1125 Carrier Kasli-SoC

Features

- RJ45 10/100/1000T Ethernet connector
- 4 SFP 12Gb/s slots for DRTIO at 2.5Gb/s
- 12 EEM ports for daughtercards
- Xilinx Zynq-7000 SoC with Kintex-7 FPGA
- SD card flash memory

Applications

- Run ARTIQ kernels
- Communicate with the host
- Control other Sinara EEM cards
- Distributed Real-Time I/O

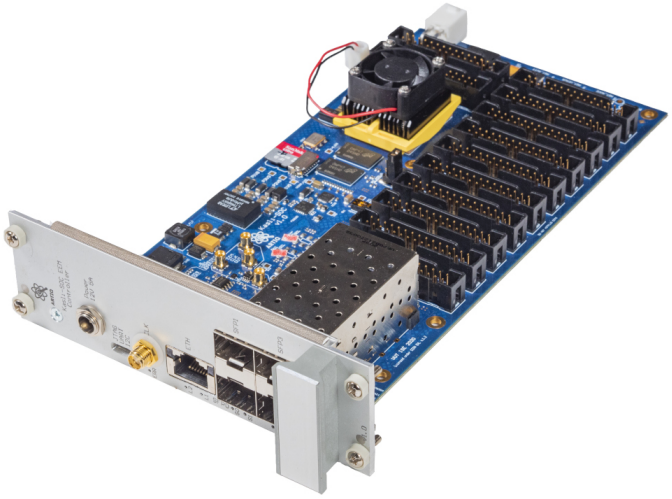

Figure 1: Kasli-SoC card

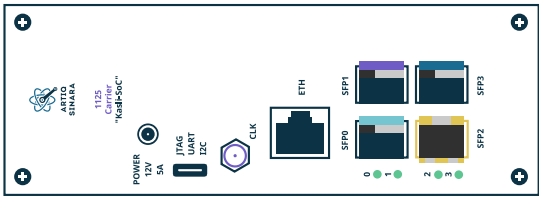

Figure 2: Kasli-SoC front panel

General Description

The 1125 Kasli-SoC Carrier card is an 8hp EEM module, designed to run ARTIQ-Zynq kernels sent over the network from a host machine. Kasli-SoC is built around a Xilinx Zynq-7000 SoC, capable of running more complex computations at high speed than its sister card 1124 Kasli 2.0. It supports up to 12 EEM connections to other EEM cards in the ARTIQ-Sinara family and up four SFP connections for comunications with other carriers. A dedicated Ethernet port is used for communications with the host.

Real-time control of EEM daughtercards is implemented using the ARTIQ RTIO system. 1ns temporal resolution can be achieved for TTL events.

4 SFP 12Gb/s slots are provided. These can be used by the ARTIQ Distributed Real-Time Input/Output (DRTIO) system, which allows for the use of additional core devices (e.g. Kasli or other Kasli-SoCs) as satellite cards, capable of running subkernels or relaying commands to a larger number of peripherals.

Source

Kasli-SoC, like all the Sinara hardware family, is open-source hardware, and design files (schematics, PCB layouts, BOMs) can be found in detail at the repository https://github.com/sinara-hw/Kasli-SOC/.

Electrical Specifications

External clock parameters are derived based on the internal termination specified in UG471[1] and the voltage range specified in DS191[2]. These figures account for the insertion loss of the RF transformer (TC2-1TX+[3]).

Table 1: Recommended Operating Conditions

Parameter	Min.	Typ.	Max.	Unit	Conditions
Clock input					
Input frequency		125		MHz	Si5324 synthesizer bypassed
		10/80/100/125		MHz	RTIO clock synthesized from input
Power	-9	1.5	5.5	dBm	
Power supply rating		12V, 5A			

Power is to be supplied either through the barrel connector in the front panel (size 5.5 mm OD, 2.5 mm ID) or the Molex connector at the back of the card (compatible with e.g. Sinara 1106 EEM AC Power Module). It is passed on to daughtercards through the EEM connections. Locking barrel connectors are supported.

