Picorv32 Cache

CS391 Final Project

Beren, Ruby, Michael, Isaac

TABLE OF CONTENTS

ONE COLUMN

Do you know what helps you make your point clear? Lists like this one:

- They’re simple
- You can organize your ideas clearly
- You’ll never forget to buy milk!

You can replace the image. Just right-click on it and select “Replace image”

01

INTRODUCTION

Goals - Implement a working Cache

Cache protocol

To put it simply, we want to implement a simple 1-set direct-mapped cache to link between the Picorv32 CPU and the BRAM.

Connecting Pico to Cache to BRAM

In order for the cache to properly sit between the Pico and BRAM we have to redo all the wiring.

Make sure the output from CPU are the inputs to cache and the output of the cache will connect to the mem_axi wires to BRAM

- Cache would act as the subordinate to the Pico, and as the manager to the BRAM

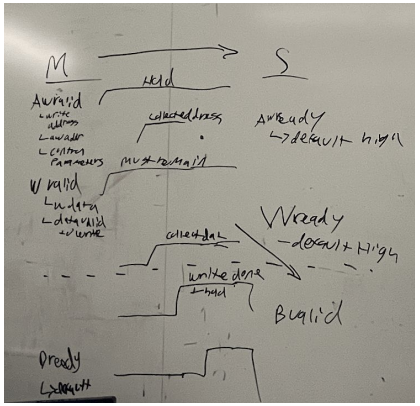
02

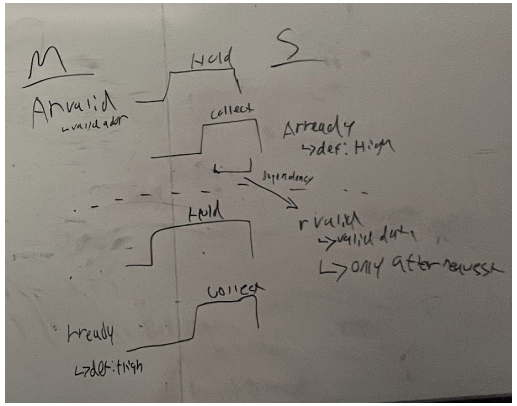
APPROACH

Approach - The Interface Logic (Axi Protocol)

- Generally, our strategy for connecting our Cache module with the Bram and the Pico, was with the Axi Protocol, as that was the existing method with which the Pico and the Bram communicated with each other.
- Given this strategy, that would place the Cache as both the Manager, and the Subordinate, depending on which module it was in communication with. This meant that we had to develop a very clear understanding of all parts of the Axi Protocol for both reads and writes.
    - On any request: CPU starts an AXI request with the cache
    - On a miss: Cache stalls CPU AXI request and starts an AXI transaction with the BRAM
    - On Hit/miss is resolved: Finish CPU AXI request and perform read/write

Approach - Axi Interaction (Planning)





- We began by drawing up a visual of all Axi interactions, both reads and writes
- This allowed us to conceptualize all the Axi signals we would need to manage and understand how our logic should be constructed within the cache to properly handle the Pico and the Bram

Approach - Axi Interaction (Interfacing)

```
//CPU AXI
input wire cpu_awvalid,
output reg cpu_awready, //We changed this (previously awread)
input wire [31:0] cpu_awaddr,
input wire [2:0] cpu_awprot, //ignore for now

input wire cpu_wvalid,
output reg cpu_wready,
input wire [31:0]  cpu_wdata,
input wire [3:0]  cpu_wstrb, //ignore for now

output reg cpu_bvalid,
input wire cpu_bready,

input wire cpu_arvalid,
output reg cpu_arready,
input wire [31:0] cpu_araddr,
input wire [2:0] cpu_arprot,// ignore for now

output reg cpu_rvalid,
input wire cpu_rready,
output reg [31:0] cpu_rdata,
```

```
//BRAM AXI

output reg bram_awvalid,
input wire bram_awready,
output reg[31:0] bram_awaddr,
output wire[2:0] bram_awprot, //hardcoded for now

output reg bram_wvalid,
input wire bram_wready,
output reg[31:0] bram_wdata,
output wire[3:0] bram_wstrb, //hardcoded for now

input wire bram_bvalid,
output reg bram_bready,

output reg bram_arvalid,
input wire bram_arready,
output reg [31:0] bram_araddr,
output wire [2:0] bram_arprot,// hardcoded for now

input wire bram_rvalid,
output reg bram_rready,
input wire [31:0] bram_rdata
```

- This lead us to be able to develop our input/output arrangement for both the modules, ensuring that the proper Manager/Subordinate arrangement was executed

Approach - Transitioning into Cache Logic Design

- With the interactions fully understood, having gone very in depth on the rules of the Axi Protocol, we were now able to bridge the Subordinate and Manager interfaces of our cache.
- We moved into developing the internal state logic that would allow the management of our cache, fully informed/handled by Axi inputs from either side.

Approach - Caching Overview

- We Implemented our cache with the usage of several states (Finite State Machine)
- The states were implemented based on the different types of Read and Write cases
- E.g. Read-hit, Read-miss, Write-Hit, Write-Miss, etc.
- Parameterized index and tag length
  - implemented with 4 arrays of register, each with 2^{index_bits} lines
  - valid, dirty, data, tag arrays
- Block size of 1 word (32 bits)

Approach - The Caching Logic

- **IDLE** → Resting state for the cache, awaits signals for reads and writes.
- **READ_MEM** → Reads the data from the cache to the cpu, returns to IDLE on completion.
- **OVERWRITE** → If there is a write hit, alters the data in the cache accordingly and sets the dirty bit to high.
- **EVICT** → On a read miss with dirty bit or a write miss with dirty bit, perform first part of write Axi handshake with BRAM.
- **EVICT_ACK** → After EVICT, handle second part of write Axi handshake, and then travel to either REFILL or OVERWRITE depending on Read or Write instruction respectively.
- **REFILL** → Handles the first part of the read handshake, requesting data from a specific address in the BRAM.
- **REFILL_ACK** → Places the data from the BRAM into the cache to be read in the READ_MEM stat.

