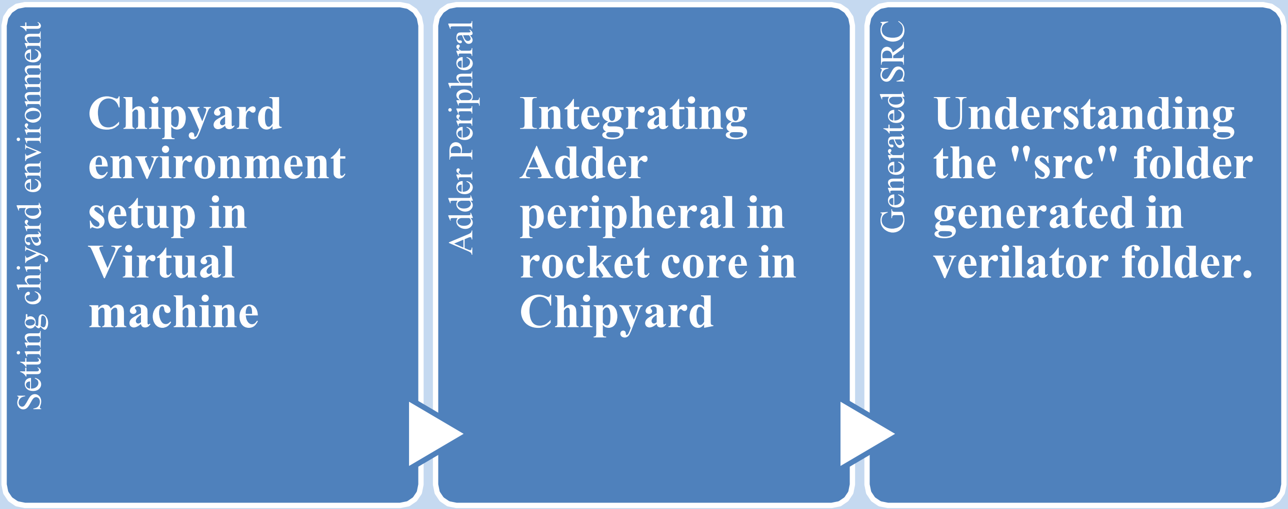
ADDER INTEGRATION IN CHIPYARD



Setting chiyard environment

Chipyard environment setup in Virtual machine

Adder Peripheral

Integrating Adder peripheral in rocket core in Chipyard

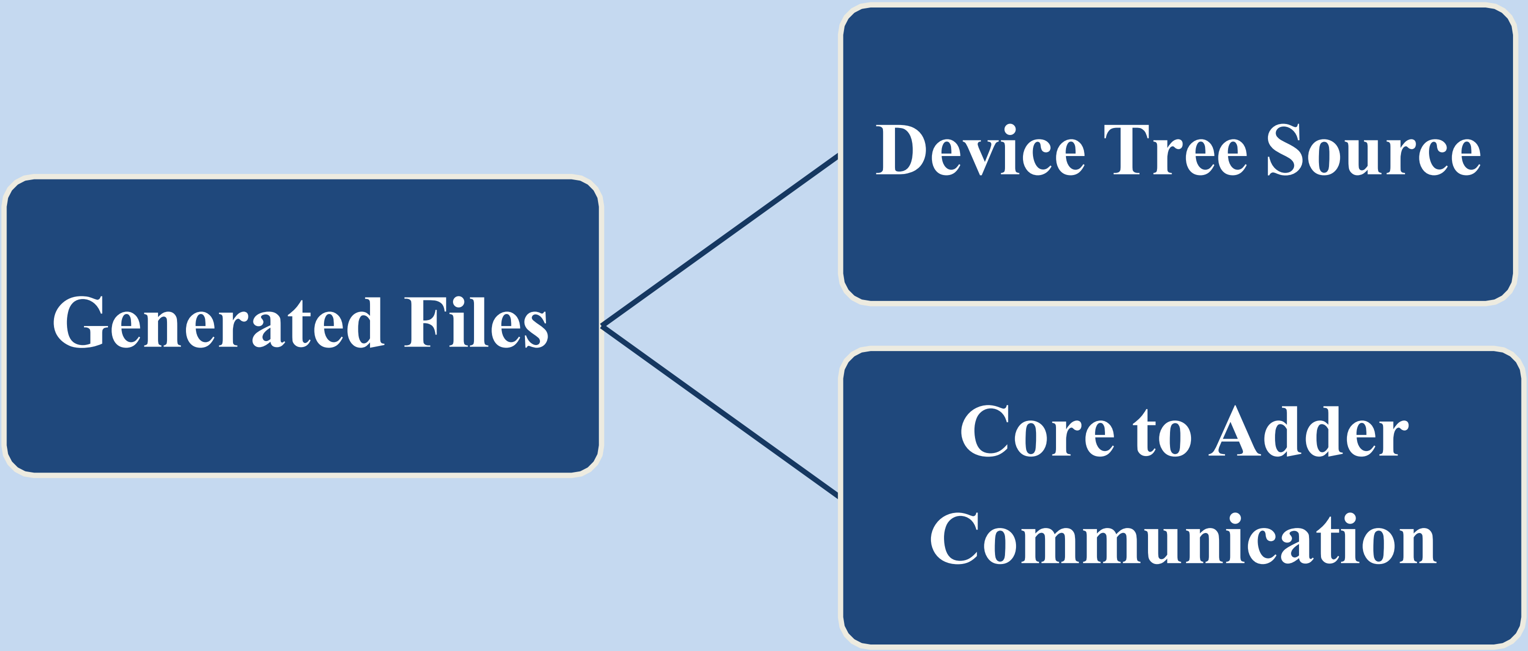
Generated SRC

Understanding the "src" folder generated in verilator folder.

COMMUNICATION OF CUSTOM PERIPHERAL CHIPYARD’S ROCKET CORE VIA TILELINK

Supervisor: Engr. Mehmoona Gul
Co-Supervisor: Dr. Aneesullah
Presented By: Ayesha Qazi
Humail Nawaz

Generated Files under discussion



Generated Files

Device Tree Source

Core to Adder Communication

Device Tree Source

- A device tree is a tree data structure with nodes that describe the devices in a system.
- A device tree is often used to describe devices which cannot necessarily be dynamically detected by a operating system.
- It is collection of nodes and their properties, where node represents a device, in a systematic way.

https://github.com/devicetree-org/devicetree-specification/releases/tag/v0.4-rc1

Device Tree Structure and Convection

Nodes

Parent and child nodes:

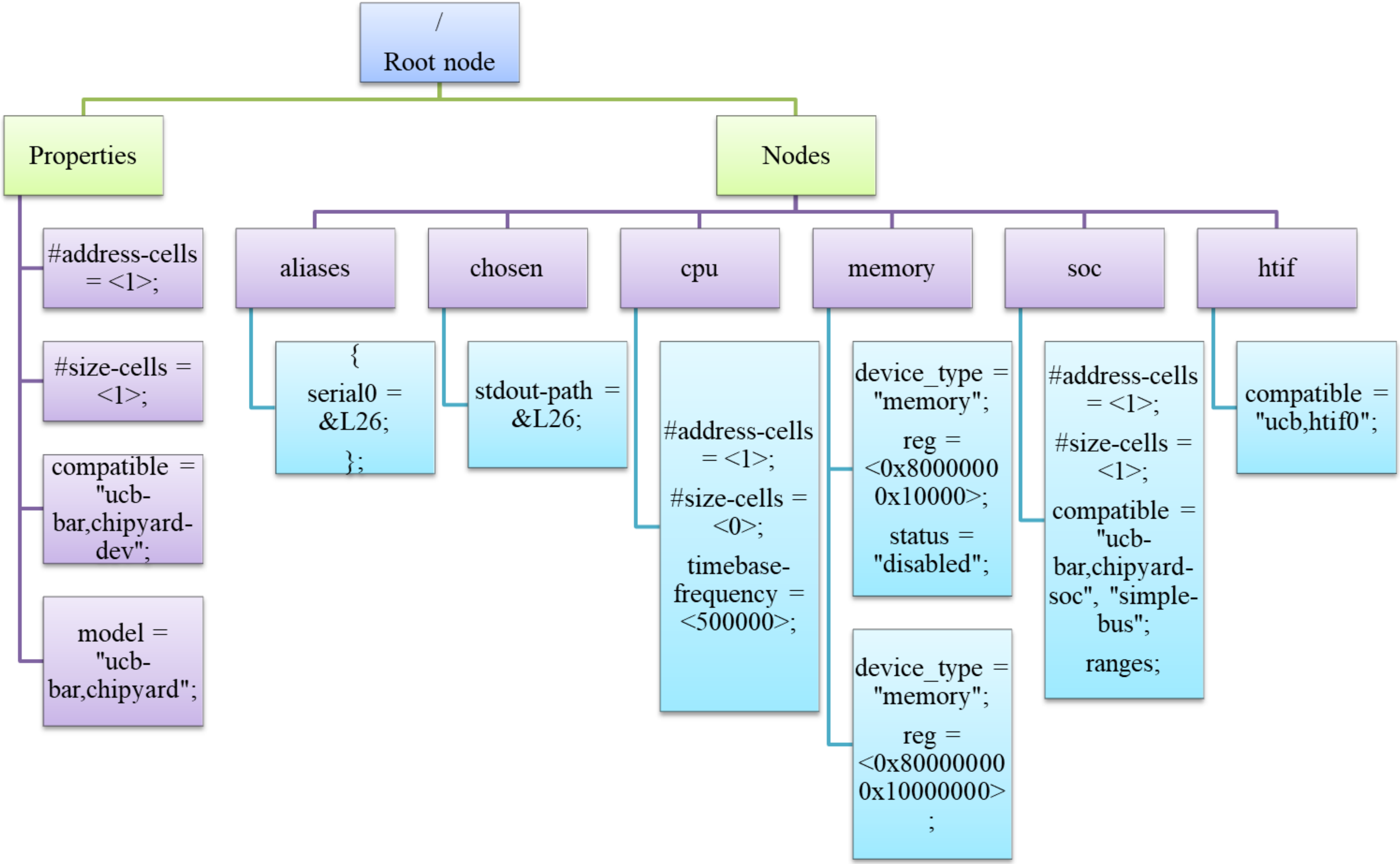
- Labels
- Unit address

Example:

```
memory@8000000 {
    device_type = "memory";
    reg = <0x8000000 0x10000>;
```

Node Properties:

1. Compatible
2. Model
3. Phandle
4. Status
5. #address and #size cells
6. Reg
7. Ranges



/

Root node

Properties

#address-cells = <1>;

#size-cells = <1>;

compatible = "ucb-bar,chipyard-dev";

model = "ucb-bar,chipyard";

Nodes

aliases

{
serial0 = &L26;
};

chosen

stdout-path = &L26;

cpu

#address-cells = <1>;

#size-cells = <0>;

timebase-frequency = <500000>;

memory

device_type = "memory";

reg = <0x8000000 0x10000>;

status = "disabled";

device_type = "memory";

reg = <0x80000000 0x10000000>;

soc

#address-cells = <1>;

#size-cells = <1>;

compatible = "ucb-bar,chipyard-soc", "simple-bus";

