Efabless Caravel Openframe

Caravel Openframe project example

Description:

The efabless Caravel Openframe chip is a ready-to-use test harness for creating designs with the Google/SkyWater 130nm open PDK. The Openframe project example comprises a small RISC-V microprocessor based on the simple 2-cycle PicoRV32 RISC-V core implementing the RV32IMC instruction set (see http://riscv.org/) and a 32-bit wishbone bus.

Core:

The processor core is the PicoRV32 design (see http://github.com/YosysHQ/picorv32). The full core description is available from the github site. The hardware implementation is the "large" variant, incorporating options IRQ, MUL, DIV, BARREL_SHIFTER, and COMPRESSED_ISA (16-bit instructions).

Core clock rate: 25 MHz maximum over all PVT conditions.

Features:

Functions/features of the SoC include:

- 1 SPI flash controller
- 1 UART
- 1 SPI master
- 2 counter-timers
- 39 general-purpose input/output channels
- 4k word (32768 bytes × 8 bits) on−board SRAM
- All-digital frequency-locked loop clock multiplier

License:

Both the Caravel Openframe chip design and the Caravel Openframe project example are open-source designs, licensed under the terms of Apache 2.0.

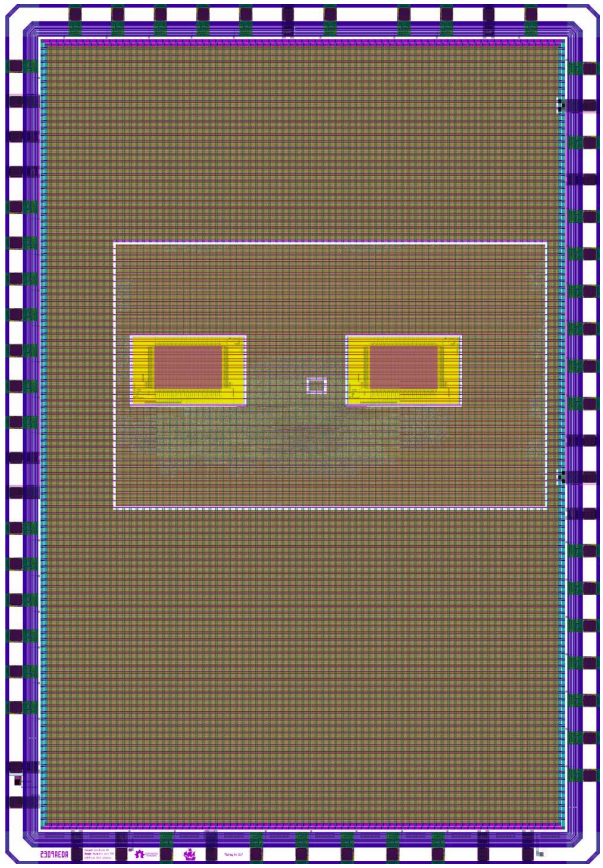

Figure 1.
Caravel Openframe harness die (3.2mm × 5.3mm)

Repository:

The complete Caravel openframe chip design may be obtained from the git repository located at https://github.com/efabless/caravel/. The project example demonstrating the use of the Openframe design for a RISC-V SoC core equivalent to that on Efabless MPW-one can be obtained from https://github.com/RTimothyEdwards/caravel_openframe_project/.

Process:

The efabless Caravel harness chip is fabricated in SkyWater 0.13µm CMOS technology, with process specifications and data at https://github.com/google/skywater−pdk/.

Version:

This document corresponds to version 1 of the Caravel openframe project example (September 2023). It is based heavily on the first Caravel harness design, from October 2020.

Revision history:

Documentation revision 0 (September 13, 2023)

Copied from the Caravel MPW-one design and modified for the Openframe project example.

Fabrication history:

The Caravel Openframe project example (PicoRV32 SoC) was submitted for fabrication on the Efabless ChipIgnite CI2309 (September 11, 2023).

Authorship:

The PicoRV32 core was designed by Claire Wolf of YosysHQ

The PicoSoC design was adapted from the example by Claire Wolf using the Wishbone bus architecture.

The original ASIC version of the PicoSoC was the Raven chip from Efabless (2019), designed by Tim Edwards and fabricated on X-Fab XH035.

The Caravel chip architecture is from Efabless (2021). The first Caravel-based PicoSoC was fabricated on the first Google-sponsored Open MPW run (MPW-one) (SkyWater sky130 process) and was designed by Tim Edwards and Mohamed Shalan and hardened using an early version of the Openlane flow, involving the entire Efabless Openlane development team.

The Caravel Openframe architecture was designed by Tim Edwards (2023).

The Caravel Openframe project example was designed by Tim Edwards (2023) and debugged and hardened by Mostafa Rady.

Pinout (64-pin QFN)

Openframe general pinout:

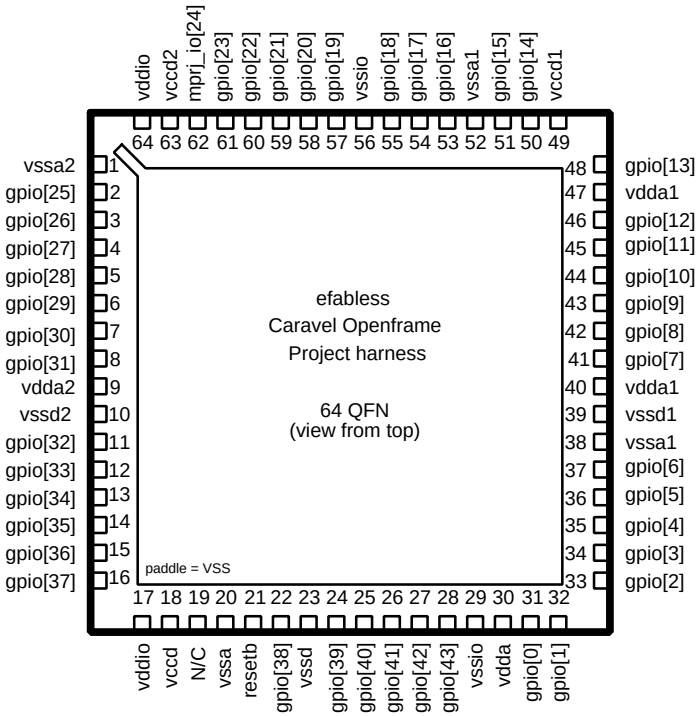

Figure 2. Caravel openframe (non-specific) pinout

Openframe project example pinout:

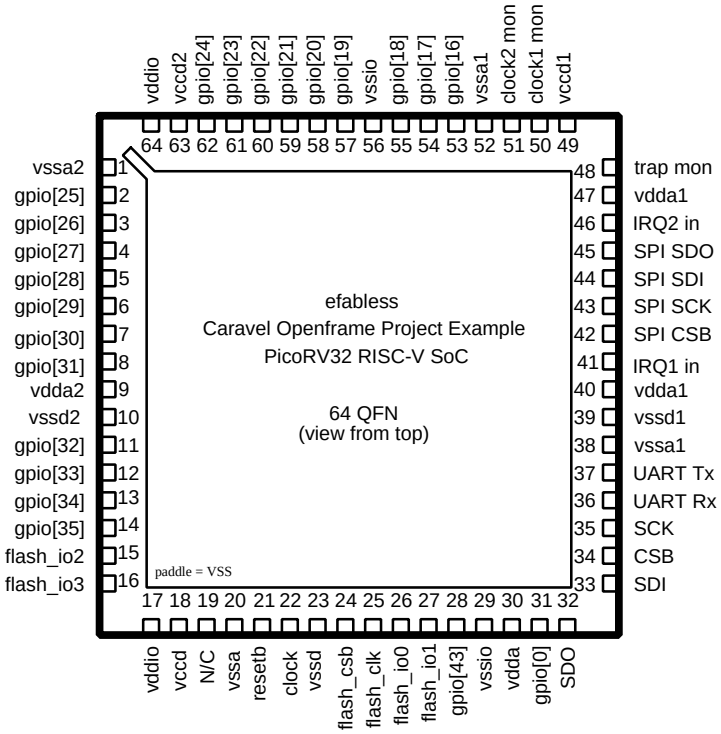

Figure 3. Pinout of the Caravel openframe project example (PicoRV32 SoC)

Pin Description (64-pin QFN)

