diretório)
- showResults.py (script para obtenção do número de gates e frequência)

Após salve o diretório “qflow-org” na pasta home (pasta pessoal), dentro desse diretório existe um arquivo chamado “install.sh”. Para a instalação da ferramenta Qflow precisa-se instalar diversos pacotes. Para isso verificou-se a necessidade da criação de um script (install.sh) que instala e configura todos os pacotes necessários. Esse script instala pacote por pacote, para instalar um pacote o usuário deverá apertar “enter”. Essa instalação de pacote por pacote é necessária para saber se está ocorrendo a instalação correta de todos os pacotes, ou seja, se ocorrer algum erro na instalação o usuário saberá qual pacote ocorreu o erro.

Abaixo na Figura 2 e Figura 3 estão sendo demonstrados a instalação do primeiro e último pacote.

No diretório “qflow-org”, para executar o script “install.sh” escreva no terminal:

**\$/install.sh**

```

aluno@aluno-VirtualBox: ~/qFlow-org
aluno@aluno-VirtualBox:~/qFlow-org$ ./install.sh
proximo: apt-get install build-essential
  
```

Figura 2 – Primeiro pacote a ser instalado pelo “install.sh”

```

aluno@aluno-VirtualBox: ~/qFlow-org
for target in osu035 osu050; do \
  (cd $target ; make install) ;\
done
make[2]: Entrando no diretório `/home/aluno/qFlow-org/qflow-1.0.96/tech/osu035'
Installing osu035 tech files
/usr/bin/install -c -d /usr/local/share/qflow/tech/osu035
for target in osu035.genlib osu035.super osu035.par osu035_stdcells.lef osu035.s
h osu035_stdcells.sp osu035.magicrc osu035.prm gate.cfg SCN4M_SUBM.20.tech osu03
5_stdcells.lib osu035_stdcells.v; do \
  /usr/bin/install -c $target /usr/local/share/qflow/tech/osu035 ;\
done
make[2]: Saindo do diretório `/home/aluno/qFlow-org/qflow-1.0.96/tech/osu035'
make[2]: Entrando no diretório `/home/aluno/qFlow-org/qflow-1.0.96/tech/osu050'
Installing osu050 tech files
/usr/bin/install -c -d /usr/local/share/qflow/tech/osu050
for target in osu050.genlib osu050.super osu050.par osu050_stdcells.lef osu050.s
h osu050_stdcells.sp osu050.magicrc osu050.prm gate.cfg SCN3ME_SUBM.30.tech osu0
5_stdcells.lib osu05_stdcells.v; do \
  /usr/bin/install -c $target /usr/local/share/qflow/tech/osu050 ;\
done
make[2]: Saindo do diretório `/home/aluno/qFlow-org/qflow-1.0.96/tech/osu050'
make[1]: Saindo do diretório `/home/aluno/qFlow-org/qflow-1.0.96/tech'
proximo: apt-get install tcsh

```

Figura 3 – Último pacote a ser instalado pelo “install.sh”

Todos os pacotes instalados pelo “install.sh” são demonstrados pela Tabela 1.

Tabela 1 – Todos os pacotes instalados pelo script “install.sh”

build-essential	clang	bison	flex	libreadline-dev	gawk
tcl-dev	libffi-dev	git	mercurial	graphviz	xdot
pkg-config	python3	iverilog	autoconf	gperf	cmake
m4	csh	libx11-dev	libncurses-dev	tcl-dev tk-dev	blt-dev
freeglut3	freeglut3-dev	freeglut3-dbg	tcsh		

Após a instalação de todos os pacotes necessários, precisa-se corrigir bugs pendentes da instalação. Durante a execução do Qflow, alguns arquivos “.tcl” são utilizados, os quais são interpretados pelo software “tclsh”. Entretanto, estes arquivos “acreditam” que o executável do tclsh está na pasta “/bin/tclsh”, ao passo que este se encontra em “/usr/bin/tclsh”. Para solucionar o problema é necessário a criação de um link na pasta “/bin/” do arquivo “/usr/bin/tclsh”. O comando abaixo realiza essa criação:

**\$ sudo ln -s /usr/bin/tclsh /bin/**

#### 4. Vhdl para Verilog (Vhdl2Verilog)

Como dito anteriormente Qflow possui ferramentas para a síntese de circuitos digitais a partir de um arquivo Verilog que acaba gerando o layout físico. Como usa-se arquivos Vhdl na disciplina, verificou-se a necessidade de obter um conversor de Vhdl para Verilog. Através do link <http://www.edautils.com/> pode-se fazer o download do tradutor do arquivo Vhdl para Verilog.

Este tradutor é gratuito, entretanto o tradutor deve ser registrado para que possa ser utilizado. Através do link <http://form.jotform.me/form/30917447754462> o usuário deverá preencher um formulário e seguir alguns passos para a obtenção do tradutor

juntamente com a licença (que é encaminhada através do e-mail fornecido no formulário).

Após receber a licença por e-mail e baixar o tradutor pelo site mencionado acima, através dos passos seguidos pelos e-mails recebidos o usuário poderá realizar a tradução de Verilog para Vhdl através de alguns passos a serem tomados.

Entre na pasta **vhdl2verilog/10OCT2015** e digite os seguintes comandos:

```
$ export EDAUTILS_ROOT=$PWD
$ set path=$EDAUTILS_ROOT/bin $path
$ export EDAUTILS_LICENSE_KEY=<sua licença>
$ ./bin/vhdl2verilog -in <arquivo_entrada.vhd> -out
<arquivo_saída.v> -top <entidade_top_level (sem .vhd)>
```

A tradução do Vhdl para Verilog estará disponível no local onde o usuário especificou o “arquivo.v”, este arquivo deverá ser copiado/recortado e inserido na pasta “source”(que será discutida mais adiante).

## 5. Usando Qflow

O primeiro passo para a síntese de um circuito é ter um diretório do projeto (diretório raiz) que será o espaço de trabalho para a utilização do Qflow, esse diretório pode possuir qualquer nome. Dentro do diretório do projeto deverá conter três subdiretórios. São eles:

- source = Contém o código-fonte Verilog
- synthesis = Contém arquivos de trabalho e de saída Verilog RTL
- layout = Contém arquivos de trabalho e saída de arquivo DEF

O arquivo de origem verilog (arquivo.v) deve ser colocado no subdiretório “source”, devido ao fato da organização dos arquivos que são gerados. A Figura 4 demonstra as pastas criadas para a utilização do Qflow.

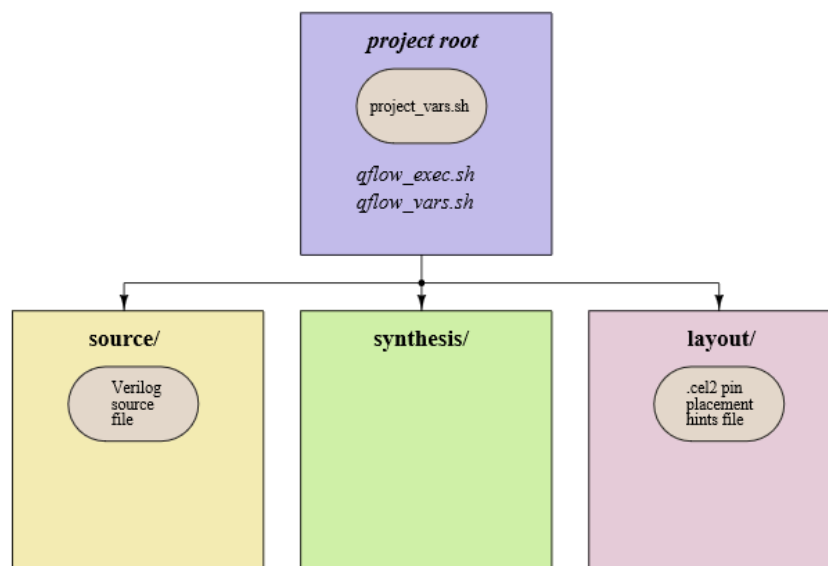


Figura 4 – Pastas criadas para a utilização do Qflow

Na pasta raiz do projeto, insira o comando abaixo para a execução do Qflow:

**\$ qflow build <arquivo.v>**

Se nenhum erro ocorrer, a síntese do circuito foi realizada com sucesso, os resultados são encontrados no arquivo “**synth.log**”. Após a execução do Qflow diversos arquivos são criados. Entre eles estão:

- **synth.log**: contém informações geradas durante o processo de síntese da arquitetura;
- **project\_vars.sh**: seta variáveis a serem utilizadas no projeto;
- **qflow\_vars.sh**: seta variáveis a serem utilizadas pelo Qflow;
- **qflow\_exec.sh**: contém o fluxo de execução do projeto.

Após a realização da síntese da arquitetura, o seguinte comando gera os resultados de tempo para a arquitetura do projeto, esse comando também deverá ser executado na pasta raiz do projeto.

**\$ qflow sta <arquivo.v>**

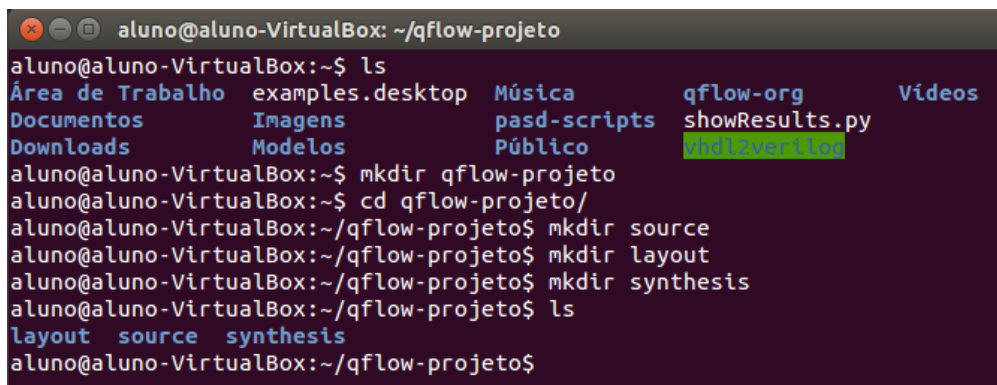
Para a extração dos resultados da área e tempo da arquitetura utiliza-se um script desenvolvido em python para essa finalidade (coleta automática dos dados arquiteturais e tempo). Esse script python quando executado deve-se passar como parâmetro o arquivo **qflow\_vars.sh**, conforme o comando demonstrado abaixo:

**\$/showResults.py qflow\_vars.sh**

## 6. Exemplo

A seguir será demonstrada primeiramente a tradução de um arquivo Vhdl para Verilog, após com o “arquivo.v” gerado, executa-se o qflow, demonstra-se o passo a passo dessa execução.

**Criação dos diretórios para o uso do Qflow:** A figura 5 demonstra a criação das pastas (diretórios) necessárias para a utilização do Qflow.

A terminal window titled 'aluno@aluno-VirtualBox: ~/qflow-projeto' showing the following commands and output:

```
aluno@aluno-VirtualBox:~$ ls
Área de Trabalho  examples.desktop  Música          qflow-org       Vídeos
Documentos        Imagens           pasd-scripts    showResults.py
Downloads          Modelos           Público         vhd2verilog.log

aluno@aluno-VirtualBox:~$ mkdir qflow-projeto
aluno@aluno-VirtualBox:~$ cd qflow-projeto/
aluno@aluno-VirtualBox:~/qflow-projeto$ mkdir source
aluno@aluno-VirtualBox:~/qflow-projeto$ mkdir layout
aluno@aluno-VirtualBox:~/qflow-projeto$ mkdir synthesis
aluno@aluno-VirtualBox:~/qflow-projeto$ ls
layout  source  synthesis
aluno@aluno-VirtualBox:~/qflow-projeto$
```