03

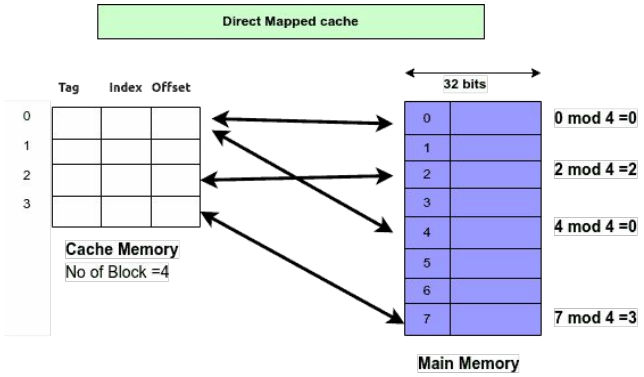
TESTING

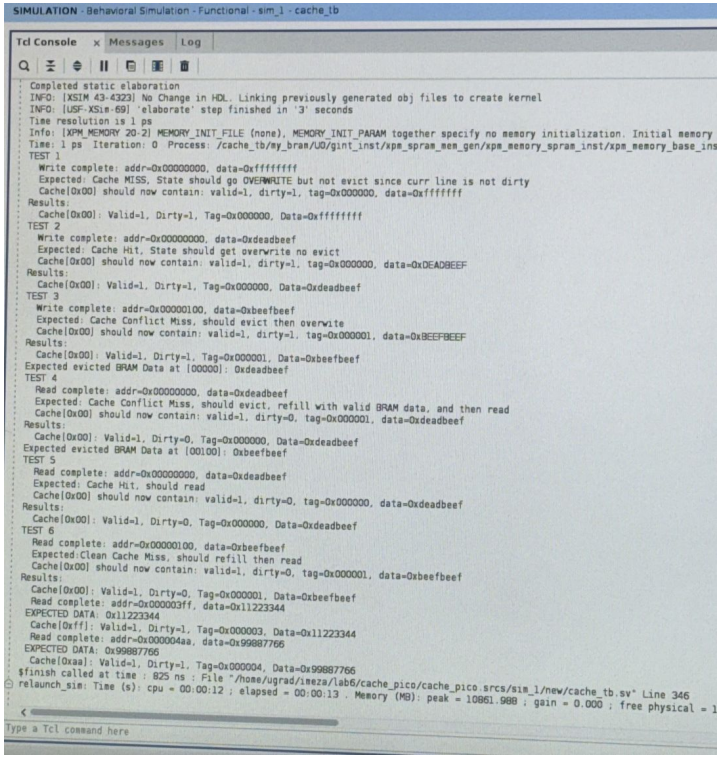
Testing Approach

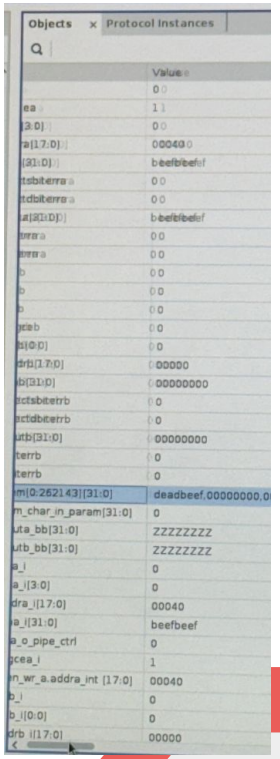
- Unit Tests
- How Cache handles Reads & Writes
- Ensures State Functionality

Type	awaddr/araddr	Rdata/Wdata	V	D	In-Cache Data	BRAM memory	Results
Write	0x000000_00	0xffffffff	1	1	0xffffffff		Write Miss (non-dirty) - Overwrite Compulsory
Write	0x000000_00	0xdeadbeef	1	1	0xdeadbeef		write hit -> overwrite
Write	0x000001_00	0xbeefbeef	1	1	0xbeefbeef	0xdeadbeef at 0x00000000	Write Miss (Dirty) (evict -> overwrite) Conflict
Read	0x000000_00	0xdeadbeef	1	0	0xdeadbeef	0xdeadbeef at 0x00000000 0xbeefbeef at 0x00000100	Read Miss (dirty) - evict -> refill -> read Conflict
Read	0x000001_00	0xbeefbeef	1	1	0xbeefbeef	0xdeadbeef at 0x00000000 0xbeefbeef at 0x00000100	Read Hit -> read
Read	0x000000_00	0xdeadbeef	1	0	0xdeadbeef	0xdeadbeef at 0x00000000 0xbeefbeef at 0x00000100	Read Miss (non-dirty) - refill -> read

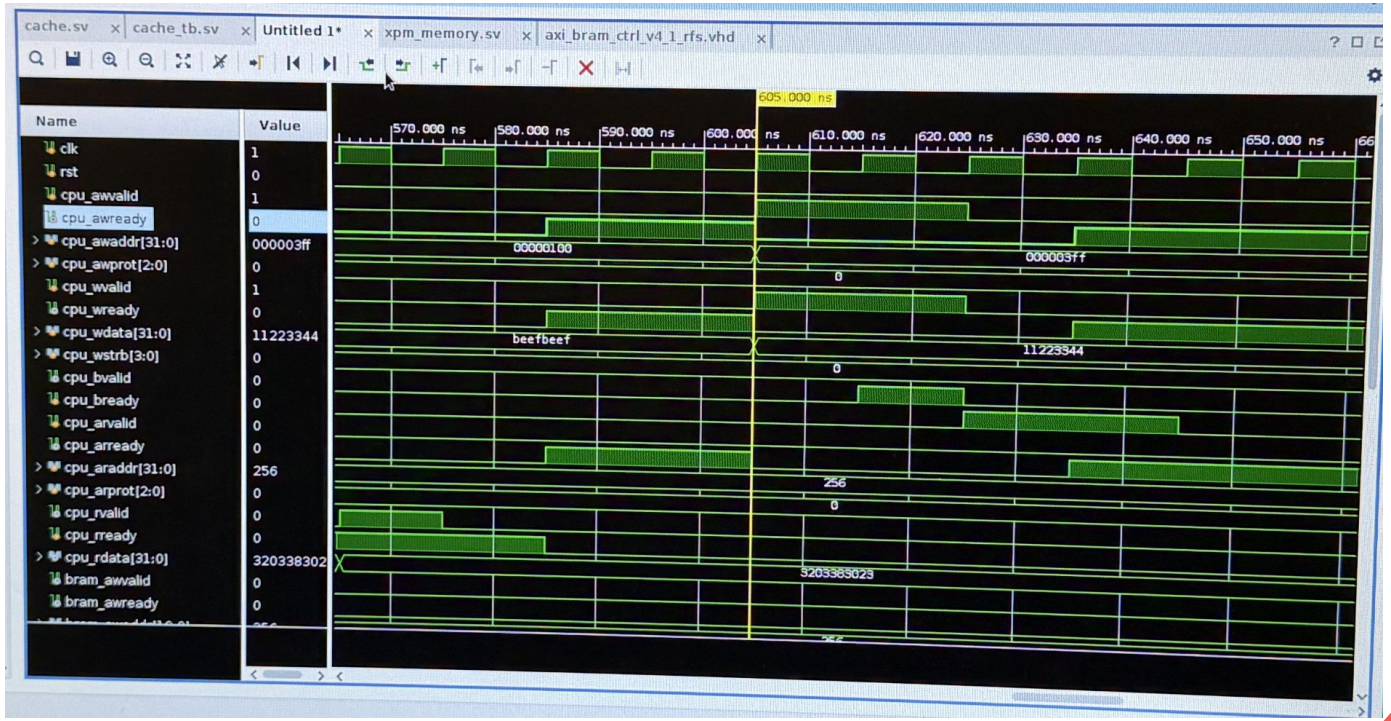
Outcomes


Direct Mapped cache
Tag
Index
Offset
32 bits
0
1
2
3
0
1
2
3
4
5
6
7
0 mod 4 =0
2 mod 4 =2
4 mod 4 =0
7 mod 4 =3
Cache Memory
No of Block =4
Main Memory


