Soft CPU Performance on modern FPGAs

http://j.mp/softcpu-on-fpgas

Author: Tim ‘mithro’ Ansell <me@mith.ro>
Status: WIP
Last Updated: 23 September 2019

TL;DR

Type	Max LUTs	Part	Min fmax	DMIPS/Mhz
Microcontroller	1000	Artix 7	250MHz+	0.5
		Ultrascale	400MHz+	
		Ultrascale+	600MHz+	
Linux Capable	2000	Artix 7	150MHz+	1.25
		Ultrascale	250MHz+	
		Ultrascale+	350MHz+	

Aim

There seems to be poor understanding behind the typical performance of soft processors that is possible on modern FPGAs. This document aims to give a high level “rule of thumb” style information around expected level of performance for soft processors (primarily RISC-V) on modern (post 2010) FPGA architectures.

Table of Contents

Background

FPGAs are heavily used for simulation of ASIC designs, however there are significantly different trade offs between designing an ASIC and designing a soft processor which efficiently performs on an FPGA. This leads to people thinking the possible speed of soft processors to be significantly lower than actually achievable.

This document mainly talks about the Xilinx Series 7 and Spartan 6 FPGA lines as they are the FPGAs the Author is most familiar with. Similar performance magnitudes can be expected from any modern FPGA released by companies like Intel (Altera), Lattice or Archonix.

Reference Projects

- VexRISC-V - https://github.com/SpinalHDL/VexRiscv
- Picorv32 - https://github.com/cliffordwolf/picorv32
- MicroBlaze -
- NEORV32 - https://github.com/stnolting/neorv32
- NaxRiscv - https://github.com/SpinalHDL/NaxRiscv

Floating Point Unit

Creating an efficient floating point unit is very different task than creating a general C-code machine and so it is separated into its own section.

Microcontroller Class

Microcontroller class processors are generally have the following characteristics;

- Designed to run bare metal C or minimal RTOS
- Firmware runs from embedded resources
- Generally, no MMU
- Memory;
    - Internal to device

    - Sizes in O(kilobytes)
- Short pipeline and low interrupt latency

Only considering 32bit architectures with GCC support. No FPU is included in these designs.

Examples

- RISC-V Examples
    - VexRISCV
    - PicoRV32
    - NEORV32
- Non-RISC-V Examples
    - LatticeMico32
    - Microblaze “Small configuration”
    - NIOS
    - mor1kx

Xilinx Series 7 FPGA - Expected numbers

	Range	Typical	Minimal	Maximum
LUT usage	500 → 1000	700	300	
F_{max}	300MHz → 400MHz	300 MHz	250 MHz	
DMips/MHz	0.5 → 1.0	0.5	0.5	
Coremark/Mhz	--	--	--	

Xilinx Ultrascale FPGA - Expected numbers

Ultrascale resource usage numbers are similar to Xilinx Series 7 but designs should get ***large*** Fmax increases.

	Range	Typical	Minimal	Maximum
LUT usage	500 → 1000			
F_{max}				
DMips/MHz				
Coremark/Mhz				

Xilinx Ultrascale+ FPGA - Expected numbers

Ultrascale+ resource usage numbers are similar to Xilinx Series 7 but designs should get ***significant*** Fmax increases.

	Range	Typical	Minimal	Maximum
LUT usage	500 → 1000			
F_{max}				

DMips/MHz				
Coremark/Mhz				

Microprocessor Class

Microprocessor class processors are generally have the following characteristics;

- Designed to run a full operating system like Linux
- Have an MMU
- Can have SMP and multicore
- Memory;
    - Sizes in O(100 megabytes) → O(gigabytes)
    - External DDR memory
- Can have long pipeline sometimes with out of order execution and speculation

Only considering 32bit architectures with GCC+Linux support. No FPU is included in these designs.