Figura 5 – Pastas criadas para a utilização do Qflow

**Tradução Vhdl2Verilog:** O arquivo do tradutor Vhdl2Verilog já foi extraído anteriormente, percebe-se isso pela figura acima (Figura 5). Após a extração é necessário entrar na pasta “vhdl2verilog” e executar os comandos abaixo (Figura 6), lembrando de que o “arquivo.vhd” escolhido está dentro da pasta “examples/uart\_trans” com o nome “uart\_trns.vhd”.

```
aluno@aluno-VirtualBox: ~/vhdl2verilog/10OCT2015
aluno@aluno-VirtualBox:~/vhdl2verilog/10OCT2015$ ls
LICENSE.txt  README.txt  setup_env.bat  setup_env.sh  vhdl2verilog.log
aluno@aluno-VirtualBox:~/vhdl2verilog/10OCT2015$ export EDAUTILS_ROOT=$PWD
aluno@aluno-VirtualBox:~/vhdl2verilog/10OCT2015$ set path=$EDAUTILS_ROOT/bin $path
aluno@aluno-VirtualBox:~/vhdl2verilog/10OCT2015$ export EDAUTILS_LICENSE_KEY=8ic1LYmrS3@camaqua.ifsul.edu.br
aluno@aluno-VirtualBox:~/vhdl2verilog/10OCT2015$ ./bin/vhdl2verilog -in examples/uart_trans/uart_trns.vhd -out examples/uart_trans/uart_trans.v -top uart_trans
Initializing environment .....
vhdl2verilog: Note: 58000: Checking out LICENSE from the EDAUtils server( it may take a while ) ...
vhdl2verilog: Note: 58001: Completed LICENSE checkout ...
```

Figura 6 – Tradução de Vhdl para Verilog.