SIMULATION - Behavioral Simulation - Functional - sim_1 - cache_tb
Tcl Console
Messages
Log
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
Time resolution is 1 ps
Info: [XPM_MEMORY 20-2] MEMORY_INIT_FILE (none), MEMORY_INIT_PARAM together specify no memory initialization. Initial memory
Time: 1 ps Iteration: 0 Process: /cache_tb/my_bram/UO/gint_inst/xpm_spram_mem_gen/xpm_memory_spram_inst/xpm_memory_base_ins
TEST 1
Write complete: addr=0x00000000, data=0xffffffff
Expected: Cache MISS, State should go OVERWRITE but not evict since curr line is not dirty
Cache[0x00] should now contain: valid=1, dirty=1, tag=0x000000, data=0xffffffff
Results:
Cache[0x00]: Valid=1, Dirty=1, Tag=0x000000, Data=0xffffffff
TEST 2
Write complete: addr=0x00000000, data=0xdeadbeef
Expected: Cache Hit, State should get overwrite no evict
Cache[0x00] should now contain: valid=1, dirty=1, tag=0x000000, data=0xDEADBEEF
Results:
Cache[0x00]: Valid=1, Dirty=1, Tag=0x000000, Data=0xdeadbeef
TEST 3
Write complete: addr=0x00000100, data=0xbeefbeef
Expected: Cache Conflict Miss, should evict then overwite
Cache[0x00] should now contain: valid=1, dirty=1, tag=0x000001, data=0xBEEFBEEF
Results:
Cache[0x00]: Valid=1, Dirty=1, Tag=0x000001, Data=0xbeefbeef
Expected evicted BRAM Data at [00000]: 0xdeadbeef
TEST 4
Read complete: addr=0x00000000, data=0xdeadbeef
Expected: Cache Conflict Miss, should evict, refill with valid BRAM data, and then read
Cache[0x00] should now contain: valid=1, dirty=0, tag=0x000001, data=0xdeadbeef
Results:
Cache[0x00]: Valid=1, Dirty=0, Tag=0x000000, Data=0xdeadbeef
Expected evicted BRAM Data at [00100]: 0xbeefbeef
TEST 5
Read complete: addr=0x00000000, data=0xdeadbeef
Expected: Cache Hit, should read
Cache[0x00] should now contain: valid=1, dirty=0, tag=0x000000, data=0xdeadbeef
Results:
Cache[0x00]: Valid=1, Dirty=0, Tag=0x000000, Data=0xdeadbeef
TEST 6
Read complete: addr=0x00000100, data=0xbeefbeef
Expected:Clean Cache Miss, should refill then read
Cache[0x00] should now contain: valid=1, dirty=0, tag=0x000001, data=0xbeefbeef
Results:
Cache[0x00]: Valid=1, Dirty=0, Tag=0x000001, Data=0xbeefbeef
Read complete: addr=0x000003ff, data=0x11223344
EXPECTED DATA: 0x11223344
Cache[0xff]: Valid=1, Dirty=1, Tag=0x000003, Data=0x11223344
Read complete: addr=0x000004aa, data=0x99887766
EXPECTED DATA: 0x99887766
Cache[0xaa]: Valid=1, Dirty=1, Tag=0x000004, Data=0x99887766
$finish called at time : 825 ns : File "/home/ugrad/imeza/lab6/cache_pico/cache_pico.srcs/sim_1/new/cache_tb.sv" Line 346
relaunch_sim: Time (s): cpu = 00:00:12 ; elapsed = 00:00:13 . Memory (MB): peak = 10861.988 ; gain = 0.000 ; free physical =
Type a Tcl command here


Objects
Protocol Instances
Value
deadbeef,00000000,0
ZZZZZZZZ
ZZZZZZZZ
00040
beefbeef
00040
00000

Waveforms


cache.sv
cache_tb.sv
Untitled 1*
xpm_memory.sv
axi_bram_ctrl_v4_1_rfs.vhd
Name
Value
clk
rst
cpu_awvalid
cpu_awready
cpu_awaddr[31:0]
cpu_awprot[2:0]
cpu_wvalid
cpu_wready
cpu_wdata[31:0]
cpu_wstrb[3:0]
cpu_bvalid
cpu_bready
cpu_arvalid
cpu_arready
cpu_araddr[31:0]
cpu_arprot[2:0]
cpu_rvalid
cpu_rready
cpu_rdata[31:0]
bram_awvalid
bram_awready
605.000 ns
570.000 ns
580.000 ns
590.000 ns
610.000 ns
620.000 ns
630.000 ns
640.000 ns
650.000 ns
00000100
000003ff
beefbeef
11223344
256
3203383023

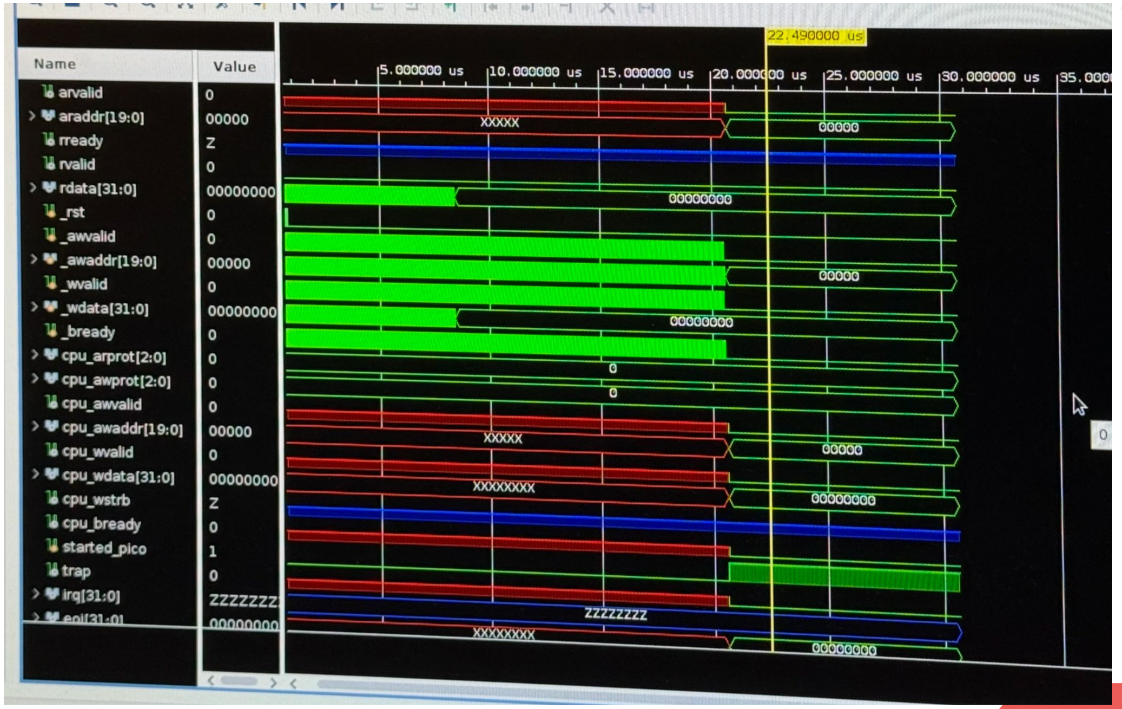
04

RESULTS

Obstacles

Unfortunately, in the end, we weren't able to get our cache to work properly with the Pico and the Bram.

- When testing it with lab3/4 testbench, the Pico was never started



05

REFLECTION

What's the efficiency of our cache implementation?

