Introduction to Xilinx Vivado tools

This document is meant to be a starting point for users who are new to using the Xilinx Vivado tools. The document will describe the basic steps to start, create, simulate, synthesize, implement and program an FPGA using Vivado through a series of screenshots and an example design which is a simple binary counter.

Starting the tool:

Use the windows application browser to start Vivado 2014.2. If a different version of the tool is installed on your computer, start the corresponding tool. The steps described in this document are very similar to other versions of the tool.

Once the tool is started, you will see a page as shown in Fig. 1. Spend some time looking at the quick take videos which talk about tool features on a wide variety of subjects.

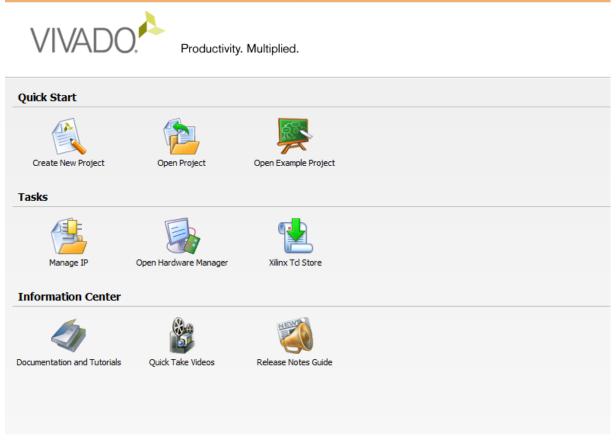


Figure 1: Vivado start page

Click on Create New Project to start a new project. You will be prompted to specify a directory and a project name. Use a meaningful name so that it is easier to open and remember what the project was about at a later point of time.

2	New Project	X
Project Nam Enter a nan	e ne for your project and specify a directory where the project data files will be stored.	
Project name:	MY_COUNTER	0
Project location:	D://LSI_2015/FPGA_INTRO_LEC/DEMO_PJT	
🖌 Create proj	ect subdirectory	
Project will be cr	reated at: D:/VLSI_2015/FPGA_INTRO_LEC/DEMO_PJT/MY_COUNTER	
		_
	< <u>B</u> ack <u>N</u> ext > Einish	Cancel

Figure 2: Choose project name and directory

Navigate through the project creation assistant by clicking on Next. Add sources, IPs and constraints if you have any. You can always add new sources, IPs and constraints later into the project.

k			New Pro	ject					x
Default Part Choose a default Xili	nx part or board for	your project. Th	iis can be chang	ed later.					
Select: 🔗 Parts 📓	Boards								
Product category:	All		▼ <u>P</u> ad	kage:	All			-	
Eamily:	All		▼ Spe	e <u>d</u> grade:	All			+	
Sub-Family:	All		▼ <u>T</u> en	np grade:	All			-	
			Si R	evision:	All			-	
			Reset All	Filters					
Search: Q- xc7a100t	cs	(4 matc	hes)			-14	14		
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlop	OS	Block RAMs	DSPs	Gb Transceivers	GTX Trar
xc7a100tcsg324-3	324	210	63400	126800		135	240	0	0
xc7a100tcsg324-2	324	210	63400	126800		135	240	0	0
xc7a100tcsg324-2L	324	210	63400	126800		135	240	0	0
📎 xc7a100tcsg324-1	324	210	63400	126800		135	240	0	0
<	Ш								> 🗆
					< <u>B</u> ack	. <u>N</u> ex	t > Ein	ish Cano	:el

Figure 3:Choose the target FPGA

Choose the target FPGA. We will use the Xilinx Artix-7 FPGA with a speed grade of 1.

If you are successful in following the above steps, you will end up with a screen which looks like the one shown in Fig. 4.