A Figura 7 é a continuação da Figura 6.

```
vhdl2verilog:          vhdl2verilog
vhdl2verilog:          Version: 10OCT2015-10/10/2015 Released on 10 Oct 2015
vhdl2verilog:          Support info: help@edautils.com
vhdl2verilog:
vhdl2verilog:          Copyright (c) 2012-2015 EDAUtils
vhdl2verilog:          All rights reserved.
vhdl2verilog:
vhdl2verilog:          Permission to use this software and its documentation for educational,
vhdl2verilog:          research and internal use within the organization and non-profit purposes
vhdl2verilog:          without a written agreement is hereby granted.
vhdl2verilog:
vhdl2verilog:          IN NO EVENT SHALL EDAUtils BE LIABLE TO ANY PARTY FOR
vhdl2verilog:          DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
vhdl2verilog:          DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE
vhdl2verilog:          OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF
vhdl2verilog:          EDAUtils HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
vhdl2verilog:          DAMAGE. EDAUtils SPECIFICALLY DISCLAIMS ANY WARRANTIES,
vhdl2verilog:          INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
vhdl2verilog:          MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE
vhdl2verilog:          SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND
vhdl2verilog:          EDAUtils HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE,
vhdl2verilog:          SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.
vhdl2verilog:          USER SHALL NOT, DIRECTLY, INDIRECTLY, ALONE, OR WITH
vhdl2verilog:          ANOTHER PARTY, (i) COPY, DISASSEMBLE, REVERSE ENGINEER, OR
vhdl2verilog:          DECOMPILE THE TOOL(S) (ii) MODIFY, CREATE DERIVATIVE TOOLS BASED
vhdl2verilog:          UPON, OR TRANSLATE THE TOOL(S) (iii) LICENSE, SELL, RENT, LEASE,
vhdl2verilog:          TRANSFER, GRANT ANY RIGHTS IN OR OTHERWISE COMMERCIALY
vhdl2verilog:          EXPLOIT THE TOOL(S) IN ANY FORM TO ANY OTHER PARTY, NOR SHALL
vhdl2verilog:
vhdl2verilog:          USER ATTEMPT TO DO ANY OF THE FOREGOING OR CAUSE OR PERMIT
vhdl2verilog:          ANY THIRD PARTY TO DO OR ATTEMPT TO DO ANY OF THE FOREGOING,
vhdl2verilog:          EXCEPT AS EXPRESSLY PERMITTED HEREUNDER. YOU ACKNOWLEDGE
vhdl2verilog:          AND AGREE THAT EDAUtils SHALL OWN ALL RIGHT, TITLE AND INTEREST
vhdl2verilog:          IN AND TO ALL INTELLECTUAL PROPERTY RIGHTS (INCLUDING
vhdl2verilog:          ALL DERIVATIVES OR IMPROVEMENTS THEREOF) IN THE TOOL(S) AND
vhdl2verilog:          ANY SUGGESTIONS, ENHANCEMENT REQUESTS, FEEDBACK,
vhdl2verilog:          RECOMMENDATIONS OR OTHER INFORMATION PROVIDED BY USERS
vhdl2verilog:          RELATING TO THE TOOL(S).
vhdl2verilog:
vhdl2verilog: Note: 51504: Running with options : -in examples/uart_trans/uart_trns.vhd -out examples/uart_trans/uart_trans.v -top uart_trans
vhdl2verilog: Warning: 53518: Physical path '/home/aluno/vhdl2verilog/10OCT2015/myEdaUtilsWork' specified in the library map file 'qmap.ini' does not exist
vhdl2verilog: Trying to create it in order to proceed further ...
vhdl2verilog: Note: 52504: Parsing file examples/uart_trans/uart_trns.vhd into logical library 'work' ...
vhdl2verilog: Note: 51604: Completed parsing file examples/uart_trans/uart_trns.vhd ...
vhdl2verilog: Note: 52506: Converting entity 'uart_trns' into verilog ...
vhdl2verilog: Note: 52507: Finished conversion of the entity 'uart_trns' into verilog ...
vhdl2verilog: Note: 52506: Converting architecture 'behv' of the entity 'uart_trns' ...
vhdl2verilog: Note: 52507: Finished conversion of the architecture 'behv' of the entity 'uart_trns' ...
vhdl2verilog: Note: 52510: Writing verilog file examples/uart_trans/uart_trans.v ...
vhdl2verilog: Note: 51921: Found 0 error(s) and 1 warning(s) ...
vhdl2verilog: Note: 51922: Thank you for using this utility
vhdl2verilog: Note: 51923: Send your suggestion/feedback to help@edautils.com
vhdl2verilog: Note: 51924: Exiting ...
aluno@aluno-VirtualBox:~/vhdl2verilog/10OCT2015$
```

Figura 7 – Tradução de Vhdl para Verilog (continuação da Figura 6).

**Usando o Qflow:** O “arquivo.v” utilizado foi obtido anteriormente (**uart\_trans.v**). Para a síntese dessa arquitetura precisa-se copiar o “arquivo.v” e colocar no diretório “source” (criado anteriormente na Figura 5). Na pasta raiz do projeto, que pela Figura 5 é a “qflow-projeto” escreva o seguinte comando:

**\$ qflow build <arquivo.v>**

Para o exemplo o <arquivo.v> é o **uart\_trans.v**, a Figura 8 demonstra os arquivos contidos no diretório raiz (“qflow-projeto”) e no diretório “source”, juntamente com o comando acima descrito.

```
aluno@aluno-VirtualBox: ~/qflow-projeto
aluno@aluno-VirtualBox:~/qflow-projeto$ ls
layout showResults.py source synthesis
aluno@aluno-VirtualBox:~/qflow-projeto$ cd source/
aluno@aluno-VirtualBox:~/qflow-projeto/source$ ls
uart_trans.v
aluno@aluno-VirtualBox:~/qflow-projeto/source$ cd ..
aluno@aluno-VirtualBox:~/qflow-projeto$ qflow build uart_trans
```

Figura 8 – Executando o Qflow.

A Figura 9 é demonstrada durante a execução do comando acima da Figura 8.