Pico:

- Example: Pico Read takes 3 clk cycles to complete

Pico plus our Cache:

- Read Hit: 2 clk cycles (one for accepting awaddr, one for sending rdata)
- Write Hit: 2 clk cycles
- Misses are very expensive, due to the number of states and AXI requests to the pico, misses make performance significantly slower
- Worst Case: Read-Miss w Dirty Data
    - Accept the requested address from CPU
    - Write dirty data to BRAM
    - Read requested data from BRAM and load into cache
    - Return requested data to CPU

What can we do to maximize efficiency?

Add more sets/cache lines

- Will can reduce the probability of conflict-misses within and improve the hit-rate

Increase block size

Reduce the amount of states used

- A simpler state machine can probably reduce the number of clock cycles used to handle each memory request

Increase AXI efficiency

- minimize the number of cycles when processing AXI requests

THANKS!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Block Design

MERCURY

Mercury is the closest planet to the Sun and the smallest one in the Solar System—it's only a bit larger than the Moon. The planet's name has nothing to do with the liquid metal, since Mercury was named after the Roman messenger god

VENUS

Venus has a beautiful name and is the second planet from the Sun. It's terribly hot—even hotter than Mercury—and its atmosphere is extremely poisonous. It's the second-brightest natural object in the night sky after the Moon

JUPITER

Jupiter is a gas giant and the biggest planet in the Solar System. It’s the fourth-brightest object in the night sky and it was named after the Roman god of the skies

VENUS

Venus has a beautiful name and is the second planet from the Sun. It’s terribly hot and its atmosphere is poisonous. It’s the second-brightest natural object in the night sky

SATURN

Saturn is a gas giant and has several rings. It’s composed mostly of hydrogen and helium. This planet was named after the Roman god of wealth
and agriculture

FOUR IDEAS

MARS

Despite being red, Mars is a cold place. It’s full of iron oxide dust, which gives the planet its reddish cast, and it’s made of basalt

VENUS

Venus has a beautiful name and is the second planet from the Sun. Venus is terribly hot and has a very poisonous atmosphere

JUPITER

Jupiter is a gas giant and the biggest planet in the Solar System. It’s the fourth-brightest object in the night sky and named after a Roman god

SATURN

Saturn is a gas giant and has several rings. It’s composed mostly of hydrogen and helium. It’s the sixth planet from the Sun

SIX IDEAS

MARS

Despite being red, Mars is actually a cold place. It’s full of iron oxide dust, which gives the planet its reddish cast

VENUS

Venus has a beautiful name and is the second planet from the Sun. It’s terribly hot, even hotter than Mercury

NEPTUNE

Neptune is the farthest planet from the Sun. It’s also the fourth-largest planet by diameter in the Solar System

EARTH

Earth is the third planet from the Sun and the only one that harbors life in the Solar System. We all live on this planet

SATURN

Saturn is a gas giant and has rings. It’s composed mostly of hydrogen and helium. It’s the sixth planet from the Sun

JUPITER

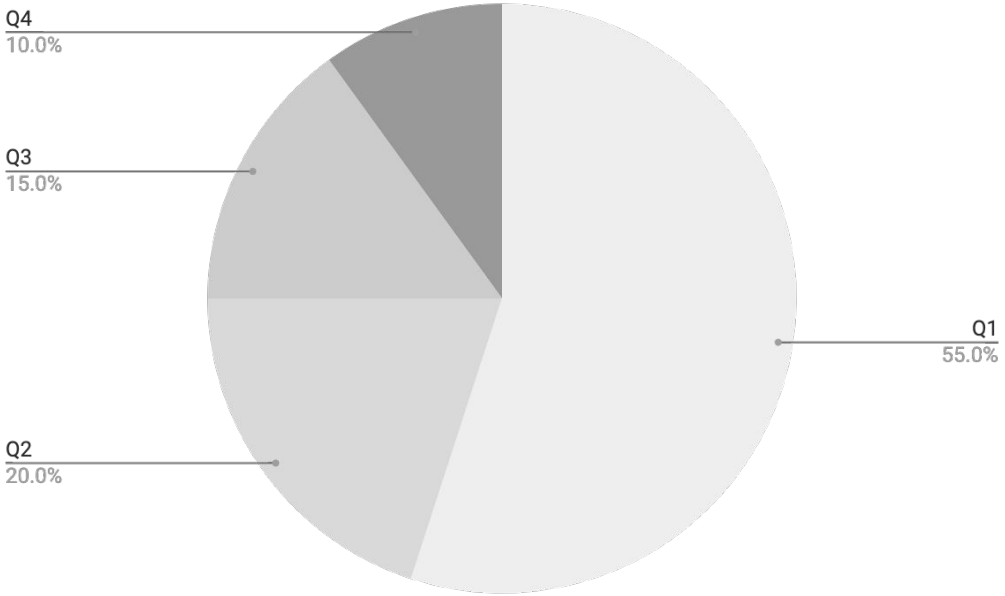
Jupiter is a gas giant and the biggest planet in the Solar System. It’s the fourth-brightest object in the night sky

4,498,300,000

Big numbers catch your audience’s attention


A PICTURE IS WORTH
A THOUSAND WORDS

PIE CHART



Follow the link in the map to modify its data and then paste the new one here. **For more info, click here**

Q1

Mars is very cold

Q2

Mercury is small

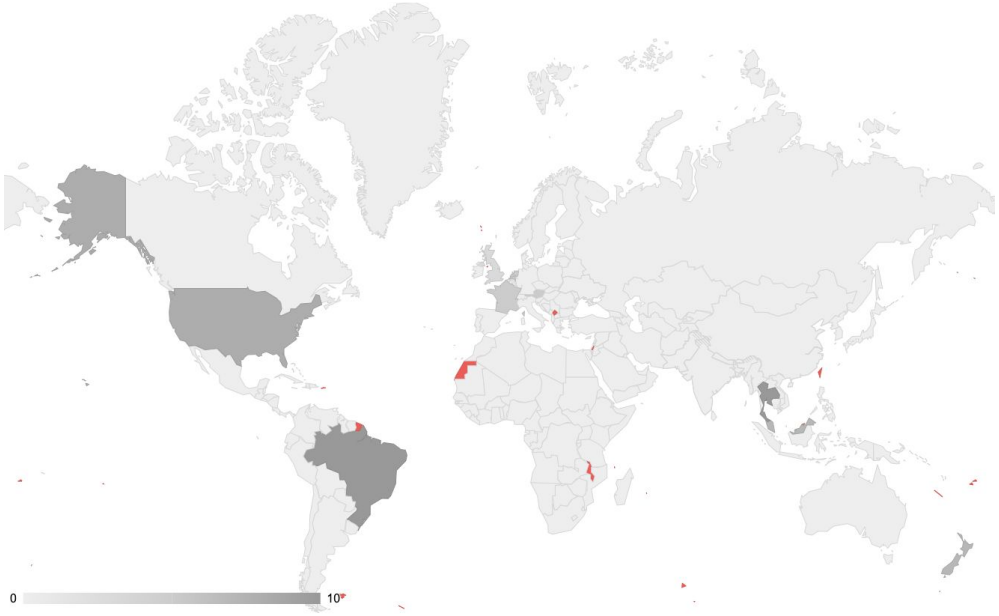
Q3

Venus is very hot

Q4

Earth has life

MAP



Follow the link in the map to modify its data and then paste the new one here. **For more info, click here**

