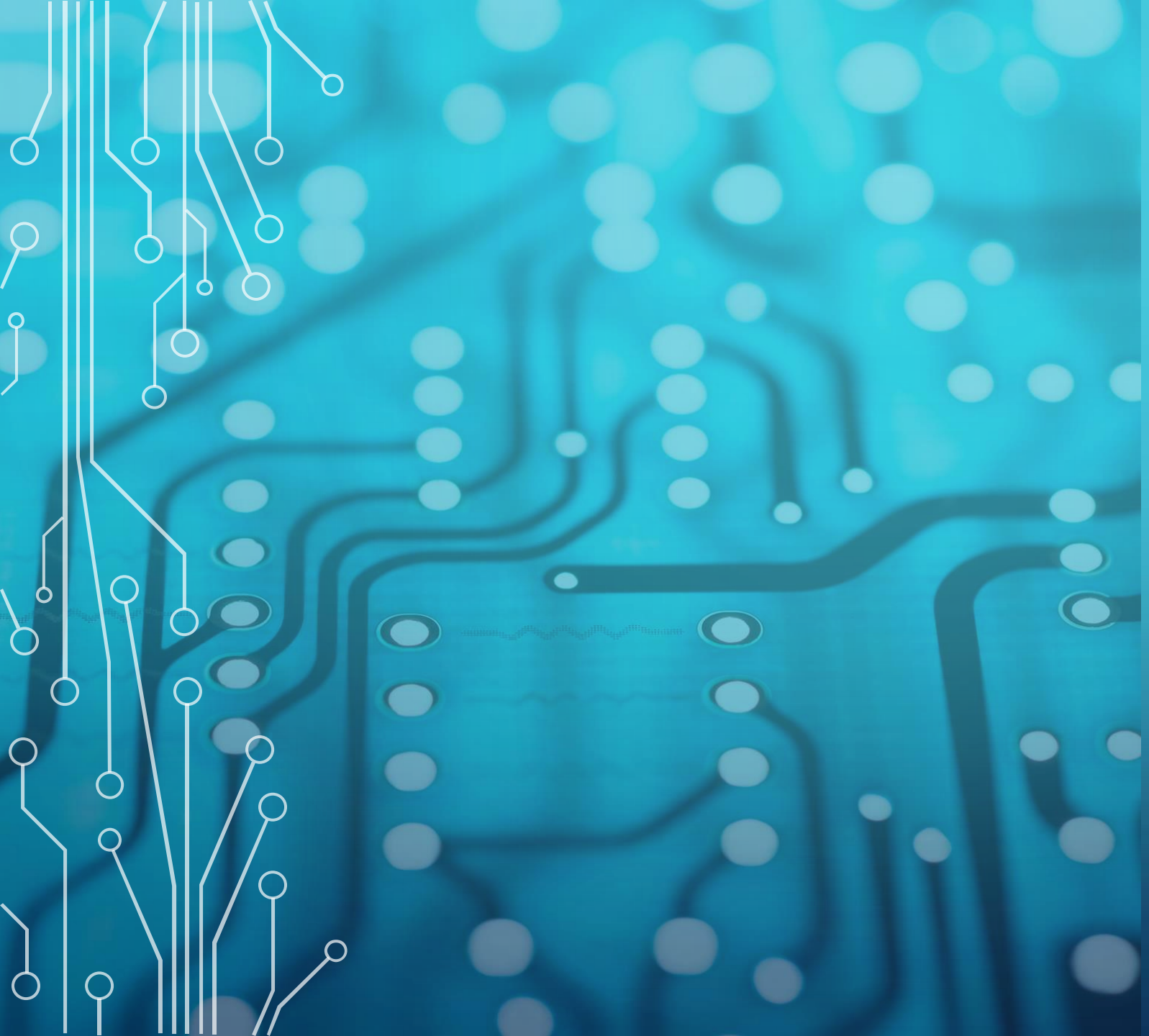
NEORV32 RUNS ON BRAVE NG-MEDIUM

KLEMEN BRAVHAR



AGENDA

- My experience with BRAVE technology in the past and present
- NEORV32 processor
- Implementation of NEORV32 on Ng-Medium FPGA
- Evaluation and benchmarking of NEORV32 on Ng-Medium development board

PAST WORKFLOW EXPERIENCE WITH BRAVE TECHNOLOGY

Functional simulation of RTL

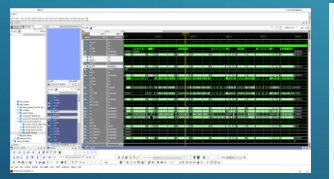
Test functionality of RTL on FPGA from other vendors