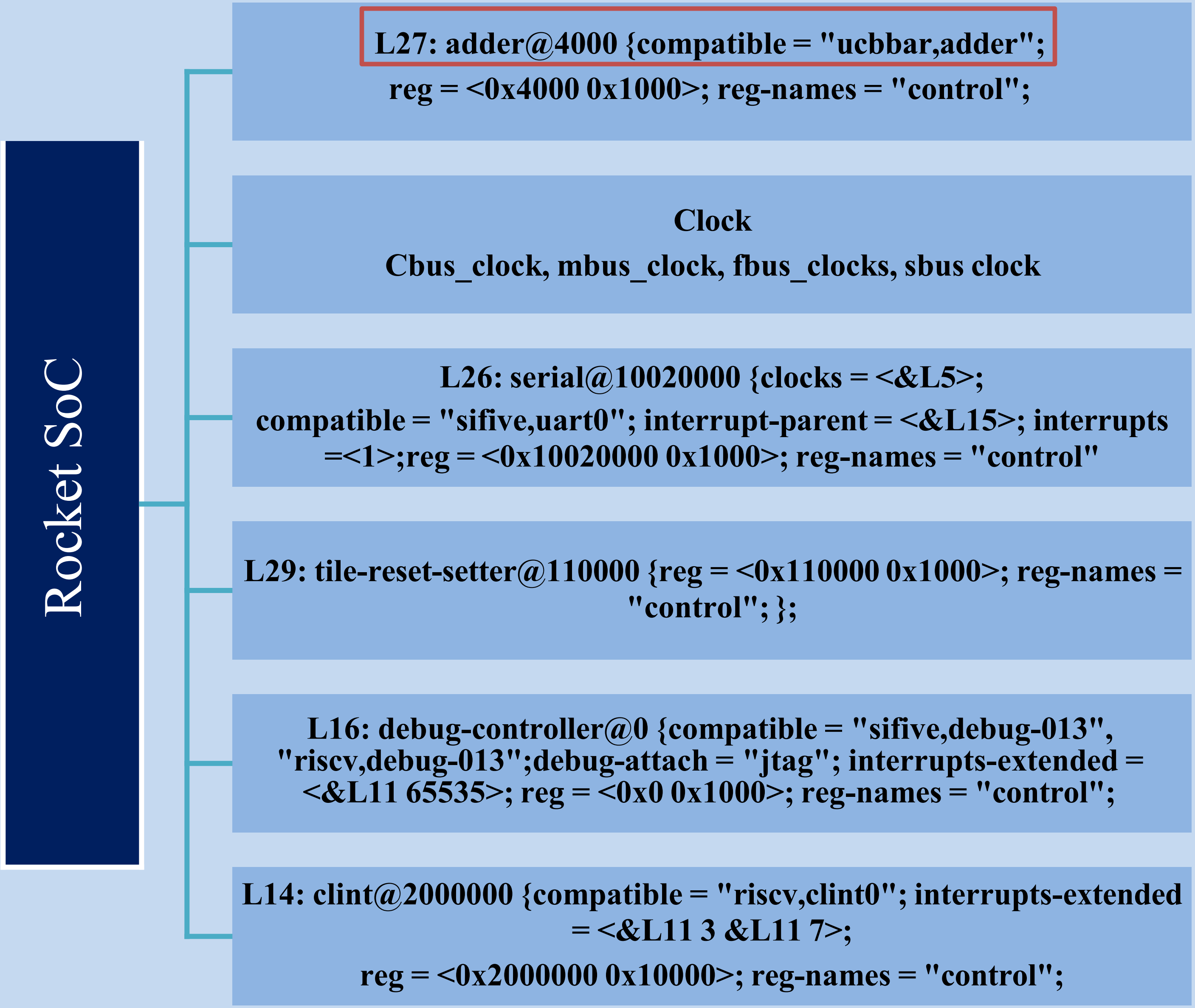
ranges;

htif

compatible = "ucb,htif0";

Device Tree Source

SoC Node



Rocket SoC

L27: adder@4000 {compatible = "ucbbar,adder";
reg = <0x4000 0x1000>; reg-names = "control";

Clock
Cbus_clock, mbus_clock, fbus_clocks, sbus clock

L26: serial@10020000 {clocks = <&L5>;
compatible = "sifive,uart0"; interrupt-parent = <&L15>; interrupts =<1>;reg = <0x10020000 0x1000>; reg-names = "control"

L29: tile-reset-setter@110000 {reg = <0x110000 0x1000>; reg-names = "control"; };