Mercury is the closest planet to the Sun and the smallest of them all

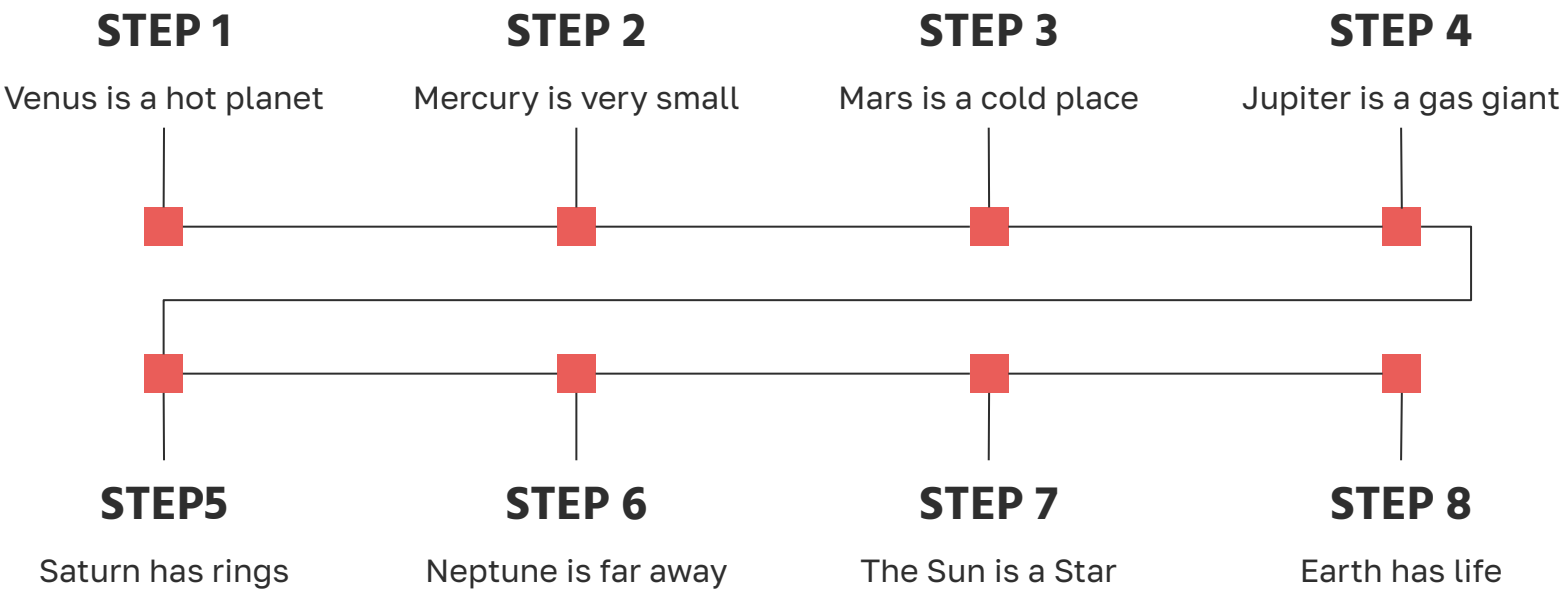
Venus has a beautiful name and is the second planet from the Sun

Mars is full of iron oxide dust, which gives the planet its reddish cast

TABLE

	TEAM A	TEAM B	TEAM C	TEAM D	TEAM E	TEAM F
MERCURY	XX	XX	XX	XX	XX	XX
MARS	XX	XX	XX	XX	XX	XX
SATURN	XX	XX	XX	XX	XX	XX
VENUS	XX	XX	XX	XX	XX	XX
JUPITER	XX	XX	XX	XX	XX	XX
EARTH	XX	XX	XX	XX	XX	XX

TIMELINE


STEP 1
Venus is a hot planet
STEP 2
Mercury is very small
STEP 3
Mars is a cold place
STEP 4
Jupiter is a gas giant
STEP5
Saturn has rings
STEP 6
Neptune is far away
STEP 7
The Sun is a Star
STEP 8
Earth has life

TARGET INFOGRAPHIC

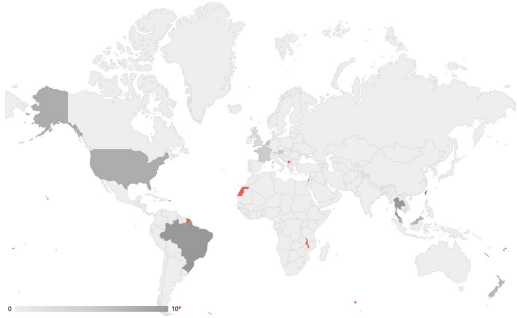
DEMOGRAPHIC

Age	25-50 years old
Gender	Male and female
Occupation	Employees and students
Family situation	Single and married people
Income	$50,000/year

BEHAVIOR

- Mercury is the closest planet to the Sun and the smallest one
- Venus has a beautiful name and is the second planet from the Sun
- Despite being red, Mars is actually a cold place. It's full of iron oxide dust