Pin #	*Name*	*Type*	*Summary description*
28	gpio[43]	Digital I/O	General purpose configurable digital I/O with pullup/pulldown, input or output, enable/disable, analog output, high voltage output, slew rate control.
14, 13, 12, 11, 8, 7, 6, 5, 4, 3, 2, 62, 61, 60, 59, 58 57, 55, 54, 53	gpio[35:16]	Digital I/O	
31	gpio[0]	Digital I/O	
25	flash clk	Digital out	Flash SPI clock
24	flash csb	Digital out	Flash SPI chip select
16, 15, 27, 26	flash io3:0	Digital I/O	Flash SPI data input/output
22	clock	Digital in	External CMOS 3.3V clock source
21	resetb	Digital in	SoC system reset (sense inverted)
32	SDO	Digital out	Housekeeping serial interface data output
33	SDI	Digital in	Housekeeping serial interface data input
34	CSB	Digital in	Housekeeping serial interface chip select
35	SCK	Digital in	Housekeeping serial interface clock
37	ser_Tx	Digital out	UART transmit channel
36	ser_Rx	Digital in	UART receive channel
41	irq1	Digital in	External interrupt
46	irq2	Digital in	External interrupt
45	spi_sdo	Digital out	Serial interface master data output
43	spi_sck	Digital out	Serial interface master clock
42	spi_csb	Digital out	Serial interface master chip select
44	spi_sdi	Digital in	Serial interface master data input
48	trap mon	Digital out	CPU trap signal monitor
50	clock 1 mon	Digital out	CPU clock monitor
51	clock 2 mon	Digital out	Secondary clock monitor
17	vddio	3.3V Power	ESD and padframe power supply
30	vdda	3.3V Power	Analog power supply
49	vccd1	1.8V Power	Digital power supply
29, 56	vssio	Ground	ESD and padframe ground
20	vssa	Ground	Analog power ground
39	vssd1	Ground	Digital power ground
18	vccd	1.8V Power	(Unused supply domain)
63	vccd2	1.8V Power	(Unused supply domain)
23	vssd	Ground	(Unused supply domain)
10	vssd2	Ground	(Unused supply domain)
40, 47	vdda1	3.3V Power	(Unused supply domain)
31, 52	vssa1	Ground	(Unused supply domain)
9	vdda2	3.3V Power	(Unused supply domain)
1	vssa2	Ground	(Unused supply domain)

Standard package:	64-pin QFN
Package size:	9 mm × 9 mm
Pin pitch:	0.5 mm

General Purpose I/O GPIO (see pinout page 3)

The GPIO pins are assignable general-purpose digital input or output. Each of the 44 GPIO pins has an individually assignable configuration, even those with fixed functions such as the flash controller I/O. Each GPIO pin is individually addressed through the wishbone bus. A subset of the GPIO pins (those that do not have additional functions) can be accessed for input and output as a single vector (see pages 10 and 11).

GPIO memory address map:

C header name	*address*	*description*
reg_gpio_0_config	**0x21000000**	GPIO[0] configuration (see Table 2)
reg_gpio_1_config	**0x21000004**	GPIO[1] configuration (see Table 2)
reg_gpio_2_config	**0x21000008**	GPIO[2] configuration (see Table 2)
reg_gpio_3_config	**0x2100000c**	GPIO[3] configuration (see Table 2)
reg_gpio_4_config	**0x21000010**	GPIO[4] configuration (see Table 2)
reg_gpio_5_config	**0x21000014**	GPIO[5] configuration (see Table 2)
reg_gpio_6_config	**0x21000018**	GPIO[6] configuration (see Table 2)
reg_gpio_7_config	**0x2100001c**	GPIO[7] configuration (see Table 2)
reg_gpio_8_config	**0x21000020**	GPIO[8] configuration (see Table 2)
reg_gpio_9_config	**0x21000024**	GPIO[9] configuration (see Table 2)
reg_gpio_10_config	**0x21000028**	GPIO[10] configuration (see Table 2)
reg_gpio_11_config	**0x2100002c**	GPIO[11] configuration (see Table 2)
reg_gpio_12_config	**0x21000030**	GPIO[12] configuration (see Table 2)
reg_gpio_13_config	**0x21000034**	GPIO[13] configuration (see Table 2)
reg_gpio_14_config	**0x21000038**	GPIO[14] configuration (see Table 2)
reg_gpio_15_config	**0x2100003c**	GPIO[15] configuration (see Table 2)
reg_gpio_16_config	**0x21000040**	GPIO[16] configuration (see Table 2)
reg_gpio_17_config	**0x21000044**	GPIO[17] configuration (see Table 2)
reg_gpio_18_config	**0x21000048**	GPIO[18] configuration (see Table 2)
reg_gpio_19_config	**0x2100004c**	GPIO[19] configuration (see Table 2)
reg_gpio_20_config	**0x21000050**	GPIO[20] configuration (see Table 2)
reg_gpio_21_config	**0x21000054**	GPIO[21] configuration (see Table 2)
reg_gpio_22_config	**0x21000058**	GPIO[22] configuration (see Table 2)
reg_gpio_23_config	**0x2100005c**	GPIO[23] configuration (see Table 2)
reg_gpio_24_config	**0x21000060**	GPIO[24] configuration (see Table 2)
reg_gpio_25_config	**0x21000064**	GPIO[25] configuration (see Table 2)
reg_gpio_26_config	**0x21000068**	GPIO[26] configuration (see Table 2)
reg_gpio_27_config	**0x2100006c**	GPIO[27] configuration (see Table 2)
reg_gpio_28_config	**0x21000070**	GPIO[28] configuration (see Table 2)
reg_gpio_29_config	**0x21000074**	GPIO[29] configuration (see Table 2)
reg_gpio_30_config	**0x21000078**	GPIO[30] configuration (see Table 2)
reg_gpio_31_config	**0x2100007c**	GPIO[31] configuration (see Table 2)
reg_gpio_32_config	**0x21000080**	GPIO[32] configuration (see Table 2)
reg_gpio_33_config	**0x21000084**	GPIO[33] configuration (see Table 2)
reg_gpio_34_config	**0x21000088**	GPIO[34] configuration (see Table 2)
reg_gpio_35_config	**0x2100008c**	GPIO[35] configuration (see Table 2)
reg_gpio_36_config	**0x21000090**	GPIO[36] configuration (see Table 2)
reg_gpio_37_config	**0x21000094**	GPIO[37] configuration (see Table 2)
reg_gpio_38_config	**0x21000098**	GPIO[38] configuration (see Table 2)

General Purpose I/O GPIO (see pinout page 3)

GPIO memory address map:

C header name	*address*	*description*
reg_gpio_39_config	**0x2100009c**	GPIO[39] configuration (see Table 2)
reg_gpio_40_config	**0x210000a0**	GPIO[40] configuration (see Table 2)
reg_gpio_41_config	**0x210000a4**	GPIO[41] configuration (see Table 2)
reg_gpio_42_config	**0x210000a8**	GPIO[42] configuration (see Table 2)
reg_gpio_43_config	**0x210000ac**	GPIO[43] configuration (see Table 2)

GPIO description, continued.

In the memory-mapped register descriptions below, each register is shown as 32 bits corresponding to the data bus width of the wishbone bus. Addresses, however, are in bytes. Depending on the instruction and data type, the entire 32-bit register can be read in one instruction, or one 16-bit word, or one 8-bit byte.

Table 1 **reg_gpio_0_config**

0x21000003								**0x21000002**								**0x21000001**								**0x21000000**								address
(unused)																readback				GPIO configuration												value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Bits 2-0 are the digital mode

bit[2:0]	
000	Disabled mode (high impedence)
001	Input mode (input only)
010	Weak pull-up mode (requires output set to 1)
011	Weak pull-down mode (requires output set to 0)
100	Open drain high (requires external pull-up resistor)
101	Open drain low (requires external pull-down resistor)
110	Output mode
111	Output low-drive mode

Bit 3 is the IB mode select (see SkyWater I/O documentation)
Bit 4 is the input trip point voltage select (see SkyWater I/O documentation)

Bit 5 is the slow slew select
- 0 Fast slew
- 1 Slow slew

Bit 6 is the input disable override
- 0 Input disable controlled by the standard pin function
- 1 Overrides the input disable of the standard pin function with the value in bit 9

Bit 7 is the output disable override
- 0 Output disable controlled by the standard pin function
- 1 Overrides the output disable of the standard pin function with the value in bit 10

Bit 8 is the output value override
- 0 Output value controlled by the standard pin function
- 1 Overrides the output value of the standard pin function with the value in bit 11

Bit 9 is the input disable value when the override (bit 6) is set
- 0 Input enabled (when override is active)
- 1 Input disabled (when override is active)

Bit 10 is the output disable value when the override (bit 7) is set
- 0 Output enabled (when override is active)
- 1 Output disabled (when override is active)

Bit 11 is the output value when the override (bit 8) is set and the output is enabled
- 0 Output = 0
- 1 Output = 1

GPIO description, continued.

Table 1 (repeated) **reg_gpio_0_config**

0x21000003								**0x21000002**								**0x21000001**								**0x21000000**								address
(unused)																readback				GPIO configuration												value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

When reading from the register, bits 11 to 0 read back the configuration value saved in the register. Bits 15-12 have additional read-only values, defined as follows:

Bit 12 (read-only) is the input disable value at the pad. If the input disable is not overridden, then the value is the input disable state applied by the standard function for the GPIO.

0 Input is enabled
1 Input is disabled

Bit 13 (read-only) is the output disable value at the pad. If the output disable is not overridden, then the value is the output disable state applied by the standard function for the GPIO.

0 Output is enabled
1 Output is disabled

Bit 14 (read-only) is the output value being applied to the pad. If the output is disabled, or the output is used for the pull-up or pull-down function, this value may not correspond to the actual value at the GPIO pin.

0 Value zero applied to output
1 Value one applied to output

Bit 15 (read-only) is the value at the pad.

0 GPIO pin is at value 0
1 GPIO pin is at value 1

All GPIO pins have the same configuration bit fields:

Table 2 **reg_gpio_N_config**

0x21000003+N*4								**0x21000002+N*4**								**0x21000001+N*4**								**0x21000000+N*4**								address
(unused)																readback				GPIO configuration												value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Table is valid for **N** from **0** to **43**

GPIO description, continued.

GPIO Standard Functions

These are the "standard" functions that are active for each GPIO if the GPIO has not been configured to override input or output. Functions have been assigned to pins to make them compatible with the Caravel harness definition, so that this chip may be used with the Caravel development board without modifications.