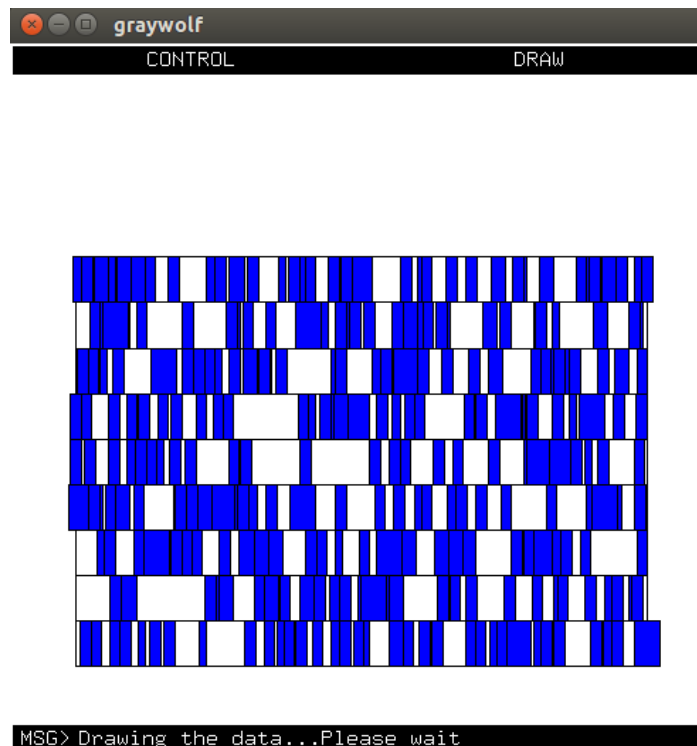


Figura 9 – Durante a execução do Qflow.

Após a execução da síntese da arquitetura e se nenhum erro apareceu durante essa síntese, precisa-se executar o comando abaixo (na pasta raiz do projeto), para saber a



frequência da arquitetura, a Figura 10 e Figura 11 (continuação da Figura 10) demonstra a utilização do comando.