2		N	MY_COUNTER - [D;/VLSI_2015/FPGA_INTRO_LEC/DEMO_PJT/MY_COUNTER/MY_COUNTER.xpr] - Vivado 2015.1	- 0 ×
File Edit Flow Tools Window			Q- Search comma	nds
🏄 🚵 📾 🖉 🎼 🐘 🗙 🔈	🔸 🕨 📸 🚳 🎉 ∑ 🧑 🔚 Default Layout 🔹	医金子		Rea
Flow Navigator 《	Project Manager - MY_COUNTER			
🔍 🎞 曲		- 0 & ×	Project Summary X	027
Project Hanager Project Settings Add Sources Language Templates Profest Settings	Q Image: An and a state of the state of		Image: Settings Project settings Project settings Dylocation Project settings Dylocation Project settings Dylocation	Edit R
 P Integrator Create Block Design Create Block Design Generate Block Design Smulation Smulation Settings Run Smulation 			Ten modele name: Inspectation Status: Not detected Status: Not detected Status: Not detected Messages: No errors or warrings Part: sch 2001csg124-1	*
RTL Analysis G Elaboration Settings D GP Elaborated Design	Hierarchy Libraries Comple Order		DRC Volutions a Tening DRC Volutions to see the control of the see the see the control of t	\$
 Synthesis 	Sources 🖓 Templates		Utilization 2 Power	\$
Synthesis Settings Settings Run Synthesis	Properties	- 0 & ×	Run Sinthesig to see utilization results Bun Independent to see power results	
Den Synthesized Design Den Synthesized Design Den Synthesized Design Den Inglementation Den Synthesized Design Den Inglementation Design and Debug Statics Statics Den Statics Den Statissem Den Synthesized Design	Select an object to see properties			
	Design Runs			- 0 2 ×
	Q Name Constraints St ⊞ → synth_1 constrs_1 No	itatus Progre ot started ot started	grees Wr6 Tr6 Wr6 Tr6 Wr6 Tr6 TPVS Faked Routes UUT N. UUTs PF N. PFs BUAH N. BUAH's DDP N. OD's Start Blanced Database Database Defaults (Invado Synthesis	2015)

Figure 4: Project home page

The left hand pane is called the Flow navigator. It enables you to go through the FPGA flow from simulation, synthesis and implementation and finally to generate the bitstream and program the FPGA. More details are shown in Fig.5.

Project settings such as synthesis, implementation and bit generation options can be modified by clocking on the Project settings tab.

We will not use the IP integrator in this course, but this is used to package your design into your own custom IP which can be used by others.

Simulations can be launched by clicking on Run Simulation tab. Remember to set the correct top level for running simulations. Often it is a testbench which is created by you are provided by the course instructor to verify your design. This can be done by right clicking on the module name in the "Sources" window and choosing the right module. Usually, the top level for simulation is different from the top level file for synthesis and implementation as the testbench contains non synthesizable constructs.

Synthesis can be launched by clicking on Run Synthesis. Synthesis nor simulation will work if there are syntax errors, and Vivado will show errors in the "Sources" window as well as the "Messages" window. A design which has been synthesized can be examined further by clicking on the "Open Synthesized Design" tab. This will enable you to examine the schematics, add additional timing constraints and other tasks.

A synthesized design can be implemented for the target FPGA by clicking on the "Run Implementation" tab. You can examine the design placement on the FPGA, and timing results once implementation is completed by opening the implemented design.

Finally a bitstream can be generated and the "Open Hardware Manager" tab can be used to connect to the FPGA and program the generated bitstream for execution on the FPGA.