GPIO[0]	GPIO vector
GPIO[1]	Housekeeping SPI SDO
GPIO[2]	Housekeeping SPI SDI
GPIO[3]	Housekeeping SPI CSB
GPIO[4]	Housekeeping SPI SCK
GPIO[5]	UART Rx
GPIO[6]	UART Tx
GPIO[7]	IRQ 1 input
GPIO[8]	SPI master CSB
GPIO[9]	SPI master SCK
GPIO[10]	SPI master SDI
GPIO[11]	SPI master SDO
GPIO[12]	IRQ 2 input
GPIO[13]	Trap signal monitor
GPIO[14]	Clock 1 monitor
GPIO[15]	Clock 2 monitor
GPIO[16–35]	GPIO vector
GPIO[36]	flash IO2
GPIO[37]	flash IO3
GPIO[38]	clock input
GPIO[39]	flash CSB
GPIO[40]	flash clock
GPIO[41]	flash IO0
GPIO[42]	flash IO1
GPIO[43]	GPIO vector

GPIO Vector Control

The base GPIO wishbone interface allows GPIO pins to be configured and controlled on an individual basis. However, that implies that to read or set the value of more than one GPIO requires multiple memory map accesses, preventing GPIO pins from being sampled or set at the same time.

To allow synchronizing of GPIO pins, a "vector control" method has been added. The vector functions are the "standard functions" of GPIOs 0, 16 to 35, and 43 (the function that is in effect for the GPIO if the input disable, output disable, or output value have not been overridden in the GPIO configuration).

Table 3 **`reg_gpio_vector_data`**

0x25000003								**0x25000002**								**0x25000001**								**0x25000000**								address
GPIO data																																value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Writing to this register updates up to 32 GPIO pins simultaneously, and reading from this register reads up to 32 GPIO pins simultaneously. 22 of the GPIOs are dedicated to vector control. The additional 10 pins are pins that have a standard function that is input only, and so can be used as outputs if the corresponding GPIO is reconfigured to enable the output driver (beware of problems that can be caused by enabling output on standard functions like the housekeeping SPI).

The GPIO pins correspond to bits in the vector data register by the following mapping. Note that in some cases, a different GPIO pin is mapped to a register read than is mapped to the register write.

Bit	GPIO # (read)	pin # (read)		GPIO # (write)	pin # (write)	
0–19	16–35	53–55, 57–62, 2–8, 11–14	gpio[16]– gpio[35]	16–35	53–55, 57–62, 2–8, 11–14	gpio[16]– gpio[35]
20	0	31	gpio[0]	0	31	gpio[0]
21	43	28	gpio[43]	43	28	gpio[43]
22	1	32	SDO	2	33	SDI
23	6	37	UART Tx	5	36	UART Rx
24	9	43	SPI SCK	4	35	SCK
25	8	42	SPI CSB	3	34	CSB
26	11	45	SPI SDO	10	44	SPI SDI
27	13	48	trap mon	7	41	IRQ 1
28	14	50	clock 1 mon	12	46	IRQ 2
29	15	51	clock 2 mon	38	24	clock
30	39	25	flash_csb	(none)	(none)	
31	40	26	flash_clk	(none)	(none)	

GPIO Vector Control *(cont.)*

For the 22 GPIO that have a standard function of the GPIO vector control, all output enables can be controlled simultaneously.

Table 4 **reg_gpio_vector_oeb**

0x25000007										**0x25000006**										**0x25000005**										**0x25000004**										address
(unused)										GPIO vector output disable																												value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							bit

The GPIO pins correspond to bits in the vector OEB register by the following mapping. All bits correspond to the same GPIO channel for both reads and writes. A bit value of 0 enables output on the corresponding GPIO channel, while a bit value of 1 disables the output.

Bit	GPIO # (read/write)	pin # (read/write)	pin name
0–19	16–35	53–55, 57–62, 2–8, 11–14	gpio[16]– gpio[35]
20	0	31	gpio[0]
21	43	28	gpio[43]

For the 22 GPIO that have a standard function of the GPIO vector control, all input enables can be controlled simultaneously.

Table 5 **reg_gpio_vector_ieb**

0x2500000b										**0x2500000a**										**0x25000009**										**0x25000008**										address
(unused)										GPIO vector input disable																												value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							bit

The GPIO pins correspond to bits in the vector IEB register by the following mapping. All bits correspond to the same GPIO channel for both reads and writes. A bit value of 0 enables input on the corresponding GPIO channel, while a bit value of 1 disables the input.

Bit	GPIO # (read/write)	pin # (read/write)	pin name
0–19	16–35	53–55, 57–62, 2–8, 11–14	gpio[16]– gpio[35]
20	0	31	gpio[0]
21	43	28	gpio[43]

Housekeeping SPI SDI (pin 33), CSB (pin 34), SCK (pin 35), and SDO (pin 32)

The “housekeeping” SPI is an SPI slave that can be accessed from a remote host through a standard 4-pin serial interface. The SPI implementation is mode 0, with new data on SDI captured on the SCK rising edge, and output data presented on the falling edge of SCK (to be sampled on the next SCK rising edge). The SPI pins are shared with user area general-purpose I/O.

SPI protocol definition

All input is in groups of 8 bits. Each byte is input msb first.

Every command sequence requires one command word (8 bits) followed by one address word (8 bits) followed by one or more data words (8 bits each), according to the data transfer modes defined below.

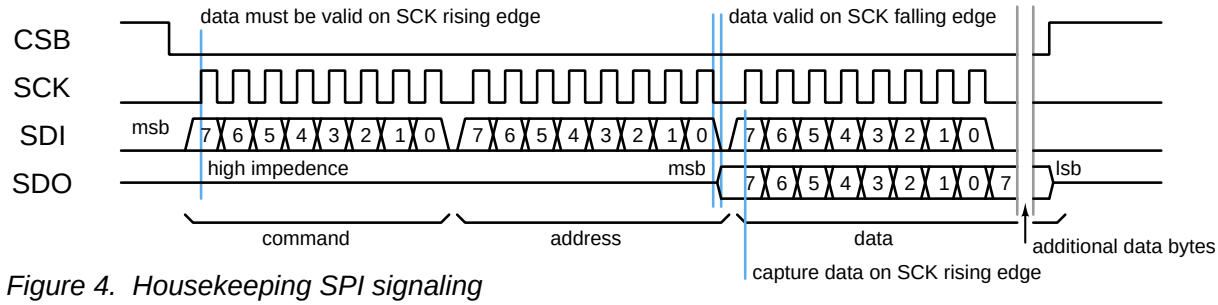

Figure 4. Housekeeping SPI signaling

Addresses are read in sequence from lower values to higher values.

Therefore groups of bits larger than 8 should be grouped such that the lowest bits are at the highest address. Any bits additional to an 8-bit boundary should be at the lowest address.

Data are captured from the register map in bytes on the falling edge of the last SCK before a data byte transfer. Multi-byte transfers should ensure that data do not change between byte reads.

CSB pin must be low to enable an SPI transmission. Data are clocked by pin SCK, with data valid on the rising edge of SCK. Output data are received on the SDO line. SDO is held high-impedance when CSB is high and at all times other than the transfer of data bits on a read command. SDO outputs become active on the falling edge of SCK, such that data are written and read on the same SCK rising edge.

After CSB is set low, the SPI is always in the "command" state, awaiting a new command.

The first transferred byte is the command word, interpreted according to Table 6 below.

Table 6 Housekeeping SPI command word definition

00000000	No operation
10000000	Write in streaming mode
01000000	Read in streaming mode
11000000	Simultaneous Read/Write in streaming mode
11000100	Pass-through (management) Read/Write in streaming mode
11000110	Pass-through (user) Read/Write in streaming mode
10nnn000	Write in n-byte mode (up to 7 bytes).
01nnn000	Read in n-byte mode (up to 7 bytes).
11nnn000	Simultaneous Read/Write in n-byte mode (up to 7 bytes).

All other words are reserved and act as no-operation if not defined by the SPI slave module.

SPI protocol definition *(continued)*

The two basic modes of operation are "streaming mode" and "n-byte mode". In "streaming mode" operation, data are sent or received continuously, one byte at a time, with the internal address incrementing for each byte. Streaming mode operation continues until CSB is raised to end the transfer.

In "n-byte mode" operation, the number of bytes to be read and/or written is encoded in the command word, and may have a value from 1 to 7 (note that a value of zero implies streaming mode). After n bytes have been read and/or written, the SPI returns to waiting for the next command. No toggling of CSB is required to end the command or to initiate the following command.

Pass-thru mode

The pass-thru mode puts the CPU into immediate reset, then sets FLASH_CSB low to initiate a data transfer to the QSPI flash. After the pass-thru command byte has been issued, all subsequent SPI signaling on SDI and SCK are applied directly to the QSPI flash (pins FLASH_IO0 and FLASH_CLK, respectively), and the QSPI flash data output (pin FLASH_IO1) is applied directly to SDO, until the CSB pin is raised. When CSB is raised, the FLASH_CSB is also raised, terminating the data transfer to the QSPI flash. The CPU is brought out of reset, and starts executing instructions at the program start address.

This mode allows the QSPI flash to be programmed from the same SPI communication channel as the housekeeping SPI, without the need for additional wiring to the QSPI flash chip.