Examples

- RISC-V Examples
    - VexRISCV - https://github.com/SpinalHDL/VexRiscv
- Non-RISC-V Examples
    - Microblaze “Linux configuration”
    - mor1kx - https://github.com/openrisc/mor1kx

Further Reading

- Running Linux with Quad-core SMP in LiteX/VexRiscv on Arty A7
- SpinalHDL/VexRiscv: A FPGA friendly 32 bit RISC-V CPU implementation

Xilinx Series 7 FPGA - Expected numbers

	Range	**Typical**	**Minimal**	**Maximum**
LUT usage	1000 → 2500	1500	1000	
F_{max}	200MHz → 300MHz	250 MHz	200 MHz	
DMips/MHz	1.0 → 1.5	1.25	1.0	
Coremark/Mhz	2.0 → 3.0	2.5	2.0	

Xilinx Ultrascale FPGA - Expected numbers

	Range	**Typical**	**Minimal**	**Maximum**
LUT usage				
F_{max}				

DMips/MHz				
Coremark/Mhz				

Xilinx Ultrascale+ FPGA - Expected numbers

	Range	**Typical**	**Minimal**	**Maximum**
LUT usage				
F_{max}				
DMips/MHz				
Coremark/Mhz				

Raw Data

VexRISCV

VexRISCV is the most popular CPU used with the LiteX SoC generation environment, which also supports BlackParrot, Rocket and PicoRV32 (see https://github.com/enjoy-digital/litex/wiki/CPUs) as a full SoC on the same FPGA boards.

```
VexRiscv small (RV32I, 0.52 DMIPS/Mhz, no datapath bypass, no interrupt) ->
    Artix 7     -> 243 Mhz 504 LUT 505 FF
    Cyclone V   -> 174 Mhz 352 ALMs
    Cyclone IV  -> 179 Mhz 731 LUT 494 FF
    iCE40       -> 92 Mhz 1130 LC

VexRiscv small (RV32I, 0.52 DMIPS/Mhz, no datapath bypass) ->
    Artix 7     -> 240 Mhz 556 LUT 566 FF
    Cyclone V   -> 194 Mhz 394 ALMs
    Cyclone IV  -> 174 Mhz 831 LUT 555 FF
    iCE40       -> 85 Mhz 1292 LC

VexRiscv small and productive (RV32I, 0.82 DMIPS/Mhz)  ->
    Artix 7     -> 232 Mhz 816 LUT 534 FF
    Cyclone V   -> 155 Mhz 492 ALMs
    Cyclone IV  -> 155 Mhz 1,111 LUT 530 FF
    iCE40       -> 63 Mhz 1596 LC

VexRiscv small and productive with I$ (RV32I, 0.70 DMIPS/Mhz, 4KB-I$)  ->
    Artix 7     -> 220 Mhz 730 LUT 570 FF
    Cyclone V   -> 142 Mhz 501 ALMs
    Cyclone IV  -> 150 Mhz 1,139 LUT 536 FF
    iCE40       -> 66 Mhz 1680 LC

VexRiscv full no cache (RV32IM, 1.21 DMIPS/Mhz 2.30 Coremark/Mhz, single cycle barrel shifter, debug
module, catch exceptions, static branch) ->
    Artix 7     -> 216 Mhz 1418 LUT 949 FF
    Cyclone V   -> 133 Mhz 933 ALMs
    Cyclone IV  -> 143 Mhz 2,076 LUT 972 FF

VexRiscv full (RV32IM, 1.21 DMIPS/Mhz 2.30 Coremark/Mhz with cache trashing, 4KB-I$,4KB-D$, single
cycle barrel shifter, debug module, catch exceptions, static branch) ->
    Artix 7     -> 199 Mhz 1840 LUT 1158 FF
    Cyclone V   -> 141 Mhz 1,166 ALMs
    Cyclone IV  -> 131 Mhz 2,407 LUT 1,067 FF

VexRiscv full max perf (HZ*IPC) -> (RV32IM, 1.38 DMIPS/Mhz 2.57 Coremark/Mhz, 8KB-I$,8KB-D$, single
cycle barrel shifter, debug module, catch exceptions, dynamic branch prediction in the fetch stage,
branch and shift operations done in the Execute stage) ->
    Artix 7     -> 200 Mhz 1935 LUT 1216 FF
    Cyclone V   -> 130 Mhz 1,166 ALMs
    Cyclone IV  -> 126 Mhz 2,484 LUT 1,120 FF

VexRiscv full with MMU (RV32IM, 1.24 DMIPS/Mhz 2.35 Coremark/Mhz, with cache trashing, 4KB-I$, 4KB-D$,
single cycle barrel shifter, debug module, catch exceptions, dynamic branch, MMU) ->
    Artix 7     -> 151 Mhz 2021 LUT 1541 FF
    Cyclone V   -> 124 Mhz 1,368 ALMs
    Cyclone IV -> 128 Mhz 2,826 LUT 1,474 FF

VexRiscv linux balanced (RV32IMA, 1.21 DMIPS/Mhz 2.27 Coremark/Mhz, with cache trashing, 4KB-I$,
4KB-D$, single cycle barrel shifter, catch exceptions, static branch, MMU, Supervisor, Compatible with
mainstream linux) ->
```

```
    Artix 7     -> 180 Mhz 2883 LUT 2130 FF
    Cyclone V   -> 131 Mhz 1,764 ALMs
    Cyclone IV  -> 121 Mhz 3,608 LUT 2,082 FF
```