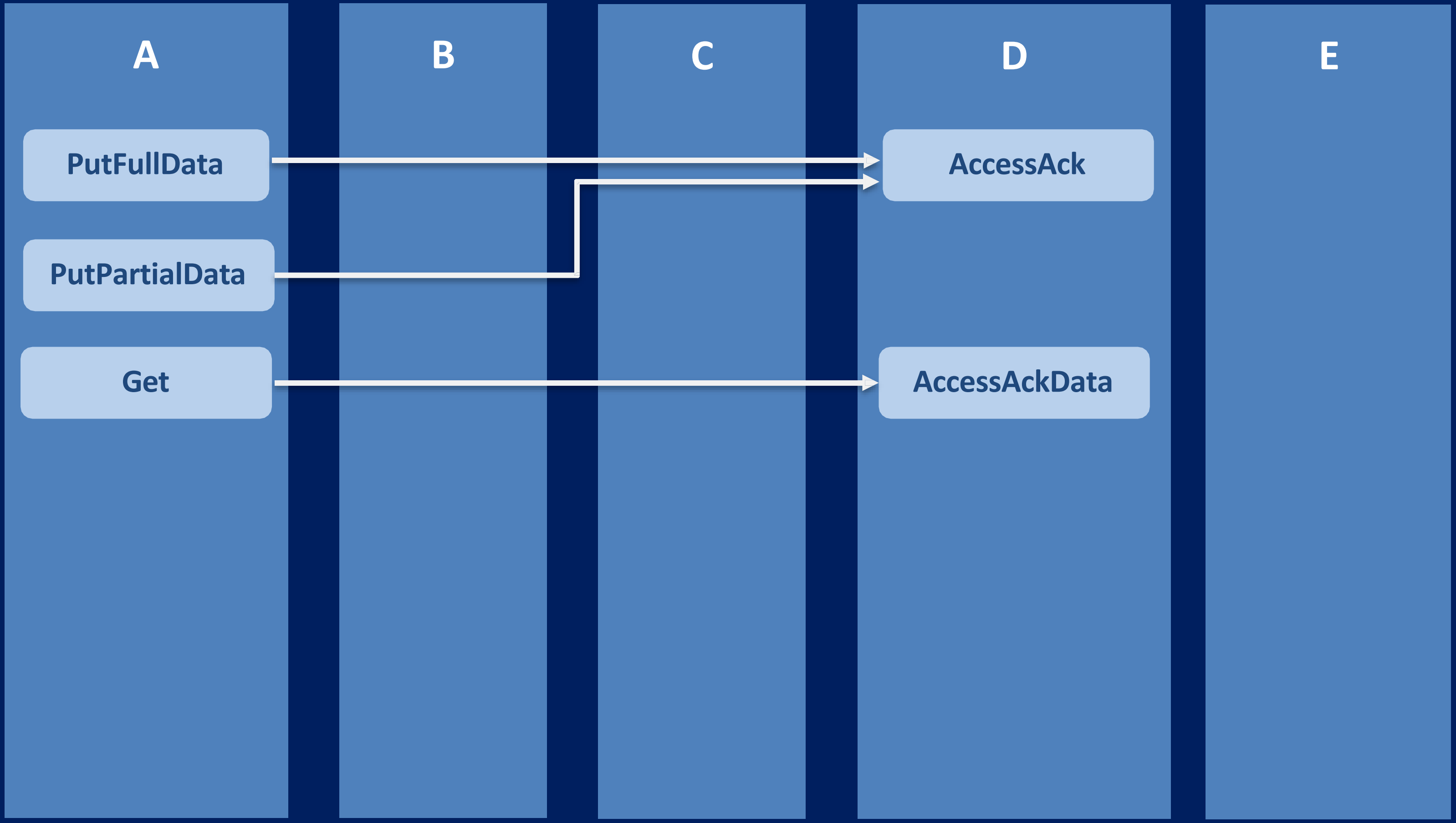
L16: debug-controller@0 {compatible = "sifive,debug-013", "riscv,debug-013";debug-attach = "jtag"; interrupts-extended = <&L11 65535>; reg = <0x0 0x1000>; reg-names = "control";

L14: clint@2000000 {compatible = "riscv,clint0"; interrupts-extended = <&L11 3 &L11 7>;
reg = <0x2000000 0x10000>; reg-names = "control";

Tilelink Uncached Lightweight

Channel	Signal	Description
A	a_valid, a_ready, a_opcode, a_size, a_address, a_data	Used for basic read and write requests.
D	d_valid, d_ready, d_opcode, d_size, d_data	Used for response data from slave to master.