SoC

Kasli-SoC features a XC7Z030-3FFG676E Xilinx Zynq-7000 System-on-Chip with a Kintex-7 FGPA and an Cortex-A9 dual-core processor to facilitate high-speed real-time control of inputs and outputs. The use of the SoC allows for more complex computations at higher speed than Kasli 2.0's purely on-FPGA CPU. Usually, the SoC is flashed with firmware and gateware binaries compiled from the ARTIQ (Advanced Real-Time Infrastructure for Quantum physics) control system, which equips the carrier board with the ability to control other Sinara EEMs and run ARTIQ experiment kernels.

A micro-USB located on the front panel is equipped for JTAG, I2C, and UART serial output. The serial interface runs at 115200bps 8-N-1.

Firmware/ARTIQ

ARTIQ is open-source and can be found in the repository https://github.com/m-labs/artiq. Orders of Sinara hardware are normally preflashed with suitable firmware and gateware binaries. Long-term support for ARTIQ systems can also be purchased, including updated binaries through AFWS (the ARTIQ Firmware Service).

ARTIQ-supported core devices based on Zynq-7000 SoCs, including Kasli-SoC, require firmware and gateware compiled from the ARTIQ-Zynq port, which can be found in the repository https://git.m-labs.hk/M-Labs/artiq-zynq.

Note on distributed RTIO (DRTIO)

DRTIO is the time and data transfer system which allows ARTIQ RTIO channels to be distributed among several core device carrier boards, synchronized and controlled by a central master device. The system itself is described

[1] https://docs.amd.com/v/u/en-US/ug471_7Series_SelectIO
[2] https://docs.amd.com/v/u/en-US/ds191-XC7Z030-XC7Z045-data-sheet
[3] https://www.minicircuits.com/pdfs/TC2-1TX+.pdf

in more detail in the ARTIQ documentation[4]. Within ARTIQ, core devices, including Kasli-SoC, can take one of three roles:

1. **Master**
   A DRTIO system must contain one DRTIO master. It controls its own local RTIO channels and the downstream DRTIO satellite(s). It requires a direct network connection to the host machine. It may make downstream connections to satellites.
2. **Satellite**
   Other core devices in a DRTIO system are DRTIO satellites. They require an upstream connection to one other core device, master or satellite, through which communications are carried to the master. They may make further downstream connections to other satellites. They may control their local RTIO channels directly through subkernels or simply pass on communications from the master.
3. **Standalone**
   When run in a non-distributed ARTIQ configuration, with a single central core device but without satellites, that core device is known as standalone.

Communication Interfaces

Communication between core devices is implemented with 1000Base-T small form-factor pluggable (SFP) interfaces. Four are available on 1125 Kasli-SoC. Each SFP connector possesses an indicator LED.

Transceiver maximum speed is 12.5 Gb/s[5]. DRTIO is normally run at 2.5 Gb/s with 8b10b encoding.

Additionally, a RJ45 10/100/1000T Ethernet port is featured for network connection to a host machine.

Upstream connection

- **Standalone/Master**
  A network-connected Ethernet cable should be attached the front panel Ethernet port to enable communication with a host machine.
- **Satellite**
  Satellites must acquire an upstream connection to another satellite or the master. The `SFP0` port should be connected to one of the free SFP slots on an upstream core device, using a cable connection with SFP transceivers.

Downstream connection

Kasli-SoC supports up to 4 DRTIO satellite connections per device. Any of the 4 downstream SFP ports (i.e. `SFP0`, `SFP1`, `SFP2`, `SFP3`) may be freely used. Port `SFPn` normally receives the destination number `n + 1`.