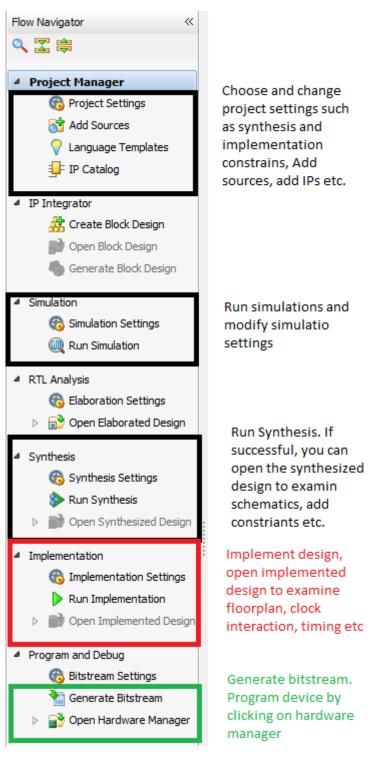


Figure 5: Project Flow navigator

The Counter example that we will use for this document uses an internal clock generator for running the counter. The code for the counter module is shown below. It contains a clk_wiz_0 component which is the clock generator we will use. There is one input clock, connected to the top level port along with two output clocks, one of which will be used by our design.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;
entity my_counter is
  Port ( start_count : in STD_LOGIC;
       rst : in STD_LOGIC;
       clk_in1 : in STD_LOGIC;
       count_val : out unsigned (2 downto 0));
end my_counter;
architecture Behavioral of my_counter is
component clk_wiz_0
port
 (-- Clock in ports
              : in std_logic;
 clk_in1
 -- Clock out ports
 clk out1
               : out std_logic;
 clk_out2
               : out std_logic;
 -- Status and control signals
 reset
             : in std_logic;
 locked
              : out std_logic
);
end component;
signal clk_out1 : std_logic;
signal clk_out2 : std_logic;
signal locked : std_logic;
signal local_count : unsigned (19 downto 0);
signal local_probe : std_logic_vector(7 downto 0);
signal local_count_val: std_logic_vector(3 downto 0);
begin
my_clock_gen: clk_wiz_0
 port map (
 -- Clock in ports
 clk_in1 => clk_in1,
 -- Clock out ports
 clk_out1 => clk_out1,
 clk_out2 => clk_out2,
 -- Status and control signals
 reset = '0',
 locked => locked
);
process(clk_out1, rst)
begin
 if(rst = '0') then
  local\_count <= (others => '0');
 elsif(rising_edge(clk_out1)) then
  if(start_count = '1') then
     local\_count <= ((local\_count) + 1);
  else
     local_count <= local_count;
 end if;
 end if;
 end process;
```

count_val <= local_count(2 downto 0) + local_count(19 downto 17); local_count_val <= std_logic_vector(local_count(3 downto 0));</pre>

end Behavioral;

The following snippet contains the code for the counter testbench.

library IEEE; use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using -- arithmetic functions with Signed or Unsigned values use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating -- any Xilinx leaf cells in this code. --library UNISIM; --use UNISIM.VComponents.all;

entity counter_testbench is -- Port (); end counter_testbench;

architecture Behavioral of counter_testbench is

component my_counter Port (start_count : in STD_LOGIC; rst : in STD_LOGIC; clk_in1 : in STD_LOGIC; count_val : out unsigned (2 downto 0)); end component;

```
signal start_count_tb : std_logic;
signal rst_tb : std_logic;
signal clk_in_tb : std_logic := '0';
signal count_val_tb : unsigned(2 downto 0);
constant clk_period : time := 2.5ns;
constant delay_val : time := 10ns;
```

begin