**\$ qflow sta <arquivo.v>**

```
aluno@aluno-VirtualBox: ~/qflow-projeto

aluno@aluno-VirtualBox:~/qflow-projeto$ qflow sta uart_trans

-----
Qflow project setup
-----

Technology set to osu035 from existing qflow_vars.sh file
Regenerating files for existing project uart_trans

Running vesta static timing analysis

-----
Vesta static timing analysis tool
(c) 2013 Tim Edwards, Open Circuit Design
-----

Parsing library "osu035_stdcells"
End of library at line 6636
Parsing module "uart_trans"
Lib Read: Processed 6637 lines.
Verilog netlist read: Processed 373 lines.
Number of paths analyzed: 334

Top 20 maximum delay paths:
Path DFFPOSX1_13/CLK to DFFPOSX1_2/D delay 4109.96 ps
Path DFFPOSX1_14/CLK to DFFPOSX1_2/D delay 4061.64 ps
Path DFFPOSX1_15/CLK to DFFPOSX1_2/D delay 3988.3 ps
Path DFFPOSX1_13/CLK to DFFPOSX1_30/D delay 3789.16 ps
Path DFFPOSX1_13/CLK to DFFPOSX1_29/D delay 3743.81 ps
Path DFFPOSX1_14/CLK to DFFPOSX1_30/D delay 3740.84 ps
Path DFFPOSX1_13/CLK to DFFPOSX1_31/D delay 3710.3 ps
Path DFFPOSX1_13/CLK to DFFPOSX1_32/D delay 3709.62 ps
Path DFFPOSX1_14/CLK to DFFPOSX1_29/D delay 3695.49 ps
Path DFFPOSX1_16/CLK to DFFPOSX1_2/D delay 3675.88 ps
Path DFFPOSX1_15/CLK to DFFPOSX1_30/D delay 3667.5 ps
Path DFFPOSX1_14/CLK to DFFPOSX1_31/D delay 3661.98 ps
Path DFFPOSX1_14/CLK to DFFPOSX1_32/D delay 3661.3 ps
Path DFFPOSX1_13/CLK to DFFPOSX1_11/D delay 3654.99 ps
Path DFFPOSX1_12/CLK to DFFPOSX1_2/D delay 3646.39 ps
Path DFFPOSX1_13/CLK to DFFPOSX1_12/D delay 3644.07 ps
Path DFFPOSX1_15/CLK to DFFPOSX1_29/D delay 3622.15 ps
Path DFFPOSX1_14/CLK to DFFPOSX1_11/D delay 3606.67 ps
Path DFFPOSX1_14/CLK to DFFPOSX1_12/D delay 3595.75 ps
Path DFFPOSX1_15/CLK to DFFPOSX1_31/D delay 3588.64 ps
Computed maximum clock frequency (zero slack) = 243.312 MHz
-----

Number of paths analyzed: 334

Top 20 minimum delay paths:
Path DFFPOSX1_1/CLK to output pin ttre delay 236.846 ps
Path DFFPOSX1_2/CLK to output pin tbre delay 239.884 ps
Path DFFPOSX1_17/CLK to output pin tro delay 239.884 ps
Path DFFPOSX1_28/CLK to DFFPOSX1_28/D delay 347.343 ps
Path DFFPOSX1_13/CLK to DFFPOSX1_13/D delay 355.072 ps
Path DFFPOSX1_23/CLK to DFFPOSX1_22/D delay 362.574 ps
Path DFFPOSX1_11/CLK to DFFPOSX1_11/D delay 365.826 ps
Path DFFPOSX1_18/CLK to DFFPOSX1_17/D delay 366.021 ps
Path DFFPOSX1_19/CLK to DFFPOSX1_18/D delay 366.021 ps
Path DFFPOSX1_20/CLK to DFFPOSX1_19/D delay 366.021 ps
Path DFFPOSX1_21/CLK to DFFPOSX1_20/D delay 366.021 ps
Path DFFPOSX1_22/CLK to DFFPOSX1_21/D delay 366.021 ps
Path DFFPOSX1_2/CLK to DFFPOSX1_2/D delay 369.532 ps
Path DFFPOSX1_26/CLK to DFFPOSX1_26/D delay 391.369 ps
Path DFFPOSX1_27/CLK to DFFPOSX1_27/D delay 391.369 ps
```