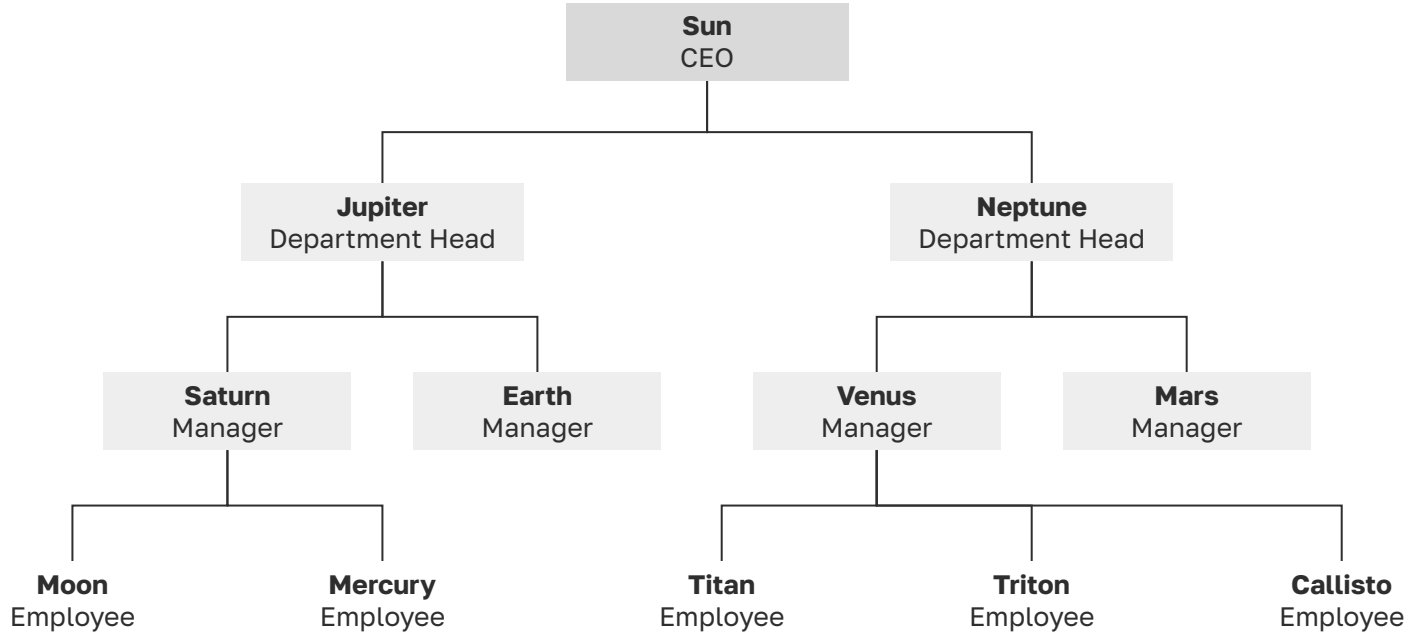
GEOGRAPHIC



REGION	Europe, Asia
AREA	Urban

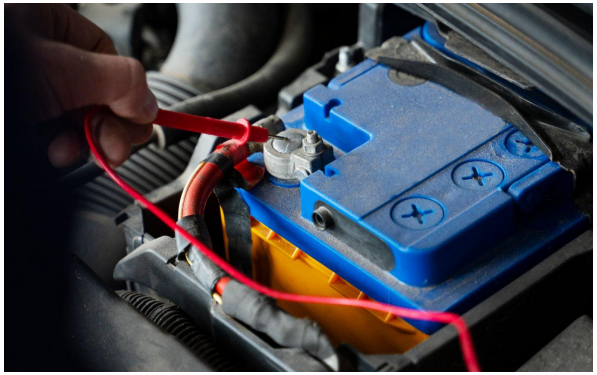
Follow the link in the graph to modify its data and then paste the new one here. **For more info, click here**

ORGANIZATIONAL CHART



ROADMAP INFOGRAPHIC

INITIATIVE	OBJECTIVE	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
Venus is very hot	Venus has a beautiful name and is the second planet												
Earth has life	Earth is the beautiful planet on which humans live												
Mars is very cold	Despite being red, Mars is actually a cold place												
Jupiter is a gas giant	Jupiter is the biggest planet. It was named after a Roman god												
Saturn has rings	Saturn is a planet composed mostly of hydrogen and helium												



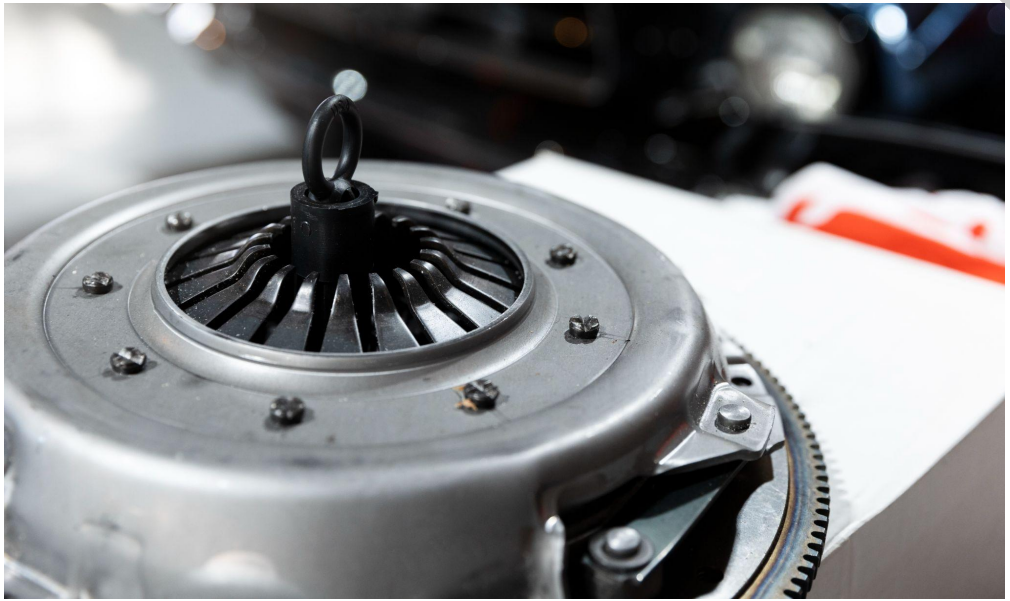


PHOTO SHOWCASE

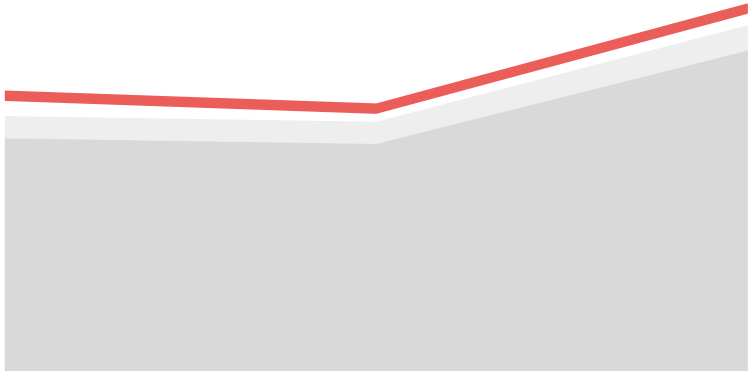
You can replace the images on the screen with others. Just right-click on any of them and select “Replace image”

ALTERNATIVE RESOURCES

Here’s an assortment of alternative resources whose style fits that of this template:

VECTORS

- Mechanic service vertical flyer



RESOURCES

Did you like the resources in this template? Get them on these websites:

VECTORS

- Electrician service poster print template

PHOTOS

- Electric car at the station charging
- High angle man charging his car
- Car being taking care of in workshop
- Close-up of car engine in the repair shop
- Woman charging her electric car at the station

Instructions for use

If you have a free account, in order to use this template, you must credit **Slidesgo** by keeping the **Thanks** slide. Please refer to the next slide to read the instructions for premium users.

As a Free user, you are allowed to:

- Modify this template.
- Use it for both personal and commercial projects.

You are not allowed to:

- Sublicense, sell or rent any of Slidesgo Content (or a modified version of Slidesgo Content).
- Distribute Slidesgo Content unless it has been expressly authorized by Slidesgo.
- Include Slidesgo Content in an online or offline database or file.
- Offer Slidesgo templates (or modified versions of Slidesgo templates) for download.
- Acquire the copyright of Slidesgo Content.

For more information about editing slides, please read our FAQs or visit our blog:
https://slidesgo.com/faqs and **https://slidesgo.com/slidesgo-school**

Instructions for use (premium users)

As a Premium user, you can use this template without attributing **Slidesgo** or keeping the "**Thanks**" slide.

You are allowed to:

- Modify this template.
- Use it for both personal and commercial purposes.
- Hide or delete the “Thanks” slide and the mention to Slidesgo in the credits.
- Share this template in an editable format with people who are not part of your team.

You are not allowed to:

- Sublicense, sell or rent this Slidesgo Template (or a modified version of this Slidesgo Template).
- Distribute this Slidesgo Template (or a modified version of this Slidesgo Template) or include it in a database or in any other product or service that offers downloadable images, icons or presentations that may be subject to distribution or resale.
- Use any of the elements that are part of this Slidesgo Template in an isolated and separated way from this Template.
- Register any of the elements that are part of this template as a trademark or logo, or register it as a work in an intellectual property registry or similar.