```
my_counter_inst : my_counter
port map(
    start_count => start_count_tb,
    rst => rst_tb,
    clk_in1 => clk_in_tb,
    count_val => count_val_tb);
```

clk_in_tb <= not clk_in_tb after clk_period;

rst_tb <= '0' after delay_val,'1' after 1500*delay_val; start_count_tb <= '0' after delay_val, '1' after 2000*delay_val;

end Behavioral;

The constraints file looks like this

set_property PACKAGE_PIN E3 [get_ports clk_in1] set_property IOSTANDARD LVCMOS33 [get_ports clk_in1] #create_clock -add -name clk_in1 -period 10.00 -waveform {0 5} [get_ports clk_board]

set_property PACKAGE_PIN C12 [get_ports {rst}] set_property IOSTANDARD LVCMOS33 [get_ports {rst}]

set_property PACKAGE_PIN U9 [get_ports {start_count}] set_property IOSTANDARD LVCMOS33 [get_ports {start_count}] # LED 0 set_property PACKAGE_PIN T8 [get_ports {count_val[0]}] set_property IOSTANDARD LVCMOS33 [get_ports {count_val[0]}] # LED 1 set_property PACKAGE_PIN V9 [get_ports {count_val[1]}] set_property IOSTANDARD LVCMOS33 [get_ports {count_val[1]}] # LED 2 set_property PACKAGE_PIN R8 [get_ports {count_val[2]}] set_property IOSTANDARD LVCMOS33 [get_ports {count_val[2]}]

set_property CFGBVS Vcco [current_design] set_property config_voltage 3.3 [current_design] Since we have a clock generator as a component we will need to instantiate/create this for our design. We can do this by clicking on "IP Catalog" in the the "Flow Navigator". Search for an IP starting with "Clocking Wizard" in the Catalog and double click on it. You should see a window which lets you customize the IP. Choose two output clocks and leave the rest of the setting unchanged. Click "OK" to generate the Clock generator IP. Fig. 6 shows a snapshot of this process.

À File Edit Flow <u>T</u> ools <u>Wi</u> ndow I		MY_COUNTER - [D:/VLSI_2015/FPGA_INTRO_LI	EC/DEMO_PJT/MY_COUNTER	R/MY_COUNTER.xpr	r] - Vivado 2015.	1						X
	Import july 100 100 100 100 100 100 100 100 100 10	•								x,• bear or comman		Ready
Flow Navigator «	Project Manager - MY_COUNTER											
۹ 🗶 🗰	Sources _ C L ×	∑ Project Summary × 9 IP Catalog ×										
	으, 🏗 🚔 📾 📾 👔 🛃	*] Search: Q dock	7 matches)	Customize IP Very Colspan="2" Customize IP Very Colspan="2" Customize IP Very Colspan="2" Customize IP Very Colspan="2" Very Colspan="2" Customize IP Very Colspan="2" Customize IP								

Figure 6: Generating a Clocking IP

Once the generation is complete, you will no longer see a ? mark under the "my_clock_gen" instance name. You are not ready to launch the simulation. Click on "Run Simulation" in the "Flow Navigator". Make sure that "counter_testbench" is the top level in your simulation sources.

Examine the "log" window in at the bottom to see if there are any errors or warnings. If simulation is successful, the simulator is launched and you can see some signals in the waveform viewer. You can choose additional signals to debug by clicking on the corresponding module instance name, then the signal and right-click-> add to waveform window.

Run the simulation for 22us by typing *run 22us* in the "Tcl Console" at the bottom. You can zoom in zoom out using the tools in the waveform viewer.

Fig. 7. Shows a snapshot of the simulation

Once you are satisfied with the simulation behaviour, you can "Run Synthesis". Make sure you keep an eye on the "log" window to see the messages that appear during synthesis. You can always view Reports by using the Reports tab at the bottom as shown in Fig. 8.

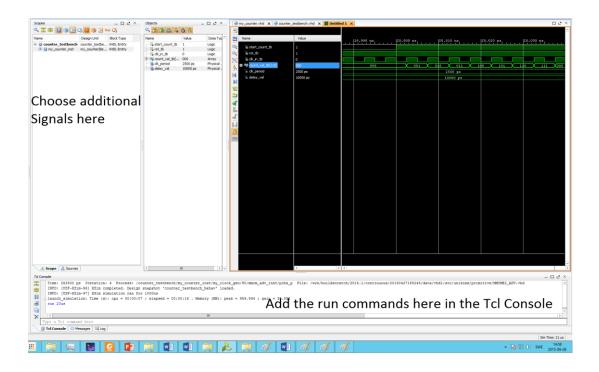


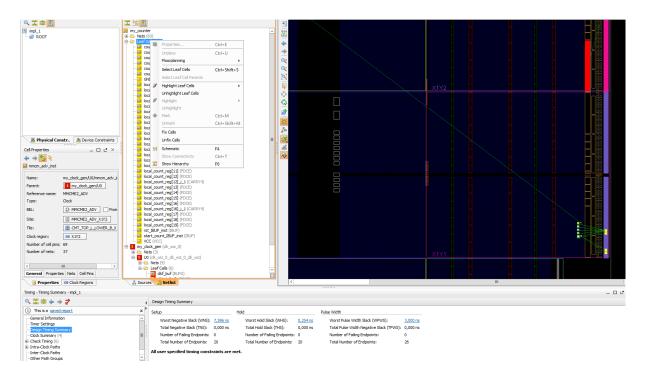
Figure 7: Simulation window

orts					
Name		Modified	Size		GUI Report
⊜ synth_1					
···· 📄 Vivado Synthesis Re	port	9/6/15 3:02 PM		16.4 KB	
📄 Utilization Report		9/6/15 3:02 PM		6.8 KB	
⊡ • Out-of-Context Module Ru	ns				
Implementation					
Opt Design (opt_design)					
Post opt_design DR					
- Place Design (place_design					
···· Vivado Implementati	-				
Pre-Placement Incre	mental R				