Figura 10 – Frequência e tempo da Arquitetura.



```

Path DFFPOSX1_14/CLK to DFFPOSX1_14/D delay 391.387 ps
Path DFFPOSX1_25/CLK to DFFPOSX1_25/D delay 391.387 ps
Path DFFPOSX1_23/CLK to DFFPOSX1_23/D delay 391.387 ps
Path DFFPOSX1_24/CLK to DFFPOSX1_24/D delay 391.387 ps
Path DFFPOSX1_1/CLK to DFFPOSX1_1/D delay 394.75 ps
Design meets minimum hold timing.
-----

Number of paths analyzed: 65

Top 20 maximum delay paths:
Path input pin mr to DFFPOSX1_2/D delay 3342.78 ps
Path input pin mr to DFFPOSX1_30/D delay 3022.86 ps
Path input pin mr to DFFPOSX1_29/D delay 2976.85 ps
Path input pin mr to DFFPOSX1_31/D delay 2944.18 ps
Path input pin mr to DFFPOSX1_32/D delay 2943.49 ps
Path input pin mr to DFFPOSX1_11/D delay 2888.4 ps
Path input pin mr to DFFPOSX1_12/D delay 2878.99 ps
Path input pin mr to DFFPOSX1_1/D delay 2737.54 ps
Path input pin mr to DFFPOSX1_17/D delay 1773.38 ps
Path input pin mr to DFFPOSX1_18/D delay 1773.38 ps
Path input pin mr to DFFPOSX1_19/D delay 1773.38 ps
Path input pin mr to DFFPOSX1_20/D delay 1773.38 ps
Path input pin mr to DFFPOSX1_21/D delay 1773.38 ps
Path input pin mr to DFFPOSX1_22/D delay 1773.38 ps
Path input pin mr to DFFPOSX1_25/D delay 1605.17 ps
Path input pin mr to DFFPOSX1_26/D delay 1605.17 ps
Path input pin mr to DFFPOSX1_27/D delay 1605.17 ps
Path input pin mr to DFFPOSX1_23/D delay 1588.91 ps
Path input pin mr to DFFPOSX1_24/D delay 1588.91 ps
Path input pin mr to DFFPOSX1_28/D delay 1231.8 ps
-----

Number of paths analyzed: 65

Top 20 minimum delay paths:
Path input pin mr to DFFPOSX1_13/D delay 64.7154 ps
Path input pin mr to DFFPOSX1_12/D delay 64.7154 ps
Path input pin sfd to DFFPOSX1_2/D delay 167.855 ps
Path input pin mr to DFFPOSX1_16/D delay 225.198 ps
Path input pin mr to DFFPOSX1_11/D delay 253.72 ps
Path input pin mr to DFFPOSX1_14/D delay 378.65 ps
Path input pin mr to DFFPOSX1_15/D delay 384.783 ps
Path input pin mr to DFFPOSX1_10/D delay 414.138 ps
Path input pin mr to DFFPOSX1_9/D delay 414.138 ps
Path input pin mr to DFFPOSX1_7/D delay 414.138 ps
Path input pin mr to DFFPOSX1_6/D delay 414.138 ps
Path input pin mr to DFFPOSX1_5/D delay 414.138 ps
Path input pin mr to DFFPOSX1_4/D delay 414.138 ps
Path input pin trc to DFFPOSX1_28/CLK delay 414.585 ps
Path input pin trc to DFFPOSX1_24/CLK delay 414.585 ps
Path input pin trc to DFFPOSX1_23/CLK delay 414.585 ps
Path input pin trc to DFFPOSX1_22/CLK delay 414.585 ps
Path input pin trc to DFFPOSX1_21/CLK delay 414.585 ps
Path input pin trc to DFFPOSX1_20/CLK delay 414.585 ps
Path input pin trc to DFFPOSX1_19/CLK delay 414.585 ps
-----

aluno@aluno-VirtualBox:~/qflow-projeto$

```

Figura 11 – Frequência e tempo da Arquitetura (continuação da Figura 10).

Para a extração dos resultados da área e frequência da arquitetura utiliza-se um script em python como citado anteriormente. (showResults.py). Abaixo está o comando necessário. A Figura 12 demonstra o comando e a saída esperada para esse exemplo, com o número de gates e a frequência máxima de operação.

**./showResults.py qflow\_vars.sh**

```
aluno@aluno-VirtualBox: ~/qflow-projeto
aluno@aluno-VirtualBox:~/qflow-projeto$ ls
layout  project_vars.sh  qflow_exec.sh  qflow_vars.sh  showResults.py  source  synthesis  synth.log
aluno@aluno-VirtualBox:~/qflow-projeto$ ./showResults.py qflow_vars.sh
Diretorio do synth.log: /home/aluno/qflow-projeto/synth.log
Biblioteca de celulas: /usr/local/share/qflow/tech/osu035/osu035_stdcells.lib

Portas utilizadas no projeto:
    OR2X2: 10
    AND2X2: 31
    XOR2X1: 3
    NOR2X1: 51
    XNOR2X1: 24
    INVX1: 17
    NAND2X1: 156

NUMERO DE GATES EQUIVALENTES UTILIZADOS: 336.0
FREQUENCIA MAXIMA DE OPERACAO: 243.312 MHZ
aluno@aluno-VirtualBox:~/qflow-projeto$
```

Figura 12 – Utilização do script em python para a obtenção dos resultados.

## 7. Links para downloads

Arquivos fontes, instaladores, scripts gerados pelo grupo:

- <https://github.com/radc/qflow>

Open Circuits Design:

- <http://opencircuitdesign.com/>

EDA Utils

- <http://www.edautils.com/>
- <http://www.edautils.com/vhdl2verilog.html>