**Overview**

This document specifies EDN hardware IP functionality. This module conforms to the [Comportable guideline for peripheral functionality.](https://github.com/lowRISC/opentitan/blob/master/doc/contributing/hw/comportability/README.md)

**Features**

The Entropy Distribution Network (EDN) block provides both hardware and software interfaces to the [CSRNG IP](https://github.com/lowRISC/opentitan/blob/master/hw/ip/csrng/README.md) module. A primary objective of the EDN block is to provide a simpler hardware interface for the peripheral blocks to use, in which case they do not have to directly interface with the CSRNG module.

* The EDN block provides a set of registers for firmware to manage a CSRNG application interface port.
* There are eight request/acknowledge hardware interfaces.
* Each hardware interface supports a fixed bus width of 32 bits.
* The EDN block has an "auto request mode" where generate and reseed CSRNG application commands can be programmed to be done continuously in hardware.
* There is also a "boot-time request mode", where a single TL-UL configuration write will trigger a proper CSRNG application command sequence to fetch the pre-FIPS entropy for tasks immediately at boot-time or after reset.
* There are two interrupts that are supported:
  + CSRNG application command has completed.
  + An internal FIFO error has occurred.
* There are two alerts that are implemented in this revision:
  + A fatal alert to report common security fatal errors and EDN specific fatal errors. A list of fatal errors are listed in the [ERR\_CODE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/data/edn.hjson#err_code) register.
  + A recoverable alert to report recoverable error. A list of fatal errors are listed in the [RECOV\_ALERT\_STS](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/data/edn.hjson#recov_alert_sts) register.

**Description**

This IP block acts as a gasket between peripheral hardware blocks and the CSRNG block. One function this IP block performs is to translate data transfer size. For example, CSRNG will return 128 bits on the genbits bus. A peripheral block will connect with a 32-bit data bus. The EDN block will move the first 32 bits from the returned genbits bus, and hold on to the remaining data until another request asks for more data. Furthermore, if data is not consumed immediately, the interface to the CSRNG will indicate back pressure to the CSRNG block. Each of the hardware interfaces can request data such that any genbits bus return can supply any requesting peripheral block.

At most one hardware peripheral block can connect to each EDN peripheral port. Hardware peripherals request more data from the EDN by asserting the req signal. When random values are available, the EDN transmits them on the bus and asserts an ack signal to signify the arrival of fresh values.

Application interface commands to the CSRNG block can be generated by either firmware or hardware.

Firmware can issue CSRNG commands on behalf of hardware peripherals, by writing the commands to the [SW\_CMD\_REQ](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/data/edn.hjson#sw_cmd_req) register. The command status response is captured in the [SW\_CMD\_STS](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/data/edn.hjson#sw_cmd_sts) register. Even when CRSNG generate commands are issued by firmware, all random values are distributed to the hardware peripherals.

If firmware applications require random values for their *own* use, they must issue the commands directly to the CSRNG, which maintains a dedicated CSRNG instance for firmware that is accessible through TL-UL.

There are two modes for EDN hardware to generate CSRNG commands. One is the "auto request mode", where two FIFOs are used to send commands. The general operation of this mode is that the CSRNG instance is set up by firmware, then the FIFOs are preloaded with commands. One FIFO can be programmed to send generate commands. The other FIFO can be programmed to send reseed commands. The [MAX\_NUM\_REQS\_BETWEEN\_RESEEDS](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/data/edn.hjson#max_num_reqs_between_reseeds) register sets the number of generate commands allowed between reseed commands. Once this is done, the EDN block can request data from the CSRNG once firmware has instantiated the associated instance through the EDN command forwarding interface. When in this mode, the EDN emits generate commands from the first FIFO to get more data. Once the [MAX\_NUM\_REQS\_BETWEEN\_RESEEDS](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/data/edn.hjson#max_num_reqs_between_reseeds) timer expires, the EDN block emits a reseed command from the second FIFO. The process of sending these two commands will repeat forever until the EDN\_ENABLE field is cleared, the AUTO\_REQ\_MODE field is cleared, or the EDN is reset.

Any of the command FIFOs can be reset by asserting the CMD\_FIFO\_RST field in the [CTRL](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/data/edn.hjson#ctrl) register.