···· 📄 IO Report ···· 📄 Utilization Report					
Control Sets Report					
Incremental Reuse F					
-Route Design (route_designed)					
···· 📄 Vivado Implementati					
···· WebTalk Report					
DRC Report					
···· Power Report					
📄 Route Status Report	t				
			_		
🛄 Tcl Console 🔶 💭 Messages 🔶	Log	Reports Design	Runs		
	T		7		
View	View	Examine all	•		
	Run logs	reports here		Examine	design run
Infors;	num logs		:	status by	clicking he
Warning;					_
erors at					
different					
diff diff diff diff diff diff diff diff					
stages					

Figure 8: Logs and Messages Viewer

Once synthesis is complete, you can "Open Synthesized Design" and examine the various menus such as creating additional constraints, viewing the schematic etc.

"Run Implementation" when you are ready and wait for implementation to finish. It is very important to make sure that the design constraints are met when implementation ends. Keep an eye on the "log" window and look at the "Messages" to understand any Warning and Errors that you might encounter. Once Implementation is complete, you can click on "Open Implemented Design" to examine the actual mapping of your design onto FPGA resources, review timing reports and also to examine clock interaction if your design has multiple clocks. The implemented design for the counter is shown in Fig. 9.





You can zoom in-out from different parts of your design as well as highlight different modules of your design by right clicking in the netlist view. Examine timing to make sure that both setup and hold slack are zero or positive. In Fig. 9 the clock generator is highlighted in red and the counter in yellow along with the pads which are used for output mapping. Note that the router places the design close to the pads to make routing easier. It is also possible to cross-probe into RTL code to examine which resource was mapped to which line of your RTL code by choosing the resource in the netlist view, right clicking and choosing "Go to source".

Another important window in Vivado is the Project summary window. Here you can get an overview of all the different stages of your design flow and examine resource utilization as well as get a power consumption estimate for your design. Fig 10. Shows a snapshot from the project summary window for the counter design. Make sure that you have no critical warning/DRC errors before you generate your bit stream. Sometimes, bit stream generation will fail if you have not made sure that the design constraints and pins are mapped correctly.

Σ	Project Summary X @ my_counter.vhd X @ counter_testbench.vhd X		2	×
Z	Project Settings	Edit	*	^
	Project name: MY_COUNTER Project coation: Dr/MSI_2015/FPGA_INTRO_LEC/DEMO_PJT/MY_COUNTER Product family: Artix-7 Project part: xc721001ccs324-1 Top module name: my_counter			
	Synthesis *	Implementation	*	
	Status: ✓ Complete Messages: ① <u>Struarnings</u> Active run: <u>synth_1</u> Part: xc7a100tsg324-1 Strategy: <u>Vivado Synthesis Defaults</u>	Status: ✓ Complete Messages: No errors or warnings Active run: imol_1 Part: xx7a100tsg324-1 Strategy: Wrado Implementation Defaults Incremental comple: None Summary: Route Status		
	DRC Violations	Timing	*	=
	No DRC violations were found.	Worst Negative Slack (WNS): 7.596 ns Total Negative Slack (TNS): 0 ns Number of Falling Endpoints: 0 Total Number of Endpoints: 20 Implemented Timing Report 5		
	Utilization - Post-Implementation	Power	*	
	FF 196 LUT 196 1/0 396 BUFG 696 MMCM 0 25 50 75 100 Utilization (%)	Total On-Chip Power: 0.211 W Junction Temperature: 26,0 °C Thermal Margin: 59,0 °C (12,8 W) Effective 03A: 4,6 °C/W Power supplied to off-chip devices: 0 W Confidence level: Medium		

Figure 10: Project summary

Click on "Generate Bitstream" when ready. If bitstream generation is successful a file with the .bit extension will be created in "YOUR_DESIGN_PATH\PROJECT_NAME\PROJECT_NAME.runs\impl_1" folder where PROJECT_NAME is the name of your project.