There are two pass-thru modes. The first one corresponds to the primary SPI flash used by the management SoC. The second one corresponds to a secondary optional SPI flash that can be defined in the user project. The pass-thru mode allows a communications chip external to the Caravel chip program either SPI flash chip from a host computer without requiring separate external access to the SPI flash. Both pass-thru modes only connect to I/O pins 0 and 1 of the SPI flash chips, and so must operate only in the 4-pin SPI mode. The user project may elect to operate the SPI flash in quad mode using a 6-pin interface.

Housekeeping SPI registers

The purpose of the housekeeping SPI is to give access to certain system values and controls independently of the CPU. The housekeeping SPI can be accessed even when the CPU is in full reset. Some control registers in the housekeeping SPI affect the behavior of the CPU in a way that potentially can be detrimental to the CPU operation, such as adjusting the trim value of the digital frequency-locked loop generating the CPU core clock.

Under normal working conditions, the SPI should not need to be accessed unless it is to adjust the clock speed of the CPU. All other functions are purely for test and debug.

The housekeeping SPI can be accessed by the CPU from a running program by enabling the SPI master, and enabling the bit that connects the internal SPI master directly to the housekeeping SPI. This configuration then allows a program to read, for example, the user project ID of the chip. See the SPI master description for details.

manufacturer_ID register address 0x01 low 4 bits and register address 0x02
The 12-bit manufacturer ID for efabless is 0x456

product_ID register address 0x03
The product ID for the Caravel openframe chip is 0x14

Housekeeping SPI registers *(continued)*

user project ID register addresses 0x04 to 0x07

The 4-byte (32 bit) user project ID is metal-mask programmed on each project before tapeout, with a unique number given to each user project.

DLL enable register address 0x08 bit 0

This bit enables the digital frequency-locked-loop clock multiplier. The enable should be applied prior to turning off the DLL bypass to allow the DLL time to stabilize before using it to drive the CPU clock.

DLL DCO enable register address 0x08 bit 1

The DLL can be run in DCO mode, in which the feedback loop to the driving clock is removed, and the system operates in free-running mode, driven by the ring oscillator which can be tuned between approximately 70 to 150 MHz by setting the trim bits (see below).

DLL bypass register address 0x09 bit 0

When enabled, the DLL bypass switches the clock source of the CPU from the DLL output to the external CMOS clock (pin 22). The default value is 0x1 (CPU clock source is the external CMOS clock).

CPU IRQ register address 0x0A bit 0

This is a dedicated manual interrupt driving the CPU IRQ channel 6. The bit is not self-resetting, so while the rising edge will trigger an interrupt, the signal must be manually set to zero before it can trigger another interrupt.

CPU reset register address 0x0B bit 0

The CPU reset bit puts the entire CPU into a reset state. This bit is not self-resetting and must be set back to zero manually to clear the reset state.

CPU trap register address 0x0C bit 0

If the CPU has stopped after encountering an error, it will raise the trap signal. The trap signal can be configured to be read from a GPIO pin, but as the GPIO state is potentially unknowable, the housekeeping SPI can be used to determine the true trap state.

DLL trim register addresses 0x0D (all bits) to 0x10 (lower 2 bits)

The 26-bit trim value can adjust the DCO frequency over a factor of about two from the slowest (trim value 0x3ffffff) to the fastest (trim value 0x0). Default value is 0x3ffefff (slow trim, -1). Note that this is a thermometer-code trim, where each bit provides an additional (approximately) 250 ps delay (on top of a fixed delay of 4.67 ns). The fastest output frequency is approximately 150 MHz while the slowest output frequency is approximately 70 MHz.

DLL output divider register address 0x11 bits 2–0

The DLL output can be divided down by an integer divider to provide the core clock frequency. This 3-bit divider can generate a clock divided by 2 to 7. Values 0 and 1 both pass the undivided DLL clock directly to the core (and should not be used, as the processor does not operate at these frequencies).

DLL output divider (2) register address 0x11 bit 5–3

The DLL 90-degree phase output is passed through an independent 3-bit integer clock divider and provided to the user project space as a secondary clock. Values 0 and 1 both pass the undivided DLL clock, while values 2 to 7 pass the clock divided by 2 to 7, respectively.

Housekeeping SPI registers *(continued)*

DLL feedback divider register address 0x12 bits 4–0

The DLL operates by comparing the input clock (pin 22) rate to the rate of the DLL clock divided by the feedback divider value (when running in DLL mode, not DCO mode). The feedback divider must be set such that the external clock rate multiplied by the feedback divider value falls between 70 and 150 MHz (preferably centered on this range, or approximately 110 MHz). For example, when using a 10 MHz external clock, the divider should be set to 11 (11 * 10 = 110). The DCO range and the number of bits of the feedback divider implies that the external clock should be no slower than around 4 to 5 MHz.

Core clock monitor divider register address 0x13 bits 7–0

The contents of this register are an 8-bit value by which the core clock is divided before being output on the monitor pin (pin 50). By default, the clock is divided by 100, so for the default 10 MHz clock on the standard Caravel development board, the clock monitor will toggle at 100 kHz.

Auxiliary clock monitor divider register address 0x13 bits 7–0

The contents of this register are an 8-bit value by which the secondary clock is divided before being output on the monitor pin (pin 51). By default, the clock is divided by 100. Because the secondary clock can exceed the rating of the GPIO, dividing down the clock is necessary for clock rates higher than approximately 50 MHz.

Housekeeping SPI registers *(continued)*

Table 7 **Housekeeping SPI register map**

Register Address	*msb* 7	6	5	4	3	2	1	*lsb* 0	*comments*
0x00	SPI status and control								unused/ undefined
0x01	unused				manufacturer_ID[11:8] (= 0x4)				read-only
0x02	manufacturer_ID[7:0] (= 0x56)								read-only
0x03	product_ID (= 0x14)								read-only
0x04–0x07	user_project_ID (unique value per project)								read-only
0x08	unused						DLL DCO enable	DLL enable	default 0x02
0x09	unused							DLL bypass	default 0x01
0x0A	unused							CPU IRQ	default 0x00
0x0B	unused							CPU reset	default 0x00
0x0C	unused							CPU trap	read-only
0x0D–0x10	DCO trim (26 bits) (= 0x3ffefff)								default 0x3ffefff
0x11	unused		DLL output divider 2			DLL output divider			default 0x12
0x12	unused			DLL feedback divider					default 0x04
0x13	Core clock monitor (clock 1) divider								default 0x64
0x14	Auxiliary clock monitor (clock 2) divider								default 0x64

QSPI Flash interface flash io0–1 (pins 26 to 27), flash csb (pin 24), and flash clk (pin 25)

The QSPI flash controller is automatically enabled on power-up, and will immediately initiate a read sequence in single-bit mode with pin "flash io0" acting as SDI (data from flash to CPU) and pin "flash io1" acting as SDO (data from CPU to flash). Protocol is according to, *e.g.,* Cypress S25FL256L.

The initial SPI instruction sequence is as follows:

0xFF	Mode bit reset
0xAB	Release from deep power-down
0x03	Read w/3 byte address
0x00	Program start address (**0x10000000**) (3 bytes) (upper byte is ignored)
0x00	
0x00	

The QSPI flash continues to read bytes, either sequentially on the same command, or issuing a new read command to read from a new address.

The behavior of the QSPI flash controller can be modified by changing values in the register below:

Table 8 **reg_spictrl**

0x2d000003								**0x2d000002**								**0x2d000001**								**0x2d000000**								address
	(unused)							(see below)								(unused)										(see below)								value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

mask bit	default	description
31	1	QSPI flash interface enable
22–20	0	Access mode (see table below)
19–16	8	Dummy clock cycle count
11–8	0	Bit-bang OE FLASH_IO3–FLASH_IO0
5	0	Bit-bang FLASH_CSB
4	0	Bit-bang FLASH_CLK
3–0	0	Bit-bang value FLASH_IO3–FLASH_IO0

Access mode bit selection (bits 22–20):

0	**000**	Single bit per clock
1	**001**	Single bit per clock (same as 0)

All additional modes (QSPI dual and quad modes) cannot be used, as the management SoC only has pins for data lines 0 and 1.

The SPI flash can be accessed by bit banging when the enable is off. To do this from the CPU, the entire routine to access the SPI flash must be read into SRAM and executed from the SRAM.

Interrupts IRQ 1 (pin 41) and IRQ 2 (pin 46)

The interrupt pin IRQ 1 triggers the CPU interrupt channel 7, and pin IRQ 2 triggers the CPU interrupt channel 8. IRQs can be handled by special CPU instructions **picorv32_maskirq_insn()** and **picorv32_retirq_insn()**.

External clock clock (pin 22)

The external clock functions as the source clock for the entire processor. On start-up, the processor runs at the same rate as the external clock. The processor program may access the housekeeping SPI to set the processor into DLL mode or DCO free-running mode. In DLL mode, the external clock is multiplied up by the feedback divider value to obtain the core clock. In DCO mode, the processor is driven by a trimmed free-running ring oscillator.

UART ser Tx (pin 37) and ser Rx (pin 36)

The UART is a standard 2-pin serial interface that can communicate with most similar interfaces at a fixed baud rate. Although the UART operates independently of the CPU, data transfers are blocking operations which will generate CPU wait states until the data transfer is completed.

The behavior of the UART can be modified by changing values in the registers below:

Table 9 **reg_uart_clkdiv**

0x20000003								**0x20000002**								**0x20000001**								**0x20000000**								address
UART clock divider																																value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The entire 32 bit word encodes the number of CPU core cycles to divide down to get the UART data bit rate (baud rate). The default value is 1.