Clock Routing

Standalone/Master

The RTIO clock is typically synthesized by the Si5324 clock multiplier and distributed by the ADCLK948 clock fanout buffer to both the SoC and the MMCX connectors. Alternatively, an external reference can be supplied through the front panel SMA connector. It is then buffered in the SoC and sent to the Si5324 for clock synthesis. Kasli-SoC supports a set of RTIO clock options:

[4] https://m-labs.hk/artiq/manual/drtio.html
[5] https://www.amd.com/en/products/adaptive-socs-and-fpgas/technologies/high-speed-serial.html

RTIO frequency	Configuration	Clock generation
100 MHz	`int_100`	internal crystal oscillator using PLL, 100 MHz output
125 MHz	`int_125`	internal crystal oscillator using PLL, 125 MHz output (default)
	`ext0_synth0_10to125`	external 10 MHz reference using PLL, 125 MHz output
	`ext0_synth0_80to125`	external 80 MHz reference using PLL, 125 MHz output
	`ext0_synth0_100to125`	external 100 MHz reference using PLL, 125 MHz output
	`ext0_synth0_125to125`	external 125 MHz reference using PLL, 125 MHz output

The clock synthesizer may also be bypassed, using the `ext0_bypass` option, which will accept a RTIO clock directly supplied to the SMA connector. The resulting signal is then routed to both the RTIO system and downstream satellites.

Clocking options in a running system should be configured by setting the value of the `rtio_clock` key to the desired configuration through the ARTIQ `artiq_coremgmt` command. For example, a RTIO frequency of 125MHz will be synthesized from an external 10 MHz signal after issuing the following command:

```
artiq_coremgmt config write -s rtio_clock ext0_synth0_10to125
```

and rebooting.

Satellite

The RTIO clock is recovered from the SFP transceiver connected to the upstream device. The resulting signal is then cleaned up by the Si5324 and routed to both the RTIO system and downstream satellites.

WRPLL

Kasli-SoC can be configured to use WRPLL, a clock recovery method making use of White Rabbit's DDMTD (Digital Dual Mixer Time Difference) and the card's Si549 oscillators, both to lock the main RTIO clock and to lock satellite clocks to master.

Configuring Boot Mode

Kasli-SoC is capable of booting either remotely, over JTAG USB, or directly from the SD card. See the ARTIQ manual for more instructions on how to correctly flash and boot a core device. Boot mode must be configured by flipping physical switches on the board. The boot mode DIP switches are located in the middle of the board. To boot from USB, flip both switches towards the label `JTAG`. To boot from the SD card, flip both switches towards the label `SD`.

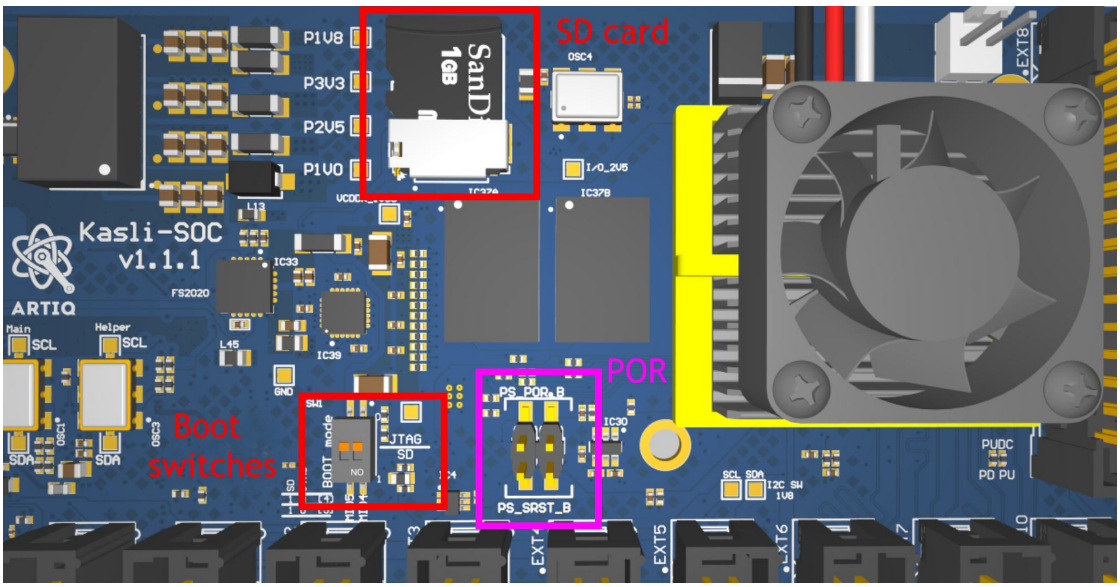

Figure 3: Position of DIP switches, SD card, and POR pins

POR jumpers and POR reset

A known Xilinx hardware bug prevents repeatedly booting over JTAG without a POR reset. If necessary, repeated boots can be made possible by physically setting a jumper on the POR pins (marked above) and triggering a reset over JTAG, see also the M-Labs POR reset script.[6]

User LEDs

Kasli-SoC designates two user LEDs for debugging purposes. Both are located on the PCB. The first, labeled `USER0`, can be found at the very bottom left of the PCB, below the SFP cage. The second, labeled `LD1`, can be found at the top left, roughly behind the micro-USB port.