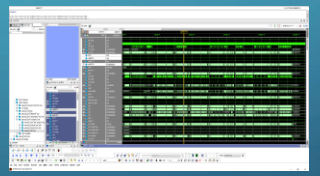
Run synthesis with NxMap 2.8.6 / NxPython

Simulation of synthesized design

Run Place&Route with NxMap 2.8.6 / NxPython

Test functionality on the BRAVE development board



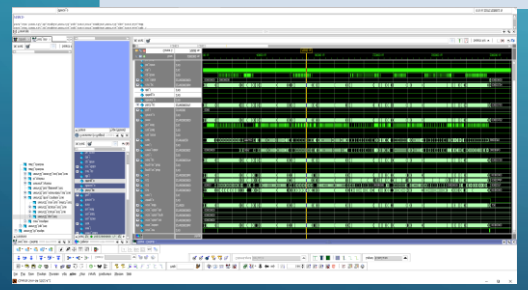


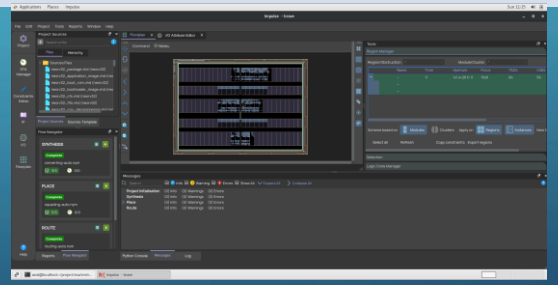
PRESENT WORKFLOW EXPERIENCE WITH BRAVE TECHNOLOGY

Functional simulation of RTL

Run synthesis and Place&Route with Impulse/NxPython

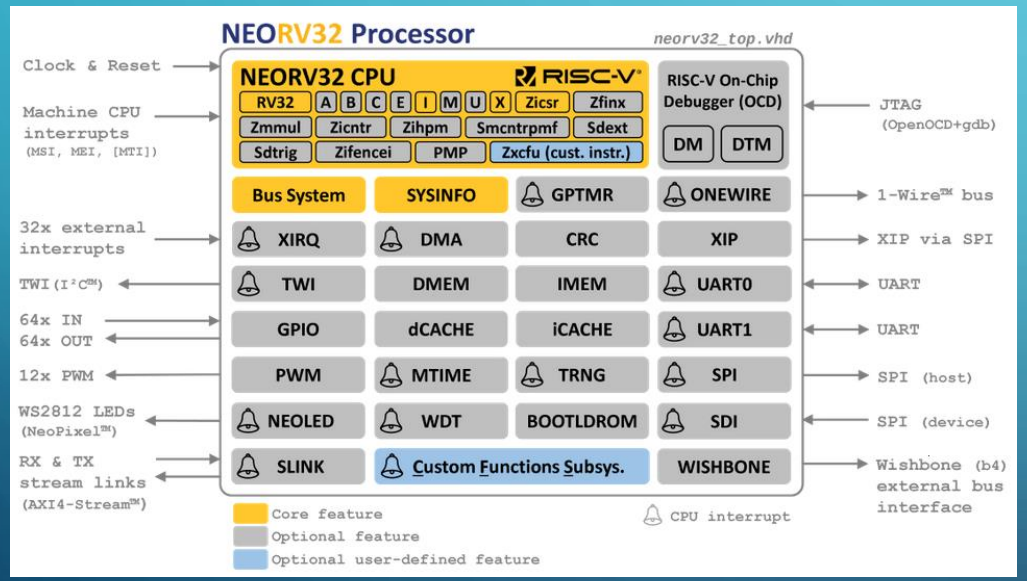
Test functionality on the BRAVE development board