Example: If the external crystal is 10MHz, then the core CPU clock runs at 10MHz.
To get 9600 baud, 10E6 / 9600 = 1042 (hex value 0x0412).

Table 10 **reg_uart_data**

0x20000007								0x20000006								0x20000005								**0x20000004**								address
(unused, value is 0x0)																																value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Writing a value to this register will immediately start a data transfer on the SER_TX pin. If a UART write operation is pending, then the CPU will be blocked with wait states until the transfer is complete before starting the new write operation. This makes the UART transmit a relatively expensive operation on the CPU, but avoids the necessity of buffering data and checking for buffer overflow. Reading a value from this register returns 255 (0xff) if no valid data byte is in the receive buffer, and returns the value of the receive buffer otherwise, and clears the receive buffer for additional reads. Note that there is no FIFO associated with the UART.

Table 11 **reg_uart_enable**

0x2000000b								0x2000000a								0x20000009								**0x20000008**								address
(unused, value is 0x0)																																value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The UART must be enabled to run (default disabled)

SPI Master SPI SDI (pin 44), SPI CSB (pin 42), SPI SCK (pin 43), and SPI SDO (pin 45)

Table 12 **reg_spi_config**

0x24000003								0x24000002								0x24000001								0x24000000								address
(undefined, reads zero)																SPI master configuration																value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Configuration bit defintions

Bit 15	Housekeeping	0 = SPI master connected to external pins 1 = SPI master connected directly to housekeeping SPI
Bit 14	SPI interrupt enable	0 = interrupt disabled 1 = interrupt enabled
Bit 13	SPI system enable	0 = SPI disabled 1 = SPI enabled
Bit 12	stream	0 = apply/release CSB separately for each byte 1 = apply CSB until stream bit is cleared (manually)
Bit 11	mode	0 = read and change data on opposite SCK edges 1 = read and change data on the same SCK edge
Bit 10	invert SCK	0 = normal SCK 1= inverted SCK
Bit 9	invert CSB	0 = normal CSB (low is active) 1 = inverted CSB (high is active)
Bit 8	MLB	0 = msb first 1 = lsb first
Bits 7–0	prescaler	count (in master clock cycles) of 1/2 SCK cycle (default value 2)

All configuration bits other than the prescaler default to value zero.

Table 13 **reg_spi_data**

0x24000007								0x24000006								0x24000005								0x24000004								address
(undefined, reads zero)																								SPI data								value
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The byte at 0x24000004 holds the SPI data (either read or write)

Reading to and writing from the SPI master is simply a matter of setting the required values in the configuration register, and writing values to or reading from reg_spi_data. The protocol is similar to the UART. A write operation will stall the CPU if an incomplete SPI transmission is still in progress. Reading from the SPI will also stall the CPU if an incomplete SPI transmission is still in progress. There is no FIFO buffer for data. Therefore SPI reads and writes are relatively expensive operations that tie up the CPU, but will not lose or overwrite data. Note that there is no FIFO associated with the SPI master.

Counter-Timer 0

The counter/timer is a general-purpose 32-bit adder and subtractor that can be configured for a variety of timing functions including one-shot counts, continuous timing, and interval interrupts. At a core clock rate of 80MHz, the longest single time interval is 26.84 seconds.

Table 14 **reg_timer0_config**

0x22000003	0x22000002	0x22000001	0x22000000	address
(undefined, reads zero)			Timer config	value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Timer configuration bit definitions

Bit 3	Counter/timer enable	1 = counter/timer enabled 0 = counter/timer disabled
Bit 2	Oneshot mode	1 = oneshot mode 0 = continuous mode
Bit 1	Updown	1 = count up 0 = count down
Bit 0	Interrupt enable	1 = interrupt enabled 0 = interrupt disabled

Table 15 **reg_timer0_value**

0x22000007	0x22000006	0x22000005	0x22000004	address
Timer value				value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The value in this register is the current value of the counter. Value is 32 bits. The register is read-write and can be used to reset the timer.

Table 16 **reg_timer0_data**

0x2200000b	0x2200000a	0x22000009	0x22000008	address
Timer data				value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The value in this register is the reset value for the comparator.

When enabled, the counter counts up or down from the value set in reg_timer_value at the time the counter is enabled. If counting up, the count continues until the counter reaches reg_timer_data. If counting down, the count continues until the counter reaches zero.

In continuous mode, the counter resets to zero if counting up, and resets to the value in reg_timer_data if counting down, and the count continues immediately. If the interrupt is enabled, the counter will generate an interrupt on every cycle.

In one-shot mode, the counter triggers an interrupt (IRQ channel 10; see page 22) when it reaches the value of reg_timer_data (up count) or zero (down count), and stops.

Note: When the counter/timer is disabled, the reg_timer_value remains unchanged, which puts the timer in a hold state. When re-enabled, counting resumes. To reset the timer, write zero to the reg_timer_value register.

Counter-Timer 1

The second counter/timer is functionally identical to the first, with different memory mapped addresses for the controls, as shown in the tables below.

Table 17 **reg_timer1_config**

0x23000003	0x23000002	0x23000001	0x23000000	address
(undefined, reads zero)			Timer config	value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

Timer configuration bit definitions

Bit 3	Counter/timer enable	1 = counter/timer enabled 0 = counter/timer disabled
Bit 2	Oneshot mode	1 = oneshot mode 0 = continuous mode
Bit 1	Updown	1 = count up 0 = count down
Bit 0	Interrupt enable	1 = interrupt enabled 0 = interrupt disabled

Table 18 **reg_timer1_value**

0x23000007	0x23000006	0x23000005	0x23000004	address
Timer value				value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The value in this register is the current value of the counter. Value is 32 bits. The register is read-write and can be used to reset the timer.

Table 19 **reg_timer1_data**

0x2300000b	0x2300000a	0x23000009	0x23000008	address
Timer data				value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	bit

The value in this register is the reset value for the comparator.

When enabled, the counter counts up or down from the value set in reg_timer_value at the time the counter is enabled. If counting up, the count continues until the counter reaches reg_timer_data. If counting down, the count continues until the counter reaches zero.

In continuous mode, the counter resets to zero if counting up, and resets to the value in reg_timer_data if counting down, and the count continues immediately. If the interrupt is enabled, the counter will generate an interrupt on every cycle.

In one-shot mode, the counter triggers an interrupt (IRQ channel 11; see next page) when it reaches the value of reg_timer_data (up count) or zero (down count), and stops.

Note: When the counter/timer is disabled, the reg_timer_value remains unchanged, which puts the timer in a hold state. When re-enabled, counting resumes. To reset the timer, write zero to the reg_timer_value register.

CPU Counter

The PicoRV32 CPU defines an additional 32-bit counter that can be enabled using the special CPU instruction **picorv32_timer_insn()**.

Interrupts (IRQ)

The interrupt vector is set to memory addres 0 (bottom of SRAM). The program counter switches to this location when an interrupt is received. To enable interrupts, it is necessary to copy an interrupt handler to memory location 0. The PicoRV32 defines 32 IRQ channels, of which the Caravel chip uses only a handful, as described in the table below. All IRQ channels not in the list below always have value zero.

Table 20 CPU IRQ channel definitions

IRQ channel	*description*
4	UART data available
6	Housekeeping SPI IRQ
7	IRQ 1 external pin (pin 41)
8	IRQ 2 external pin (pin 46)
9	SPI master data available, when enabled (see Table 12)
10	Timer 0 expired, when enabled (see Table 14)
11	Timer 1 expired, when enabled (see Table 17)

The Caravel PicoRV32 implementation does not enable IRQ QREGS (see PicoRV32 description).

The handling of interrupts is beyond the scope of this document (see RISC-V instruction set description, and the PicoRV32 instruction set extensions for interrupts). All interrupts are masked and must be enabled in software.

SRAM

The Caravel chip has an on-board memory of 4kB (1024 words of width 32 bits each). The memory is located starting at address 0 (zero).

