Signal	Width	Direction	Description
clk_i	1	In	Main clock input for synchronous operations
rst_ni	1	In	Active-low synchronous reset
rst_shadowed_ni	1	In	Active-low reset for shadow registers
clk_edn_i	1	In	Clock input for entropy source synchronization
rst_edn_ni	1	In	Active-low reset for entropy source
tl_i	Struct	In	Data input from tile link
kmac_data_i	Struct	In	Data response from KMAC
kmac_en_masking_i	1	In	Masking enable signal for KMAC
lc_keymgr_en_i	Struct	In	Enable signal from lifecycle controller
lc_keymgr_div_i	Struct	In	Divider input from lifecycle controller
otp_key_i	Struct	In	OTP-provided cryptographic keys
otp_device_id_i	Struct	In	Device identifier from OTP
flash_i	Struct	In	Flash-controller provided keys and seeds
edn_i	Struct	In	Entropy distribution input
rom_digest_i	Struct	In	ROM digest data
alert_rx_i	Struct	In	Alerts received from other modules
tl_o	Struct	Out	Data output to tile link
aes_key_o	Struct	Out	Key request output to AES
kmac_key_o	Struct	Out	Key request output to KMAC
otbn_key_o	Struct	Out	Key request output to OTBN
kmac_data_o	Struct	Out	Data request to KMAC
edn_o	Struct	Out	Entropy distribution request
intr_op_done_o	1	Out	Interrupt signal indicating operation completion
alert_tx_o	Struct	Out	Alert signal transmission

Functional Description (Generated by funcgen)

Verilog Design Modules Functional Description

Module: `keymgr` (File: keymgr.sv)

Purpose

The `keymgr` module is the top-level module of the Key Manager, responsible for generating, managing, and distributing cryptographic keys within the system. It interfaces with various cryptographic modules and external control signals to ensure secure key operations.

Parameters

- `AlertAsyncOn` [default: `{NumAlerts{1'b1}}`]: Controls the asynchronous nature of alerts.
- `UseOtpSeedsInsteadOfFlash` [default: `1'b0`]: Determines the source of seed values (OTP vs Flash).
- `KmacEnMasking` [default: `1'b1`]: Enables masking for KMAC.
- Multiple `RndCnst` parameters (e.g., `RndCnstLfsrSeed`, `RndCnstRandPerm`): Default seeds and permutations for randomness.

Ports

- **Control and Clock Signals**
  - `clk_i` (input, 1-bit): Primary clock signal.
  - `rst_ni`, `rst_shadowed_ni`, `rst_edn_ni` (input, 1-bit each): Reset signals.
  - `clk_edn_i` (input, 1-bit): EDN clock signal.
- **Bus Interface**
  - `tl_i` (input): Bus transaction request.
  - `tl_o` (output): Bus transaction response.
- **Key and Data Outputs**
  - `aes_key_o`, `kmac_key_o`, `otbn_key_o` (output): Key requests to other crypto modules.
  - `kmac_data_o` (output): Data request to KMAC.
- **Input Interfaces**
  - `kmac_data_i` (input): Data response from KMAC.
  - `kmac_en_masking_i` (input): Indicator of KMAC masking.
  - Multiple config control signals (e.g., `lc_keymgr_en_i`, `otp_key_i`, `flash_i`): Various control and data inputs.
  - `edn_i`, `rom_digest_i` (input): EDN and ROM_CTRL data.
- **Interrupts and Alerts**
  - `intr_op_done_o` (output): Operation done interrupt.
  - `alert_rx_i`, `alert_tx_o` (bidirectional): Alert signals to/from other modules.

Internal Signals

- Various internal logic signals to manage state, generate randomness, handle errors, and manage configuration.