Depending the CPU configuration, on the ICE40-hx8k FPGA with icestorm for synthesis, the full SoC has the following area/performance :

- RV32I interlocked stages => 51 Mhz, 2387 LC 0.45 DMIPS/Mhz
- RV32I bypassed stages => 45 Mhz, 2718 LC 0.65 DMIPS/Mhz

```
Murax interlocked stages (0.45 DMIPS/Mhz, 8 bits GPIO) ->
    Artix 7     -> 216 Mhz 1109 LUT 1201 FF
    Cyclone V   -> 182 Mhz 725 ALMs
    Cyclone IV  -> 147 Mhz 1,551 LUT 1,223 FF
    iCE40       ->  64 Mhz 2422 LC (nextpnr)

MuraxFast bypassed stages (0.65 DMIPS/Mhz, 8 bits GPIO) ->
    Artix 7     -> 224 Mhz 1278 LUT 1300 FF
    Cyclone V   -> 173 Mhz 867 ALMs
    Cyclone IV  -> 143 Mhz 1,755 LUT 1,258 FF
    iCE40       ->  66 Mhz 2799 LC (nextpnr)
```

VexRiscv linux 16KB I$ 16 KB D$ ->
  Artix 7 -> 187 Mhz 3614 LUT 2466 FF

VexRiscv linux 16KB I$ 16 KB D$ + FPU 64 bits ->
  Artix 7 -> 165 Mhz 7504 LUT 5805 FF

PicoRV32

PicoRV32 can also be used through the LiteX SoC generation environment, which also supports BlackParrot, Rocket and VexRISCV (see https://github.com/enjoy-digital/litex/wiki/CPUs) as a full SoC on the same FPGA boards.

Performance

Dystone

Dhrystone benchmark results: 0.516 DMIPS/MHz (908 Dhrystones/Second/MHz)

Resource Usage

Xilinx Series 7

Core Variant	Slice LUTs	LUTs as Memory	Slice Registers
PicoRV32 (small)	761	48	442
PicoRV32 (regular)	917	48	583

PicoRV32 (large)	2019	88	1085

Frequency

While this core has a very high f_{max}, do note that this core has very low DMIPs/MHz compared to most. PicoRV32 is optimized for high f_{max} so it can be used in the same clock domain as a high performance peripheral.

Device	Device	Speedgrade	Clock Period (Freq.)
Xilinx Kintex-7T	xc7k70t-fbg676-2	-2	2.4 ns (416 MHz)
Xilinx Kintex-7T	xc7k70t-fbg676-3	-3	2.2 ns (454 MHz)
Xilinx Virtex-7T	xc7v585t-ffg1761-2	-2	2.3 ns (434 MHz)
Xilinx Virtex-7T	xc7v585t-ffg1761-3	-3	2.2 ns (454 MHz)
Xilinx Kintex UltraScale	xcku035-fbva676-2-e	-2	2.0 ns (500 MHz)
Xilinx Kintex UltraScale	xcku035-fbva676-3-e	-3	1.8 ns (555 MHz)
Xilinx Virtex UltraScale	xcvu065-ffvc1517-2-e	-2	2.1 ns (476 MHz)
Xilinx Virtex UltraScale	xcvu065-ffvc1517-3-e	-3	2.0 ns (500 MHz)
Xilinx Kintex UltraScale+	xcku3p-ffva676-2-e	-2	1.4 ns (714 MHz)
Xilinx Kintex UltraScale+	xcku3p-ffva676-3-e	-3	1.3 ns (769 MHz)
Xilinx Virtex UltraScale+	xcvu3p-ffvc1517-2-e	-2	1.5 ns (666 MHz)
Xilinx Virtex UltraScale+	xcvu3p-ffvc1517-3-e	-3	1.4 ns (714 MHz)