Caravel Openframe project example PicoRV32 SoC simplified block diagram

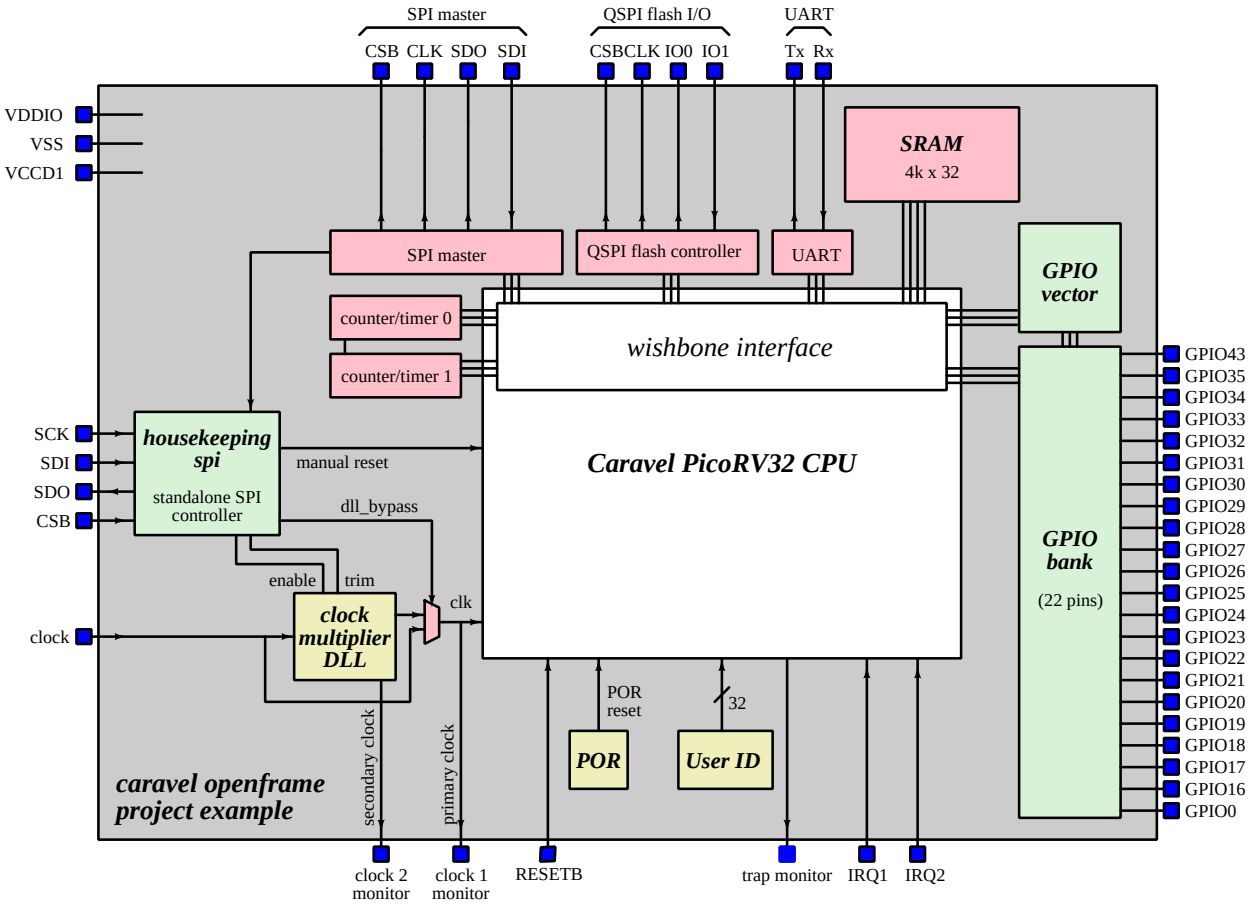

Figure 6. Openframe project example block diagram

Programming

The RISC-V architecture has a **gcc** compiler. The best reference for getting the correct cross-compiler version is the PicoRV32 source at

`https://github.com/YosysHQ/picorv32`.

Specifically, see the top-level **`README.md`** file section “Building a pure RV32I Toolchain.”

For programming examples specifically for the Caravel chip (assuming a correct installation of a RISC-V gcc toolchain as described above), see

`https://github.com/RTimothyEdwards/caravel_openframe_project`

The directory **`verilog/dv`** contains example source code to program the Caravel chip along with the header file **`defs.h`** that defines the memory-mapped locations as described throughout this text.

The **`verilog/dv`** directory contains a **`Makefile`** that compiles hex files and runs simulations of test programs that exercise various features of the chip. Currently most simulations are done in the **cocotb** environment, but one testbench **`gpio_vector`** provides a simple C-code and verilog testbench example.

For documentation on the provided demonstration circuit board and driver software, see the repository **`https://efabless/caravel_board`**

Additional references

See **`https://riscv.org/`**
`https://riscv.org/software-status/`

Memory Mapped I/O summary by address

Address (bytes)	Function		
0x00 00 00 00	Flash SPI / overlaid SRAM (4k words) start of memory block		
0x00 00 0f ff	End of SRAM		
0x10 00 00 00	Flash SPI start of program block		Program to run starts here on reset.
0x10 ff ff ff	Maximum SPI flash addressable space (16MB) with QSPI 3-byte addressing		
0x1f ff ff ff	Maximum SPI flash addressable space (32MB)		
0x20 00 00 00	UART clock divider select (system clock freq. / baud rate)		
0x20 00 00 04	UART data (returns `0xffffffff` if receiver buffer is empty)		
0x20 00 00 08	UART enable		
0x21 00 00 00	GPIO[0] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 04	GPIO[1] configuration	SDO	default `0x006` (bidirectional)
0x21 00 00 08	GPIO[2] configuration	SDI	default `0x4c1` (input)
0x21 00 00 0c	GPIO[3] configuration	CSB	default `0x4c1` (input)
0x21 00 00 10	GPIO[4] configuration	SCK	default `0x4c1` (input)
0x21 00 00 14	GPIO[5] configuration	UART Rx	default `0x4c1` (input)
0x21 00 00 18	GPIO[6] configuration	UART Tx	default `0x2c6` (output)
0x21 00 00 1c	GPIO[7] configuration	IRQ 1	default `0x1c3` (input, pulldown)
0x21 00 00 20	GPIO[8] configuration	SPI CSB	default `0x2c6` (output)
0x21 00 00 24	GPIO[9] configuration	SPI SCK	default `0x2c6` (output)
0x21 00 00 28	GPIO[10] configuration	SPI SDI	default `0x4c1` (input)
0x21 00 00 2c	GPIO[11] configuration	SPI SDO	default `0x006` (bidirectional)
0x21 00 00 30	GPIO[12] configuration	IRQ 2	default `0x1c3` (input, pulldown)
0x21 00 00 34	GPIO[13] configuration	trap monitor	default `0x2c6` (output)
0x21 00 00 38	GPIO[14] configuration	clock 1 monitor	default `0x2c6` (output)
0x21 00 00 3c	GPIO[15] configuration	clock 2 monitor	default `0x2c6` (output)
0x21 00 00 40	GPIO[16] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 44	GPIO[17] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 48	GPIO[18] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 4c	GPIO[19] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 50	GPIO[20] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 54	GPIO[21] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 58	GPIO[22] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 5c	GPIO[23] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 60	GPIO[24] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 64	GPIO[25] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 68	GPIO[26] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 6c	GPIO[27] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 70	GPIO[28] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 74	GPIO[29] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 78	GPIO[30] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 7c	GPIO[31] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 80	GPIO[32] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 84	GPIO[33] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 88	GPIO[34] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 8c	GPIO[35] configuration	general-purpose	default `0x4c1` (input)
0x21 00 00 90	GPIO[36] configuration	flash IO2	default `0x006` (bidirectional)
0x21 00 00 94	GPIO[37] configuration	flash IO3	default `0x006` (bidirectional)
0x21 00 00 98	GPIO[38] configuration	clock	default `0x4c1` (input)
0x21 00 00 9c	GPIO[39] configuration	flash CSB	default `0x2c6` (output)
0x21 00 00 a0	GPIO[40] configuration	flash clock	default `0x2c6` (output)
0x21 00 00 a4	GPIO[41] configuration	flash IO0	default `0x006` (bidirectional)
0x21 00 00 a8	GPIO[42] configuration	flash IO1	default `0x006` (bidirectional)
0x21 00 00 ac	GPIO[43] configuration	general-purpose	default `0x4c1` (input)

Memory Mapped I/O summary by address *(continued)*

Address (bytes)	Function
	bits 0–2 = digital mode (see below) bit 3 = IB mode select (0 = , 1 =) bit 4 = input voltage trip point select (0 = , 1 =) bit 5 = slow slew (0 = fast slew, 1 = slow slew) (default 0) bit 6 = override input enable (0 = standard function, 1 = override) bit 7 = override output enable (0 = standard function, 1 = override) bit 8 = override output value (0 = standard function, 1 = override) bit 9 = input enable value when overridden (0 = enable, 1 = disable) bit 10 = output enable value when overridden (0 = enable, 1 = disable) bit 11 = output value when overridden (0 = low, 1 = high) Digital mode bits — Digital mode description bit 2 1 0 0 0 0 — Disabled 0 0 1 — Input only 0 1 0 — Pull-up 0 1 1 — Pull-down 1 0 0 — Open drain high 1 0 1 — Open drain low 1 1 0 — Output 1 1 1 — Weak drive output
0x22 00 00 00	Counter/Timer 0 configuration register (lower 4 bits) bit 0 = enable (0 = hold, 1 = count) bit 1 = oneshot (0 = continuous count, 1 = one-shot count) bit 2 = updown (0 = count down, 1 = count up) bit 3 = irq enable (0 = disabled, 1 = trigger IRQ channel 10 on timeout)
0x22 00 00 04	Counter/Timer 0 current value Set or read the 32-bit current value.
0x22 00 00 08	Counter/Timer 0 reset value Set or read the 32-bit reset (down-count) or compare (up-count) value.
0x23 00 00 00	Counter/Timer 1 configuration register (lower 4 bits) bit 0 = enable (0 = hold, 1 = count) bit 1 = oneshot (0 = continuous count, 1 = one-shot count) bit 2 = updown (0 = count down, 1 = count up) bit 3 = irq enable (0 = disabled, 1 = trigger IRQ channel 11 on timeout)
0x23 00 00 04	Counter/Timer 1 current value Set or read the 32-bit current value.
0x23 00 00 08	Counter/Timer 1 reset value Set or read the 32-bit reset (down-count) or compare (up-count) value.
0x24 00 00 00	SPI master configuration register bits 0–7 = prescaler (core clock / (prescaler + 1) = SPI clock rate / 2) (default 2) bit 8 = mlb (0 = msb first, 1 = lsb first) (default 0) bit 9 = invcsb (0 = csb active low, 1 = csb active high) (default 0) bit 10 = invsck (0 = normal sck, 1 = inverted sck) (default 0) bit 11 = mode (0 = read/write on opposite sck edge, 1 = same edge) (default 0) bit 12 = stream (0 = raise csb after each byte, 1 = keep csb low until stream bit cleared) bit 13 = enable (0 = SPI master disabled, 1 = SPI master enabled) bit 14 = irq enable (0 = disabled, 1 = SPI read valid triggers interrupt channel 9) bit 15 = housekeeping (0 = disconnected, 1 = connected)