NEORV32 PROCESSOR - SOC

ARCHITECTURE OF NEORV32 RISK-V PROCESSOR

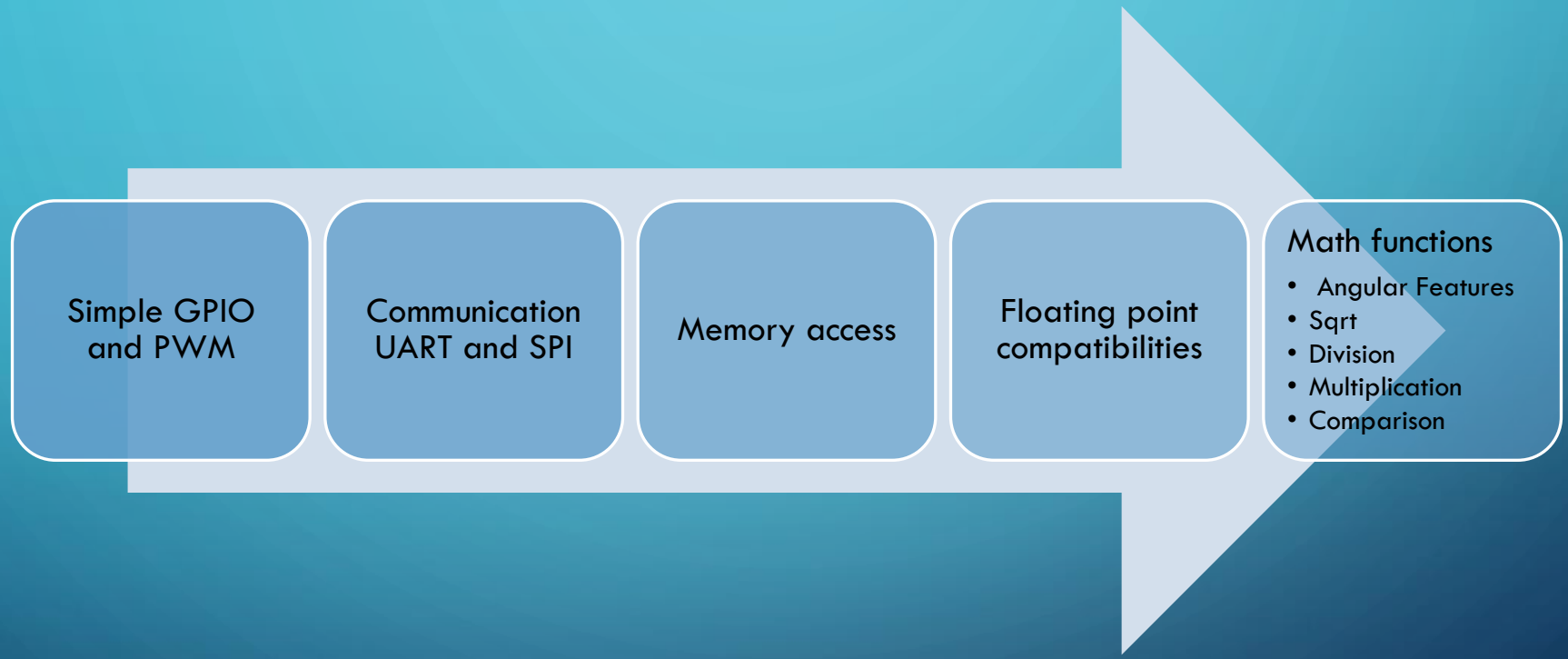


Source: https://github.com/stnolting/neorv32

KEY FEATURES

- Supports RISC-V instruction set => ISA extension
- Intuitive to use
- VHDL and Verilog
- User can write its own functions
- Flexible memory space of instructions and data memory
- It supports gcc and g++ compilers
- SW code can be simulated together with the NEORV32 core
- SW code can be uploaded to internal “FLASH” memory during core synthesis steps or via UART

EVALUATING AND BENCHMARKING IMPLEMENTED NEORV32 SOC ON NG-MEDIUM



EVALUATING AND BENCHMARKING IMPLEMENTED NEORV32 SOC ON NG-MEDIUM

NEORV32

RISC-V

+

IMPLEMENTATION OF NEORV32 ON BRAVE NG-MEDIUM

4-LUT	DFF	XLUT	1-bit carry	Register file block	DSP	Memory Block	WFG	PLL
3788 / 32256 (12%)	1921 / 32256 (6%)	0 / 2016 (0%)	830 / 8064 (11%)	4 / 168 (3%)	0 / 112 (0%)	22 / 56 (40%)	1 / 32 (4%)	1 / 4 (25%)

Domain		Frequency		Hold/Removal Summary			Setup/Recovery Summary		
Source	Target	Required	Actual	Slack	Minimum Data Arrival Time	Minimum Required Relationship	Slack	Maximum Data Arrival Time	Maximum Required Relationship
Input	rg~s_clk (Falling)	-	-	-	3.219ns	-	-	3.396ns	-
Input	rg~s_clk (Rising)	-	-	-	2.665ns	-	-	2.798ns	-
rg~s_clk (Rising)	Output	-	-	-	7.500ns	-	-	9.337ns	-
rg~s_clk (Falling)	rg~s_clk (Falling)	-	-	1.042ns	1.042ns	0ps	25.420ns	1.236ns	26.656ns
rg~s_clk (Falling)	rg~s_clk (Rising)	-	-	16.893ns	3.565ns	-13.328ns	9.587ns	3.741ns	13.328ns
rg~s_clk (Rising)	rg~s_clk (Rising)	37.515 MHz	64.842 MHz	1.000ns	1.000ns	0ps	11.234ns	15.422ns	26.656ns
Total									6

THANK YOU FOR YOUR TIME
QUESTIONS