MicroBlaze

https://en.wikipedia.org/wiki/MicroBlaze

> *The performance-optimized version expands the execution pipeline to 5 stages, allowing top speeds of more than 700 MHz (on Virtex UltraScale+ FPGA family)*

https://www.xilinx.com/support/documentation/white_papers/wp501-microblaze.pdf

Table 3: **MicroBlaze Preset Performance in Xilinx Cost-Optimized Portfolio Devices**

Device (Speed Grade)		Microcontroller (1.1DMIPs/MHz)		Real-Time Processor (1.3DMIPs/MHz)		Applications Processor (1.4DMIPs/MHz)	
		F_{MAX}	DMIPs	F_{MAX}	DMIPs	F_{MAX}	DMIPs
FPGA:	Spartan-7 (-2)	194	260	161	216	131	140
	Artix-7 (-3)	226	303	184	247	153	164
SoC:	Zynq-7000S (-2)	185	248	155	208	125	134
	Zynq-7000 (-3)	224	300	181	243	168	180

http://tnt.etf.bg.ac.rs/~oe4nrs/pdf/mb_faq.pdf

Depending on the configured options and target FPGA, MicroBlaze uses between about 800 and 2600 look-up tables (LUTs).

https://www.xilinx.com/support/documentation/ip_documentation/ru/microblaze-mcs.html

Performance and Resource Utilization for MicroBlaze MCS v3.0

Virtex UltraScale - 397 Fmax(MHz) + 626 LUTs

MicroBlaze Performance Metrics: Based on Vivado 2019.2

Device	Microcontroller (1.04 DMIPs/MHz)		Real-Time Processor (1.31 DMIPs/MHz)		Applications Processor (1.31 DMIPs/MHz)	
	Fmax	DMIPS	Fmax	DMIPS	Fmax	DMIPS
Cost-Optimized Portfolio Devices						
Spartan-7 (-2)	**178 MHz**	185	**155 MHz**	203	**120 MHz**	157
Artix-7 (-3)	**204 MHz**	212	**172 MHz**	225	**146 MHz**	191
Zynq 7000S (-2)	**187 MHz**	194	**156 MHz**	204	**129 MHz**	169
Zynq-7000 (-3)	**212 MHz**	220	**171 MHz**	224	**141 MHz**	185
FPGAs, 3D ICs, and MPSoCs						
Kintex-7 (-3)	**298 MHz**	310	**228 MHz**	299	**204 MHz**	267
Virtex-7 (-3)	**300 MHz**	312	**238 MHz**	312	**208 MHz**	272
Kintex UltraScale (-3)	**393 MHz**	409	**280 MHz**	367	**242 MHz**	317
Virtex UltraScale (-3)	**384 MHz**	399	**283 MHz**	371	**245 MHz**	321
Kintex UltraScale+ (-3)	**518 MHz**	539	**384 MHz**	503	**345 MHz**	452
Virtex UltraScale+ (-3)	**505 MHz**	525	**396 MHz**	519	**327 MHz**	428
Zynq UltraScale+ MPSoC (-3)	**493 MHz**	513	**379 MHz**	496	**329 MHz**	431

NEORV32

A size-optimized, customizable full-scale 32-bit RISC-V soft-core CPU and SoC written in platform-independent VHDL.
The core implements a little-endian von-Neumann architecture using two pipeline stages. Each stage uses a multi-cycle processing scheme. The CPU supports three privilege levels (machine and optional user and debug_mode), three standard RISC-V machine interrupts (MTI, MEI, MSI), a single non-maskable interrupt plus 16 *fast interrupt requests* as custom extensions. It also supports all standard RISC-V exceptions (instruction/load/store misaligned address & bus access fault, illegal instruction, breakpoint, environment call). As a special "execution safety" extension, *all* invalid, reserved or malformed instructions will raise an exception.