For more information about editing slides, please read our FAQs or visit our blog:
https://slidesgo.com/faqs and **https://slidesgo.com/slidesgo-school**

Fonts & colors used

This presentation has been made using the following fonts:

Fira Sans Extra Condensed
(https://fonts.google.com/specimen/Fira+Sans+Extra+Condensed)

Roboto Slab
(https://fonts.google.com/specimen/Roboto+Slab)

Storyset

Create your Story with our illustrated concepts. Choose the style you like the most, edit its colors, pick the background and layers you want to show and bring them to life with the animator panel! It will boost your presentation. Check out **how it works**.

Pana

Amico

Bro

Rafiki

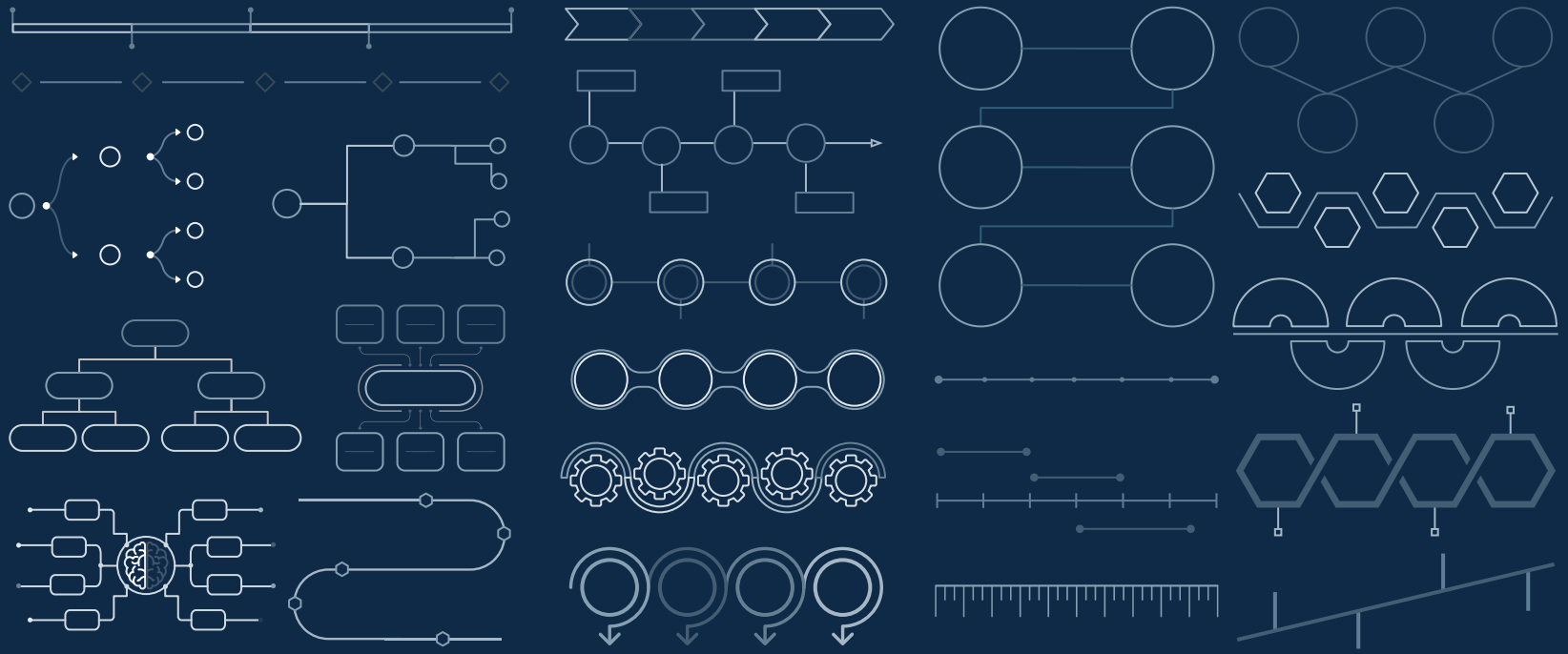
Cuate

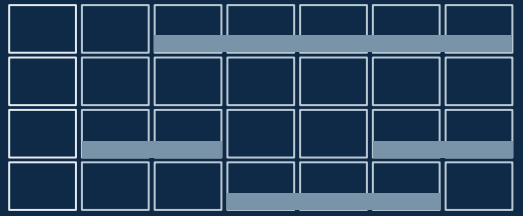
Use our editable graphic resources...

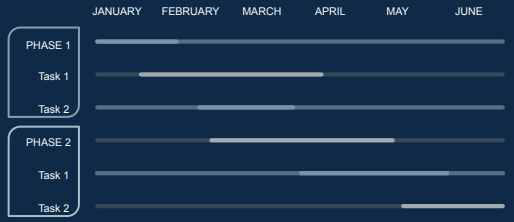
You can easily **resize** these resources without losing quality. To **change the color**, just ungroup the resource and click on the object you want to change. Then, click on the paint bucket and select the color you want. Group the resource again when you're done. You can also look for more **infographics** on Slidesgo.

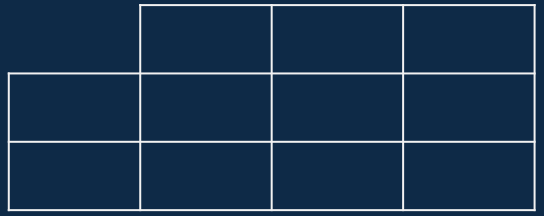


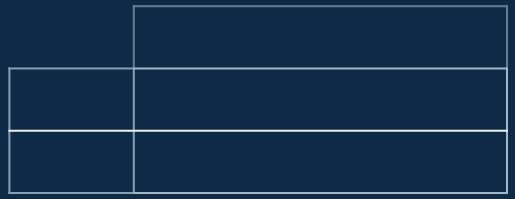


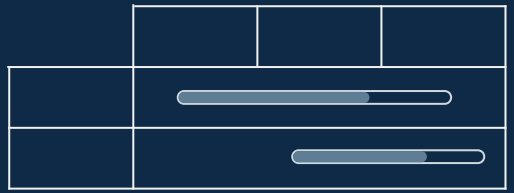



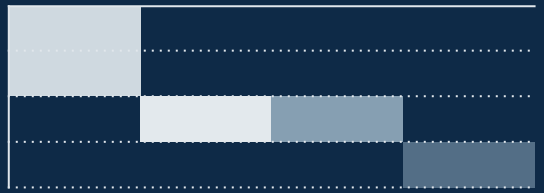


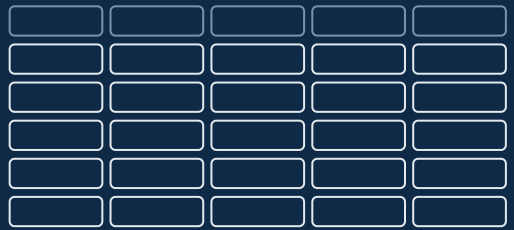

JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
PHASE 1
Task 1
Task 2
PHASE 2
Task 1
Task 2