ARTIQ System Description Entry

ARTIQ/Sinara firmware/gateware is generated according to a JSON system description file, allowing gateware to be specific to and optimized for a certain system configuration.

An example description file for a system using 1125 Kasli-SoC as a master core device might begin:

```
"target": "kasli_soc",
"variant": "my_variant",
"hw_rev": "v1.0",
"base": "master",
"peripherals": [    ]
```

[6] https://git.m-labs.hk/M-Labs/zynq-rs/src/branch/master/kasli_soc_por.py

where the `peripherals` list contains the corresponding entries for peripherals (daughtercards) in use.

For all accepted keys and values, see the JSON schema `coredevice_generic.schema.json` in the ARTIQ repository.[7].

Example ARTIQ Code

The sections below demonstrate simple usage scenarios of extensions on the ARTIQ control system. These extensions make use of the resources of the 1125 Kasli-SoC carrier. They do not exhaustively demonstrate all the features of the ARTIQ system.

The full documentation for ARTIQ software and gateware, including guides for their use, is available at https://m-labs.hk/artiq/manual/. Please consult the manual for details and reference material of the functions and structures used here.

Direct Memory Access (DMA)

Instead of directly emitting RTIO events, sequences of RTIO events can be recorded in advance and stored in the local SDRAM. The event sequence can then be replayed at a specified timestamp. This is of special advantage in cases where RTIO events are too closely placed to be generated as they are executed, as events can be replayed at a higher speed than the on-FPGA CPU alone is capable of.

The following example records an LED event sequence and replays it twice consecutively using `CoreDMA`. When run, the LED should blink twice.

```
@kernel
def record(self):
    with self.core_dma.record("led_blink"):
        delay(100*ms)
        self.led0.on()
        delay(100*ms)
        self.led0.off()

@kernel
def playback(self, n):
    handle = self.core_dma.get_handle("led_blink")
    self.core.break_realtime()
    for _ in range(n):
        self.core_dma.playback_handle(handle)

@kernel
def run(self):
    self.core.reset()
    self.record()
    self.playback(2)
```

Stored waveforms can be referenced and replayed in different kernels, but cannot be retrieved and must be regenerated if the core device is rebooted.

Dataset manipulation with core device cache

Experiments may require values computed or found in previously executed kernels. To avoid invoking an RPC or sacrificing the pre-computation in `prepare()` stage, data can be stored in the core device cache.

[7] https://github.com/m-labs/artiq/blob/release-8/artiq/coredevice/coredevice_generic.schema.json

The following code snippets describe two experiments, in which the data from the first experiment is cached. The data is then retrieved and printed in hexadecimal form in the second experiment.

```
@kernel
def put(self, key, value):
    self.core_cache.put(key, value)

# First experiment
@kernel
def run(self):
    self.put("data", [0xCAFE, 0xDEAD, 0xBEEF])
```

```
@kernel
def get(self, key):
    return self.core_cache.get(key)

@rpc(flags={"async"})
def p(self, p):
    print([hex(_) for _ in p])

# Second experiment
@kernel
def run(self):
    self.p(self.get("data"))
```

Similar to DMA, cached data is no longer retrievable once the core device has been rebooted.

Ordering Information

To order, please visit https://m-labs.hk and choose 1125 Carrier Kasli-SoC in the ARTIQ/Sinara hardware selection tool. Cards can be ordered as part of a fully-featured ARTIQ/Sinara crate or standalone through the 'Spare cards' option. Otherwise, orders can also be made by writing directly to mailto:sales@m-labs.hk.