Functionality

- **State Machines and Control Logic:**
  - State machines manage the sequence of key operations, responding to various inputs and transitioning through states like reset, key generation, and operation completion.
  - Manages random number generation using LFSR and reseeding through interactions with EDN.
- **Combinational and Sequential Operations:**
  - Combines various seeds and data inputs as part of cryptographic operations using logic and LFSR.
  - Uses assertions to ensure constraints on data path widths and correct operation.
- **Instantiations:**
  - **Keymgr Configuration Management:**
    - `u_cfgen`: Manages the configuration enable signal.
  - **Regeneration Modules:**
    - `u_reseed_ctrl`: Manages reseeding control and entropy management.
  - **Key and Data Handling Instantiations:**
    - `u_lfsr`: LFSR for obtaining entropy, seeded appropriately.
    - `u_ctrl`: Overall control of the key management operations.
    - `u_kmac_if`: Interface to KMAC, managing data flow and checking for errors.
    - `u_sideload_ctrl`: Sideload key control for managing keys used by hardware accelerators.

Module: `keymgr_cfg_en` (File: keymgr_cfg_en.sv)

Purpose

Defines the configuration enable logic for the key manager, allowing configuration to be reset and controlled correctly based on initialization and specific conditions.

Parameters

- `NonInitClr`: Controls whether the clear operation affects the output during non-initial states.

Ports

- `clk_i`, `rst_ni`: Clock and reset signals.

- `init_i`, `en_i`, `set_i`, `clr_i`: Control signals that influence the output.
- `out_o`: Configuration enable output signal.

Internal Signals

- `out_q`, `init_q`: Tracks the output and initialization states.

Functionality

- **Sequential Logic:**
  - Flip-flops hold the state of `out_q` and `init_q`, with clock and reset considerations.
  - Sequential control through always_ff blocks modifies these flip-flops based on input conditions.
- **Control Logic:**
  - Computes the valid clear, set, and disable based on initialization and other control signals.
- Responds to `set_i` and `clr_i` to control the configuration enable signals, influenced by `NonInitClr`.

Module: `keymgr_ctrl` (File: keymgr_ctrl.sv)

Purpose

Provides control logic for key manager operations, managing key creation and state transitions according to configured settings.

Parameters

- `KmacEnMasking`: Controls masking support for KMAC operations.

Ports

- Various input signals to receive faults, errors, operation requests, and data inputs.
- Output signals including operation status, control flags like data_valid_o, error and fault reporting, and intermediate control signals.

Internal Signals

- `state_q`, `state_d`: Current and next state of the control logic.
- `cnt`: Counter for operations.
- Signals to handle operation requests, acknowledgments, and state transitions.

Functionality

- **State Machines:**

  - Manages multiple states from initialization to generating keys to working with faults and errors.
  - Interfaces with other modules to manage data integrity and key updates.
- **Control Mechanisms:**
  - Decides whether to advance states, generate IDs, or generate outputs based on current configuration and operational requirements.
- **Error Handling:**
  - Processes various indicators of error states and sets appropriate outputs for fault management.

Module: `keymgr_data_en_state` (File: keymgr_data_en_state.sv)

Purpose

Controls the enablement of data paths for hardware and software keys, providing redundancy for data validation checks.

Ports

- Control signals for enablement and errors (`data_hw_en_o`, `data_sw_en_o`, `fsm_err_o`).

Internal Signals

- `state_q`, `state_d`: States for controlling data enablement.

Functionality

- **Finite State Machine:**
  - Encodes states to manage enablement of data paths through strict state changes.
- Utilizes sparse FSM encoding to ensure robustness against state transition errors.

Module: `keymgr_err` (File: keymgr_err.sv)

Purpose

Collects and determines sync/async errors and faults based on ongoing operations in the key manager.

Ports

- Input signals to capture errors and faults, output signals to report detected conditions.

Internal Signals

- Registers and logic to track errors and faults during operations.

Functionality

- **Error Handling:**
  - Differentiate between sync and async errors/faults.
  - Provides error signals and tracks conditions relevant to key management based on inputs.

Module: `keymgr_input_checks` (File: keymgr_input_checks.sv)

Purpose

Checks the validity of input data to ensure that only valid keys and other data are processed.

Parameters

- `KmacEnMasking`, `NumRomDigestInputs`: Controls key masking and number of input checks.

Ports

- Inputs for key, seed, and other data, outputs for various validity indicators.

Internal Signals

- Logic arrays to handle checks and padding.

Functionality

- **Input Validity Checks:**
  - Uses utility functions to ensure data integrity based on logical checks and threshold comparisons.