The other mode is "boot-time request mode", where only the hardware generates CSRNG application interface commands. In this mode a single instantiate command is sent, followed by a stream of generate commands. This sequence fetches the initial random values needed for the system to boot. Use of boot-time request mode, though simpler in operation, is only for applications which do not require FIPS-approved random values. Please see the [entropy\_src IP documentation](https://github.com/lowRISC/opentitan/blob/master/hw/ip/entropy_src/README.md) for more information on trade-offs when creating CSRNG seeds before the completion of the FIPS-required health checks. In boot-time request mode the generate commands continue until EDN\_ENABLE field is cleared (set to false), the BOOT\_REQ\_MODE field is cleared, or the EDN is reset. Note that when the EDN\_ENABLE field is cleared or the BOOT\_REQ\_MODE field is cleared, an uninstantiate command needs to be sent by firmware to destroy the instance in csrng. Note that the EDNs and CSRNG should always be reset together to ensure proper instantiation or uninstantiation of state variables.

**Security**

All module assets and countermeasures performed by hardware are listed in the hjson countermeasures section. Labels for each instance of asset and countermeasure are located throughout the RTL source code.

The receiving FIFO for genbits from CSRNG will have a hardware check on the output bus. This is done to make sure repeated values are not occurring. Only 64 bits (out of 128 bits) are checked, since this is statistically significant, and more checking would cost more silicon. It is expected that an endpoint requiring high-quality entropy will do an additional consistency hardware check on the 32 bit data bus. Additionally the FIPS signal on the endpoint bus should also be checked for high-quality entropy consumers. Boot request mode is an example where the FIPS signal will not be ever be set, and consuming endpoint of low-quality entropy do not need to check this signal.

**Example Topology**

In general, the OpenTitan random number subsystem consists of one entropy\_src, one CSRNG, and one or more EDNs. The entropy\_src only supports one connection to a CSRNG, but the CSRNG has multiple application interface ports for connecting to EDN's or other hardware blocks. In the example below, two EDN modules are used to distribute genbits from the CSRNG to peripheral modules. (Note that this is just an example; consult the specification of a specific top-level design for the specific configuration used there.)

**Theory of Operation**

The EDN is for distributing random number streams to hardware blocks, via peripheral ports on the EDN. Each block connected to a peripheral port is referred to as an endpoint.

To enable the EDN block, set the EDN\_ENABLE field in the [CTRL](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#ctrl) register..

**Interaction with CSRNG Application Interface Ports**

The CSRNG application interface implements the "function envelopes" recommended by [NIST SP 800-90A](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf) for random number generation, and these function envelopes establish certain requirements for the order of operations. For instance, the application interface port must receive an explicit instantiate command before receiving any generate commands. The sequences of commands generated by a particular EDN are either controlled by the EDN state machine or by commands forwarded from firmware through the [SW\_CMD\_REQ](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_req) register.

Whenever commands are directly forwarded from firmware to the CSRNG through the [SW\_CMD\_REQ](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_req) register, firmware must poll the CMD\_REG\_RDY and the CMD\_RDY bits of the [SW\_CMD\_STS](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_sts) register before sending any further commands. CMD\_RDY indicates whether the EDN is ready to receive a new command, whereas CMD\_REG\_RDY indicates whether the EDN is ready to accept the next word of the command. Each command consists of a command header, which is represented by a single word. Furthermore, a command can have up to 12 words of additional data. To check whether a command has been acknowledged by the CSRNG, the CMD\_ACK bit of the [SW\_CMD\_STS](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_sts) register can be polled. Whenever CMD\_ACK is high, the CMD\_STS bit can be checked to find out if an error occured.

Note that CSRNG commands are to be written into the [SW\_CMD\_REQ](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_req), [RESEED\_CMD](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#reseed_cmd), and [GENERATE\_CMD](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#generate_cmd) registers. CSRNG command format details can be found in [CSRNG](https://github.com/lowRISC/opentitan/blob/master/hw/ip/csrng/README.md).

There are two broad modes for state machine control: auto request mode and boot-time request mode.

**Boot-time Request Mode**

Random values are needed by peripherals almost immediately after reset, so to simplify interactions with the boot ROM, boot-time request mode is the default mode.

In boot-time request mode, the command sequence is fully hardware-controlled and no command customization is possible. In this mode, the EDN automatically issues a special reduced-latency instantiate command followed by the default generate commands. This means, for instance, that no personalization strings or additional data may be passed to the CSRNG application interface port in this mode. On exiting, the EDN issues an uninstantiate command to destroy the associated CSRNG instance.