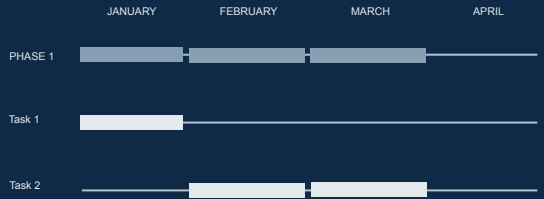


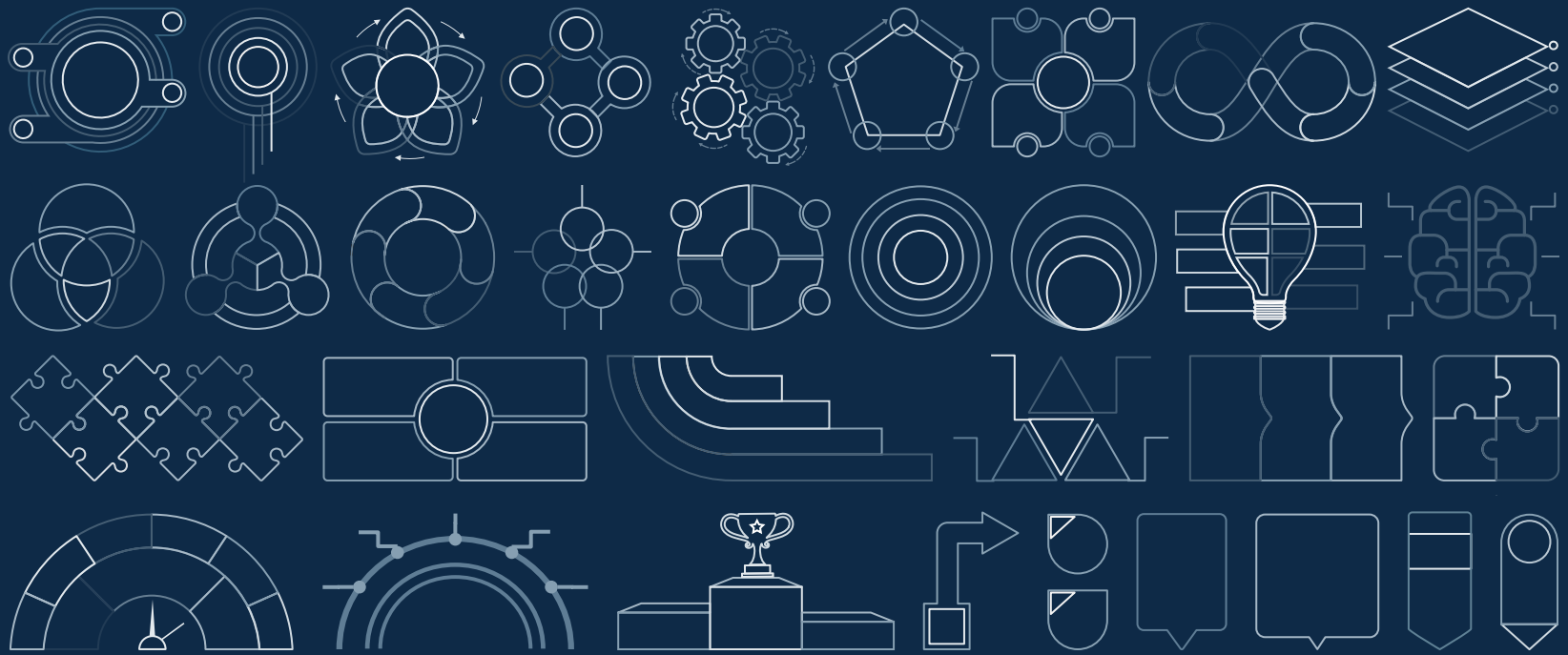


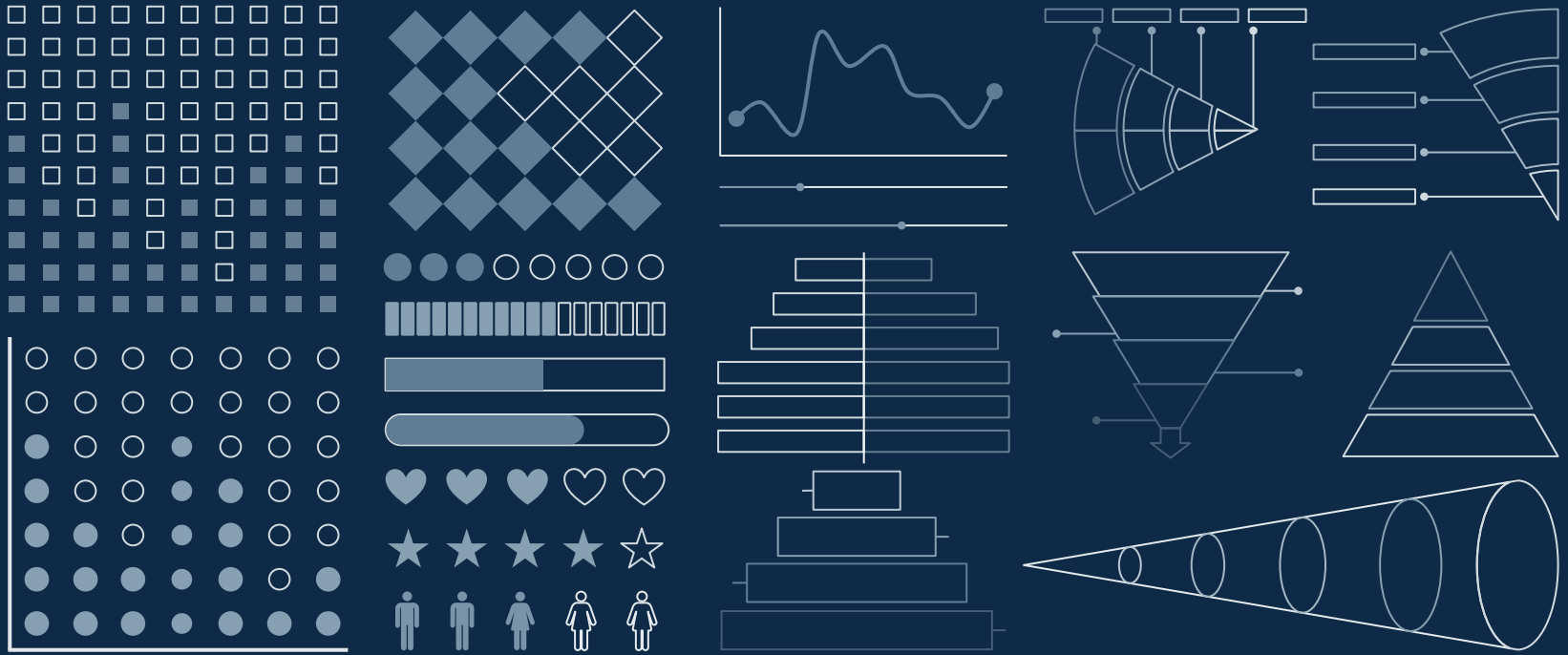







JANUARY
FEBRUARY
MARCH
APRIL
PHASE 1
Task 1
Task 2





...and our sets of editable icons

You can **resize** these icons without losing quality.
You can **change the stroke and fill color**; just select the icon and click on the **paint bucket/pen**.
In Google Slides, you can also use **Flaticon's extension**, allowing you to customize and add even more icons.

Educational Icons

Medical Icons

Business Icons

Teamwork Icons

Help & Support Icons

Avatar Icons

Creative Process Icons

Performing Arts Icons

Nature Icons

SEO & Marketing Icons


slidesgo