Module: `keymgr_kmac_if` (File: keymgr_kmac_if.sv)

Purpose

Interacts with the KMAC module to handle key-based operations, ensuring data is sent and received according to protocol standards.

Parameters

- `RndCnstRandPerm`, `MaxAdvDataWidth`: Constants for random permutation and data width limitations.

Ports

- Clock, reset, and control ports for managing operations and interfacing with KMAC.

Internal Signals

- State and control signals to manage data transmission with KMAC.

Functionality

- **State Machines and Control:**
  - State machine to regulate KMAC operations, maintaining FSM integrity and ensuring only valid transactions proceed.
- **Error Management:**
  - Performs error checking and communicates status of operations in case of invalid data or conditions.

Module: `keymgr_op_state_ctrl` (File: keymgr__op__state__ctrl.sv)

Purpose

Controls the current operation's state, managing requests and coordinating KMAC interactions.

Ports

- Manages operation requests, KMack interactions, and acknowledgment signals to control logic.

Internal Signals

- State logic to manage idle, advanced, and other operational states.

Functionality

- **State Transitions:**
  - Logic for state transitions upon receiving operational requests.
- **Control Enablement:**
  - Checks validity and determines when to engage KMAC.

Module: `auto` (File: keymgr__reg__top.sv)

Purpose

Auto-generated register management module for the Key Manager, facilitating register interactions and error checking.

Ports

- Clock, reset, and `tlul_pkg` input/output for communication.

Internal Signals

- Signals managing register interfaces and TL-UL adapter connections.

Functionality

- **Register Management:**
  - Automatically interfaces with various registers, verifying writes and managing access enables.
- **Error Detection:**
  - Checks for address misses and write errors and provides interfaces for test asserts and checks.

Module: `keymgr_reseed_ctrl` (File: keymgr_reseed_ctrl.sv)

Purpose

Controls entropy reseeding, interfacing with an entropy distribution network (EDN) and providing entropy to LFSR within the key manager.

Ports

- Clock, reset signals, ports for receiving EDN input, and LFSR output.

Internal Signals

- Logic to control entropy requests to EDN and manage counts for reseeding.

Functionality

- **Entropy Management:**
  - Determines when to request new entropy based on controls and timing.
- **Error Handling:**
  - Provides mechanism to ensure reseed requests do not exceed specified intervals.

Module: `keymgr_sideload_key` (File: keymgr_sideload_key.sv)

Purpose

Manages sideload keys, providing a simple interface for setting, clearing, and holding valid key states.

Parameters

- `KeyWidth`: Configurable width of the session key.

Ports

- Interacts with the key's set, clear, and enable signals to determine valid operation.

Internal Signals

- Key holding registers and logic flags for tracking validity.

Functionality

- **Registers and Logic:**
  - Stores and controls the validity of the loaded keys, responding to set and clear signals.

Module: `keymgr_sideload_key_ctrl` (File: keymgr_sideload_key_ctrl.sv)

Purpose

Coordinates the management of all sideload keys, handling enablement, selection, and error conditions for different destination keys.

Ports

- Inputs for control and key management, outputs for prng and key propagation.

Internal Signals

- State tracking and enablement logic to ensure proper selection of active slots.

Functionality

- **Control Operations:**
  - FSM to control key sideloading, clearing, and wiping.
- **Error and Validation:**
  - Ensures only valid keys are selected and tracked, mitigating unexpected errors.

Inter-Module Connections

The `keymgr` module is the central controller, interfacing across: - **Configuration and Control**: Using `keymgr_cfg_en` and `keymgr_ctrl` to manage key generation and transitions. - **Entropy Management**: Collaborates with

keymgr_reseed_ctrl for secure entropy sources. - **Interface Implementations**: The keymgr_kmac_if provides external KMAC connections, while keymgr_sideload_key_ctrl deals with hardware sideloading tasks. - **Data Integrity**: Checked at multiple interfaces with keymgr_input_checks and keymgr_err. - **Internal and External Alerts**: Managed via prim_alert_pkg interfaces for proactive error handling and operation interruptions.

Overall, the key manager's architecture is designed for flexibility and security, interacting with various sub-modules to maintain a resilient keying system fundamental to cryptographic operations.