Once firmware initialization is complete, it is important to exit this mode if the endpoints ever need FIPS-approved random values. Should another generate command be needed, it can only be issued after exiting boot mode. This is done by either *clearing* the EDN\_ENABLE field or *clearing* the BOOT\_REQ\_MODE field in [CTRL](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#ctrl) to halt the boot-time request state machine. Firmware must then wait for the transition of the state machine by polling the CMD\_RDY field of the [SW\_CMD\_STS](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_sts) register or wait for the state machine to enter the SW mode by polling the [MAIN\_SM\_STATE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#main_sm_state) register.

It should be noted that when in boot-time request mode, no status will be updated that is used for the software port operation. If some hang condition were to occur when in this mode, the main state machine debug register should be read to determine if a hang condition is present. There is a limit to how much entropy can be requested in the boot-time request mode BOOT\_GEN\_CMD command (GLEN = 4K). It is the responsibility of software to switch to the software mode of operation before the command has completed. If the BOOT\_GEN\_CMD command ends while an endpoint is requesting, EDN will never ack and the endpoint bus will hang.

**Note on Security Considerations when Using Boot-time Request Mode**

Boot-time request mode is not intended for normal operation, as it tolerates the potential use of preliminary seeds for the attached CSRNG instance. These preliminary seeds are described as "pre-FIPS" since they are released from the entropy\_src before the complete start-up health-checks recommended by FIPS have been completed. Thus pre-FIPS seeds have weaker guarantees on the amount of physical entropy included in their creation. As detailed in the [entropy\_src documentation](https://github.com/lowRISC/opentitan/blob/master/hw/ip/entropy_src/README.md), only the first CSRNG seed created after reset is pre-FIPS. All following seeds from the entropy\_src are passed through the full FIPS-approved health checks. Therefore at most one EDN can receive a pre-FIPS seed after reset. Since boot-time request mode EDN streams may be FIPS non-compliant, firmware must at some point disable boot-time request mode and reinitialize the EDN for either firmware-driven operation or auto request mode.

**Multiple EDNs in Boot-time Request Mode**

If many endpoints require boot-time entropy, multiple boot-time EDNs may be required, as the EDN has a fixed maximum number of peripheral ports. Since physical entropy generation takes time, there exists a mechanism to prioritize the EDNs, to match the boot priority of each group of attached endpoints. To establish an order to the instantiation of each EDN, enable them one at a time. To ensure that the most recently enabled EDN will get next priority for physical entropy, poll the [MAIN\_SM\_STATE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#main_sm_state) CSR until EDN's state machine has moved past the BootInsAckWait state (at which point CSRNG has acknowledged the Instantiate command sent by that EDN). Once that has happened, the next EDN can be enabled.

If using boot-time request mode, the CSRNG seed material used for the first-activated EDN is the special pre-FIPS seed, which is specifically tested quickly to improve latency. The first random values distributed from this EDN will therefore be available roughly 2ms after reset. The entropy\_src only creates one pre-FIPS seed, so any other EDNs must wait for their seeds to pass the full FIPS-recommended health checks. This means that each subsequent EDN must wait an additional 5ms before it can start distributing data. For instance, if there are three boot-time request mode EDN's in the system, the first will start distributing data 2ms after reset, the second will start distributing data 7ms after reset, and the third will start distributing data 12ms after reset.

**Auto Request Mode**

Before entering auto request mode, it is the responsibility of firmware to first generate an instantiate command for the EDN-associated instance via the [SW\_CMD\_REQ](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_req) register. The required generate and reseed commands must also be custom generated by firmware and loaded into the respective command replay FIFOs via the [GENERATE\_CMD](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#generate_cmd) and [RESEED\_CMD](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#reseed_cmd) registers. These generate commands will be issued as necessary to meet the bandwidth requirements of the endpoints. The reseed commands will be issued once every MAX\_NUM\_REQS\_BETWEEN\_RESEEDS generate requests. The reset value is 0. This implies that no generate commands will be issued unless this value is changed by firmware. For details on the options for application interface commands please see the [CSRNG IP Documentation](https://github.com/lowRISC/opentitan/blob/master/hw/ip/csrng/README.md). Once the CSRNG instance has been instantiated, and the generate and reseed commands have been loaded, auto request mode can be entered by programming the [CTRL](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#ctrl) register with EDN\_ENABLE and AUTO\_REQ\_MODE fields are enabled. Note that if BOOT\_REQ\_MODE is asserted the state machine will enter boot-time request mode, even if AUTO\_REQ\_MODE is asserted.