- https://stnolting.github.io/neorv32/#_rationale
- https://github.com/stnolting/neorv32#getting-started

CPU (including Zicsr extension)	Executable Size	CoreMark Score	CoreMarks/MHz	Total Clock Cycles	Executed Instructions	Average CPI
rv32i	28 756 bytes	36.36	**0.3636**	5595750503	1466028607	**3.82**
rv32imc	22 008 bytes	68.97	**0.6897**	2981786734	611814918	**4.87**
Rv32imc + FAST_MUL_EN + FAST_SHIFT_EN	22 008 bytes	90.91	**0.9091**	2265135174	611814948	**3.70**

See https://stnolting.github.io/neorv32/#_fpga_implementation_results

GRVI Phalanx - austere RISC-V processing element

http://fpga.org/wp-content/uploads/2017/08/HotChips29-GRVI-Phalanx-poster.png

Xilinx Ultrascale
LUTs: 320
Fmax: 375MHz

Taiga

https://gitlab.com/sfu-rcl/Taiga

RISC-V RV32IMA

Xilinx Ultrascale+
LUTs: 2,189
FFs: 1,135
Freq: 196MHz

Ariane (aka cva6 from EPFL)

https://github.com/lowRISC/ariane
https://github.com/openhwgroup/cva6

Ariane is an ASIC core and hence gets ***terrible*** numbers on an FPGA.

RISC-V RV64IMAC

Xilinx Ultrascale+
LUTs: 26,126
FFs: 19,910
Freq: 108MHz

Black Parrot

https://github.com/black-parrot/black-parrot

BlackParrot is an ASIC core and hence gets ***terrible*** numbers on an FPGA.

RISC-V RV64IMAFD

Xilinx Ultrascale+
LUTs: 32,759
FFs: 11,709
Freq: 63MHz

Note: Getting Black Parrot thru Vivado is very challenging (HDL language issues, and the FPGA projects are stale.) The critical paths appear to go through arithmetic elements, so it is possible that it fails to leverage hardened blocks correctly.

BlackParrot can also be used through the LiteX SoC generation environment, which also supports PicoRV32, Rocket and VexRISCV (see https://github.com/enjoy-digital/litex/wiki/CPUs) as a full SoC on the same FPGA boards.

NaxRiscv

NaxRiscv is a Out of order superscalar softcore. It support both RV32 and RV64 with upstream linux.

For the following configuration :

- RV32IMASU, dual issue,OoO, linux compatible
- 64 bits fetch, 2 decode, 3 issue, 2 retire
- Shared issue queue with 32 entries
- Renaming with 64 physical registers
- 3 execution unit (2*Int/Shift, 1*Branch/load/store/mul/div/csr/env)
- LSU with 16 load queue, 16 store queue
- Load hit predictor (3 cycles load to use delay)
- Store to load bypass / hazard free predictor
- I$ 16KB/4W, D$ 16KB/4W 2 refill 2 writeback slots
- MMU with ITLB 6 way/192 entries, DTLB 6 way/192 entries
- BTB 1 way/512 entries, GSHARE 1 way/4KB, RAS 32 entries

Performance :

- Dhrystone : 2.59 DMIPS/Mhz 1.42 IPC (-O3 -fno-common -fno-inline)
- Coremark : 4.70 Coremark/Mhz 1.19 IPC (-O3 and so many more random flags)
- Embench-iot : 1.55 baseline 1.32 IPC (-O2 -ffunction-sections)

On Artix 7 speed grade 3 :

- 14.6 KLUT, 9.8 KFF, 12.5 BRAM, 4 DSP
- 145 Mhz

For RV64IMASU In a similar configuration than the above RV32 (2*Int/Shift, 1*Branch/load/store/mul/div/csr/env)

Performance :

- Dhrystone : 2.54 DMIPS/Mhz (-O3 -fno-common -fno-inline)
- Coremark : 4.75 Coremark/Mhz (-O3, u32 as s32 and so many more random flags)
- Embench-iot : 1.70 baseline (-O2 -ffunction-sections)

On Artix 7 speed grade 3 :

- 19.5 KLUT, 12.0 KFF, 12.5 BRAM, 16 DSP
- 140 Mhz

So overall, the RV64 support do not has too much of an impact compared to RV32, mostly because the current critical path are in the addresses and control paths, which stays relatively similar between the two (39 bits for RV64, 32 bits for RV32).