Memory Mapped I/O summary by address *(continued)*

Address (bytes)	Function
0x24 00 00 04	SPI master data register (low 8 bits) Write data to send to low byte or read received data from low byte.
0x25 00 00 00	GPIO vector data in/out (32 bits) 0 = low 1 = high
0x25 00 00 04	GPIO vector output disable (22 bits) 0 = enabled 1 = disabled
0x25 00 00 08	GPIO vector input disable (22 bits) 0 = enabled 1 = disabled
0x2d 00 00 00	QSPI controller config bit 31 MEMIO enable (reset = 1) 0 = bit-bang mode bit 22 DDR enable bit 21 QSPI enable bit 20 CRM enable bits 19-16 Read latency cycles bits 11-8 I/O output enable bits (bit bang mode) bit 5 Chip select line (bit bang mode) bit 4 Serial clock line (bit bang mode) bits 3-0 Data bits (bit bang mode)
0x41 00 00 00	Debug registers (reserved space, unused)

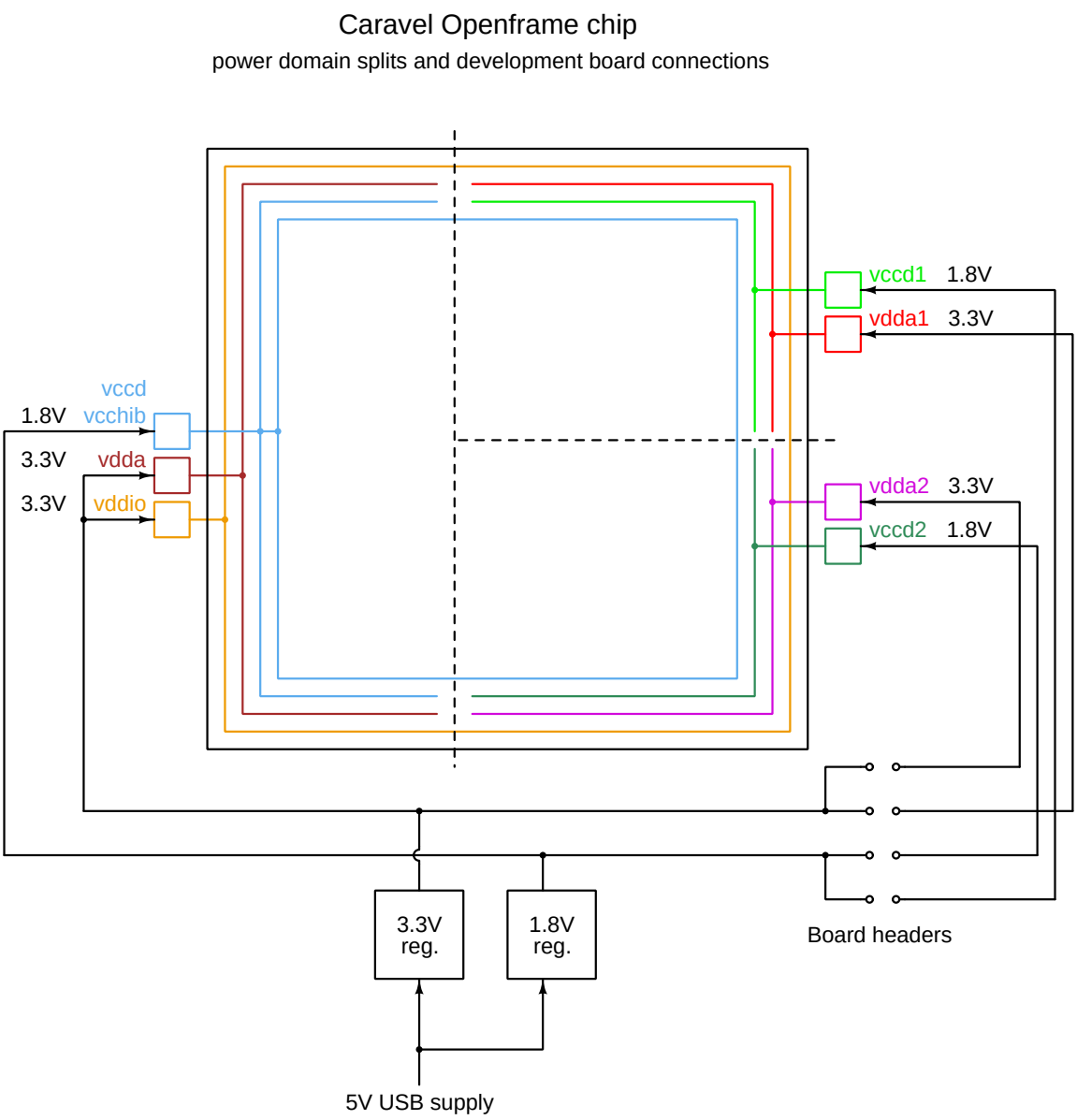

Figure 7. Openframe chip power domain splits and connections to development board.

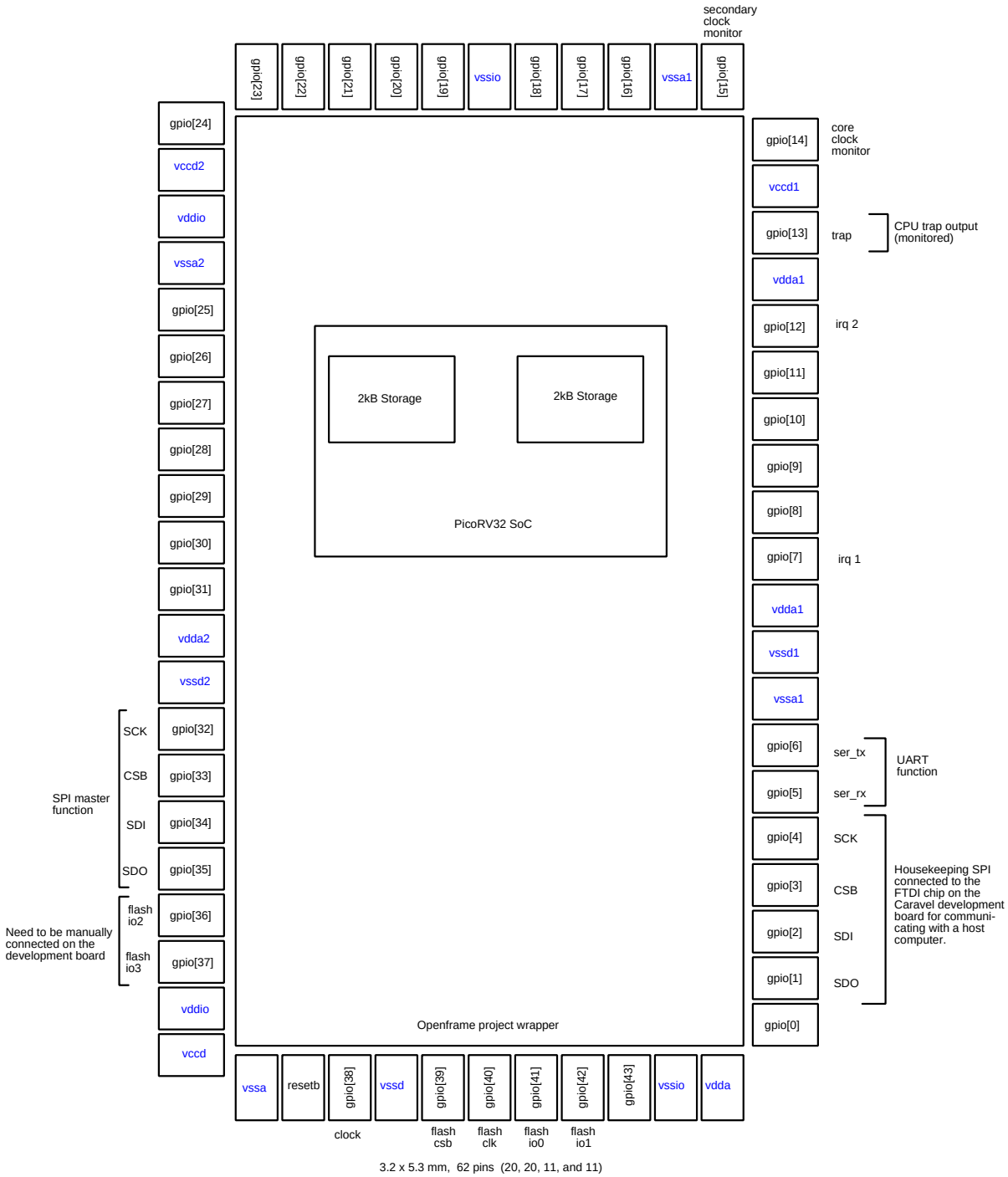

Figure 8. Basic floorplan of the Caravel Openframe project example (PicoRV32 SoC)