To issue any new commands other than those stored in the generate or reseed FIFOs, it is important to disable auto request mode, by deasserting the AUTO\_REQ\_MODE field in the [CTRL](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#ctrl) register. Firmware must then wait until the current command is completed by polling the [MAIN\_SM\_STATE](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#main_sm_state) register. Once the state machine returns to the Idle or SWPortMode states, new firmware-driven commands can be passed to the CSRNG via the [SW\_CMD\_REQ](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#sw_cmd_req) register. The generate and reseed FIFOs are reset under four circumstances. These circumstances are (a) when the EDN is disabled, (b) when the SWPortMode state is entered, (c) when the boot sequence has completed, or (d) when the EDN enters the Idle state after it finishes operation in auto mode.

It should be noted that when in auto request mode, no status will be updated that is used for the software port operation once the instantiate command has completed. If some hang condition were to occur when in this mode, the main state machine debug register should be read to determine if a hang condition is present.

**Note on State Machine Shutdown Delays**

When leaving boot-time request mode or auto request mode, the EDN state machine waits for completion of the last command, before sending a shutdown acknowledgement to firmware. The longest possible commands are the instantiate or reseed requests, which typically take about 5ms, due to the time required to gather the necessary physical entropy. By contrast, the largest possible generate command allowed by [NIST SP 800-90A](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf) is for 219 bits (or 4096 AES codewords). Assuming an AES encryption delay of 16 clocks, and a 100 MHz clock frequency, the longest allowable generate command would take only 0.7 ms to complete.

**Note on Sharing of CSRNG Instance State Variables**

Once an application interface port has received an instantiate command, that application interface port then has access to a unique CSRNG instance, which is shared by all endpoints on the same EDN. Therefore from a security perspective, an attack to that particular CSRNG instance is an attack on all the endpoints that share the same EDN. Meanwhile, seeds and other state variables specific to a particular CSRNG instance are not shared between endpoints on *separate* EDN instances, or with any hardware devices with direct connections to dedicated CSRNG application interface ports.

**Interactions with Peripheral Devices**

Peripheral ports distribute data to the endpoints using four signals: req, ack, bus, and fips.

Fresh (i.e. previously unseen) random values are distributed to the endpoints via the 32 bit bus signal, in response to a req signal. Whenever new values are placed on the bus, the ack is asserted until the values are consumed by the endpoint, as indicated by simultaneous assertion of the req and ack signals in the same cycle. Otherwise ack is deasserted until enough fresh bits are received from CSRNG. The bus data will persist on the bus until a new req is asserted. This persistence will allow an asynchronous endpoint to capture the correct data sometime after the ack de-asserts.

The fips signal is used to identify whether the values received on the bus have been prepared with complete adherence to the recommendations in NIST SP 800-90. If the fips signal is deasserted, it means the associated CSRNG instance has been instantiated with a pre-FIPS seed.

**Design Details**

**EDN Initialization**

After power-up, the EDN block is disabled. A single TL-UL configuration write to the [CTRL](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/registers.md#ctrl) register will start random-number streams processing in boot-time request mode. CSRNG application commands will be sent immediately. Once these commands have completed, a status bit will be set. At this point, firmware can later come and reconfigure the EDN block for a different mode of operation.

The recommended write sequence for the entire entropy system is one configuration write to ENTROPY\_SRC, then CSRNG, and finally to EDN (also see [Module enable and disable](https://github.com/lowRISC/opentitan/blob/master/hw/ip/edn/doc/programmers_guide.md#module-enable-and-disable)).

**Interrupts**

The EDN module has two interrupts: edn\_cmd\_req\_done and edn\_fatal\_err.

The edn\_cmd\_req\_done interrupt should be used when a CSRNG command is issued and firmware is waiting for completion.

The edn\_fatal\_err interrupt will fire when a fatal error has been detected. The conditions that cause this to happen are FIFO error, a state machine error state transition, or a prim\_count error.

**Waveforms**

See the [CSRNG IP](https://github.com/lowRISC/opentitan/blob/master/hw/ip/csrng/README.md) waveform section for the CSRNG application interface commands.

**Peripheral Hardware Interface - Req/Ack**

The following waveform shows an example of how the peripheral hardware interface works. This example shows the case where the boot-time mode in the ENTROPY\_SRC block is enabled. This example also shows the case where the next request will change the prior data by popping the data FIFO.