On successful generation of the .bit file, click on "Open Hardware Manager" to launch the hardware manager to program the FPGA. Make sure that the FPGA is connected to your computer through a USB cable and that the FPGA is powered on.

When you click on the "Open hardware manger" you will get a prompt saying "no hardware is connected". Click on "Open Target" followed by "Open a New Target". Choose "Local Server" as the FPGA will be connected locally to your machine and click "Next" as shown in Fig.11.

If the FPGA is recognized, a new window showing the name of the FPGA will pop us as shown in Fig.12. Click "Next" and the hardware manager will open with the FPGA connected to the computer.

Right Click on the FPGA, and choose bit file as shown in Fig. 13. You can now program the FPGA and verify that your design works on the FPGA.

Hardware Manager - unconnected	
No hardware target is open. <u>Open target</u>	
Hardware _ 🗆 🖻 🗶	M my_counter.vhd x M counter_testbench.vhd x
< 五 ≑ 圖 ◎ ▶ ▶ ■	D://LSI_2015/FPGA_INTRO_LEC/DEMO_PJT/MY_COUNTER/MY_COUNTER.srcs/sources_1/imports/new/my_counter.vhd
	<pre>65 signal locked : std_logic; 66 signal local_count : unsigned (19 downto 0); 67 signal local_probe : std_logic_vector(7 downto 0); 68 signal local_count_val : std_logic_vector(3 downto 0); 69 70 begin 71 72 72 73 my_clock gen; clk wiz 0</pre>
No content	// 74 p A Open New Hardware Target X
no content	75 Hardware Server Settings 76 cl 77 cl 77 cl 78 cl 80 81 re 82 10 83);
Properties □ Ľ ×	84 85 86 87 88 89 90 loca
Select an object to see properties	91 92 proc 93 beg1 94 1f 95 96 e1 97 98 99 100 101 en 102 en 103 end 104

Figure 11: Connect to a hardware target

à			Open New Hardware Target
	dware target	from the list o	f available targets, then set the appropriate JTAG clock (TCK) ted devices, decrease the frequency or select a different target.
Hardware Target	ts		
Type N	ame	יע	AG Clock Frequency
	g	4592245A 150	
Hardware Device	e (for unkno	wn devices sn	pecify the Instruction Register (IR) length)
Name	ID Code	IR Length	
xc7a100t_0	13631093	6	
Hardware server	: localhost:3	121	
			< Back Next > Einish Cancel

Figure 12: FPGA connected

🚴	
<u>File Edit Flow Tools Window Layout View Help</u>	
🯄 🚵 🕼 🕼 🗎 🐂 🗙 🐝 🚳 🛄 Default Layout	💌 💓 🔖 🔖 💿 Dashboard 🕶 🗐 🖏
Hardware Manager - localhost/xilinx_tcf/Digilent/210274592245A	
(i) There are no debug cores. Program device Refresh device	
Hardware _ 🗆 🖻 😕	
State = Sta	
Name Status	
🖃 📔 localhost (1) Connected	
⊡ 🔮 xilinx_tcf/Digilent/210274592245A (1) Open	
è. P	rogram Device
Select a bitstream programming file and download probes file that corresponds to the debug cores co	it to your hardware device. You can optionally select a debug phrained in the bitstream programming file.
Bitstream file: FPGA_INTRO_LEC/DEMO_PJ	T/MY_COUNTER/MY_COUNTER.runs/impl_1/my_counter.bit 💿 🗔
Debug probes file:	
✓ Enable end of startup check	
	Program Cancel
Usadoren Davier Deservice 🗖 🖓 🗸	

Figure 13: Choose and program bit file

A working example project can be found under

🖸 🔜 🗢 EITF35			- □ >
ile Home Share View			~
· → · ↑ 📙 › This PC › shared (\\fs-n.ı	net.lth.se) (S:) > Courses > eit > EITF35	✓ 乙 Search	EITF35
Desktop	^ Name ^	Date modified Ty	rpe Size
Documents	COUNTER_EXAMPLE_PJT	2016-09-01 10:50 Fi	le folder
🕂 Downloads	USER_MANUALS	2016-09-01 10:52 Fi	le folder
👌 Music	VGA_REF_DESIGN	2016-09-01 10:51 Fi	le folder
Pictures			
Videos			
🟪 Windows (C:)			
' 🛖 shared (\\fs-n.net.lth.se) (S:)			
🗸 📙 Courses			
> 📙 arkitektur			
> 📙 bme			
> 📙 byggmiljo			
> 🔤 construction			
control			
cs			
> 📙 design			
✓ 📙 eit			
> 📙 EDI610			
> 📙 EIT020			
> 📙 EITF11			
✓ 📙 EITF35			
> 📙 COUNTER_EXAMPLE_PJT			
> 📙 USER_MANUALS			
> VGA_REF_DESIGN			
> EITF40			
> 📙 EITN50			
ETEN05			
ETT051			
ETTN01			
energy			
- food			
> 📙 fysik			
🔄 iml			
🔄 immun			
> 📙 kemiteknik			
> 🔤 kilu			
> <mark>M</mark>			
maths	v <		
items	¥ (