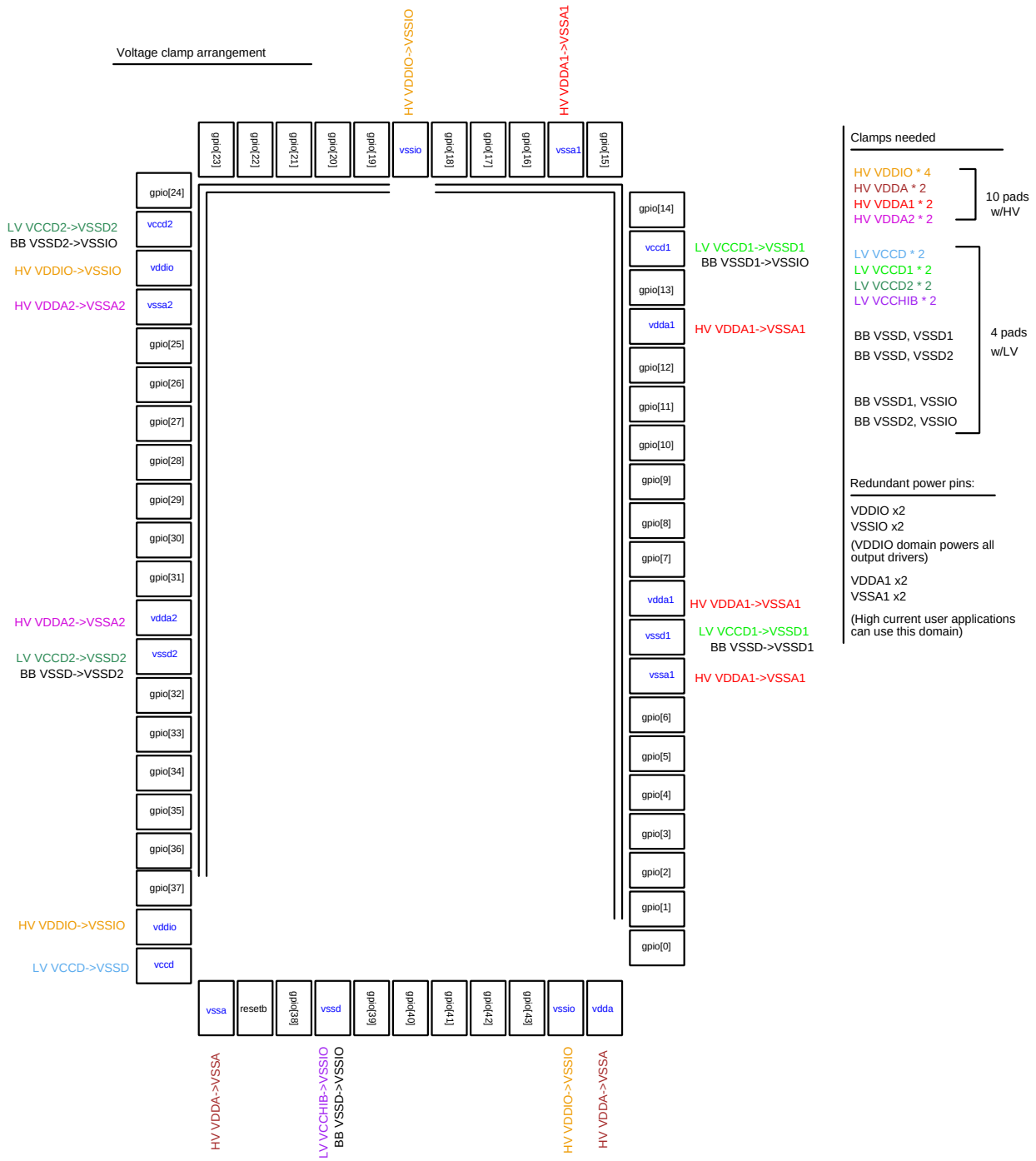

Figure 9. Voltage clamp locations and connections on the Caravel Openframe padframe

	minimum	*typical*	*maximum*	*units*
Supply voltage (VDDIO):	1.8	3.3	5.0	V
Core digital supply voltage (VCCD):	1.62	1.8	1.98	V
Junction temperature:	–40	27	100	°C
V_{OH}	0.8 ·VDDIO			V
V_{OL}			0.4	V
Management area power		TBD		mW
Storage area power		TBD		mW

Efabless "caravel" openframe project harness 64-pin QFN package wirebond diagram

(Die plot updated for Caravel Openframe version v1)

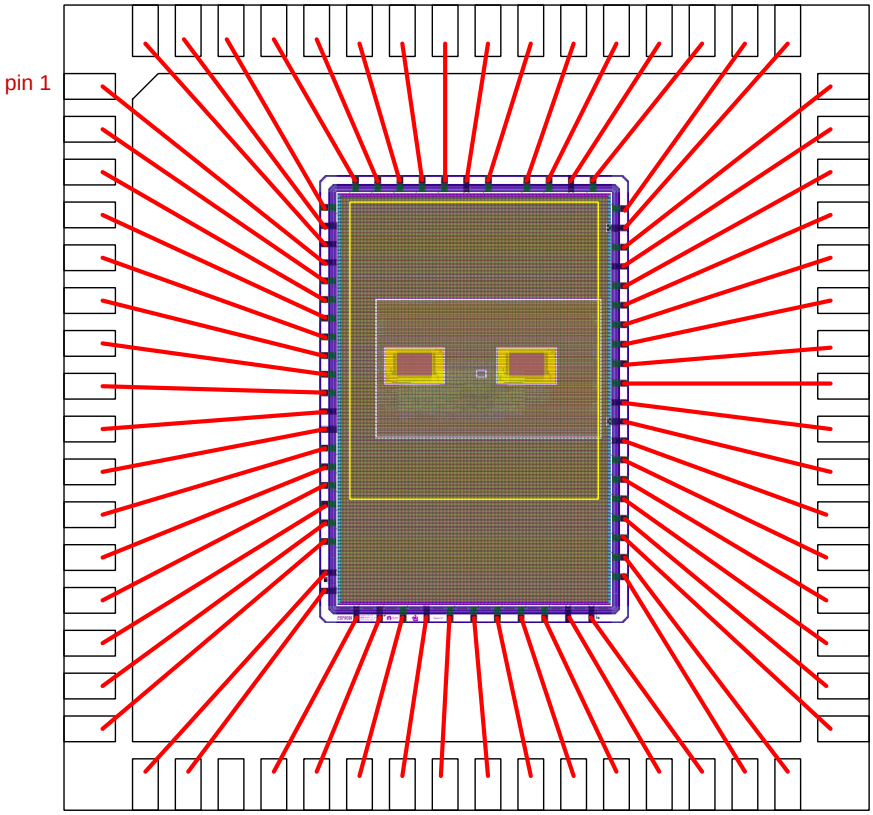

Figure 9. Wirebond diagram for the Caravel Openframe project example in a 64-pin QFN package.

package size = 9 mm × 9 mm
paddle size = 7.63 mm
pin pitch = 0.5 mm

Caravel padframe dimensions = 3.588 mm × 5.188 mm

Caravel die dimensions = 3.60 mm × 5.20 mm

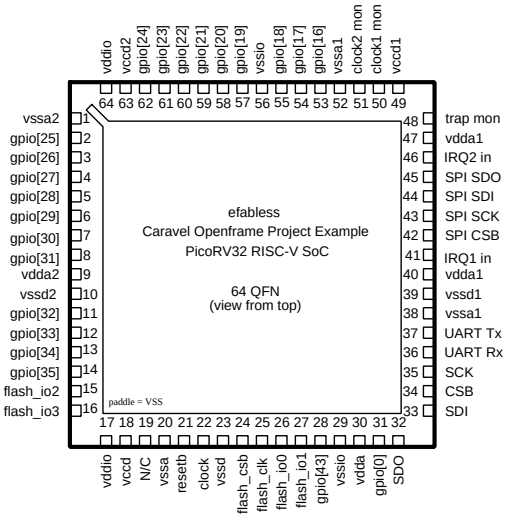

Figure 10. 64-pin QFN package pinout

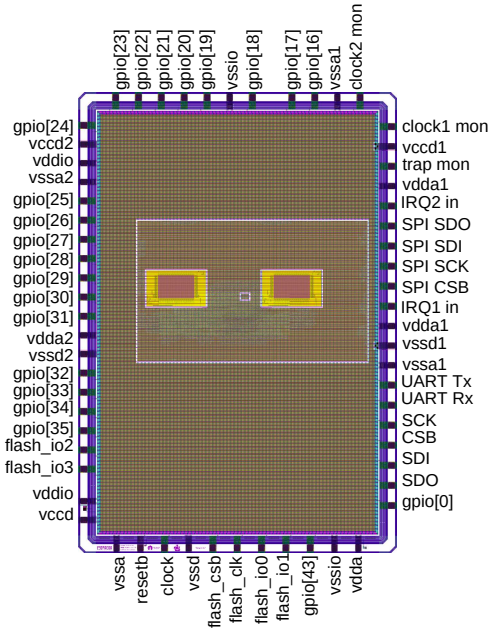

Figure 11. Caravel openframe die pinout

Known errors in the efabless Caravel Openframe harness version 1:

There are no known errors in Caravel Openframe version 1 at this time.

Documentation errata:

There are no known errors in the Caravel Openframe documentation at this time.