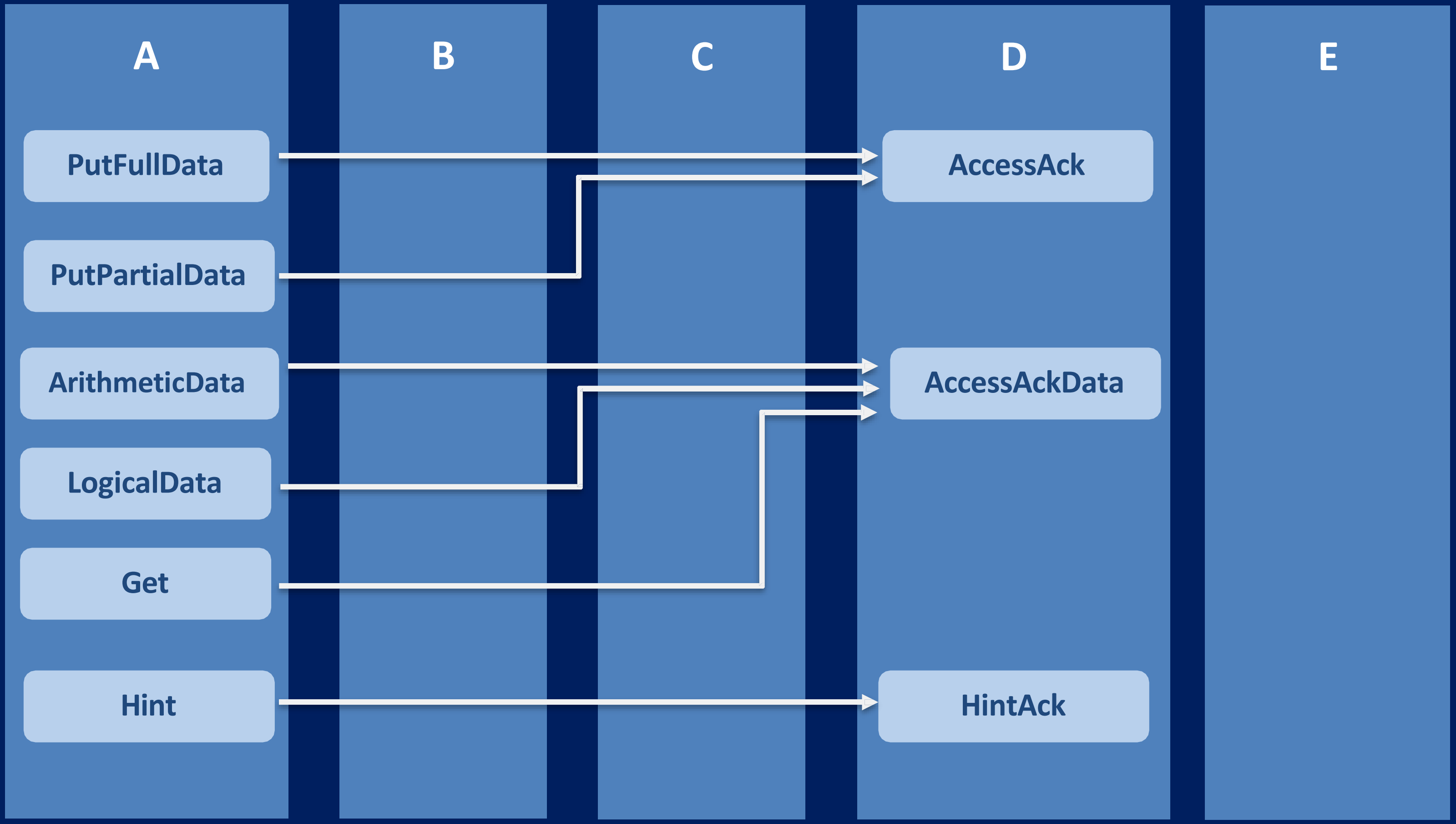
Tilelink Uncached Lightweight Messages



Tilelink Uncached Heavyweight

Channel	Signal	Description
A	a_valid, a_ready, a_opcode, a_param, a_size, a_address, a_data, a_mask	Used for more complex read/write requests, including partial writes.
C	c_valid, c_ready, c_opcode, c_param, c_size, c_data	Used for voluntary releases and writebacks, typically in larger systems.
D	d_valid, d_ready, d_opcode, d_param, d_size, d_data, d_denied	Used for response data, including error signals for access denial.

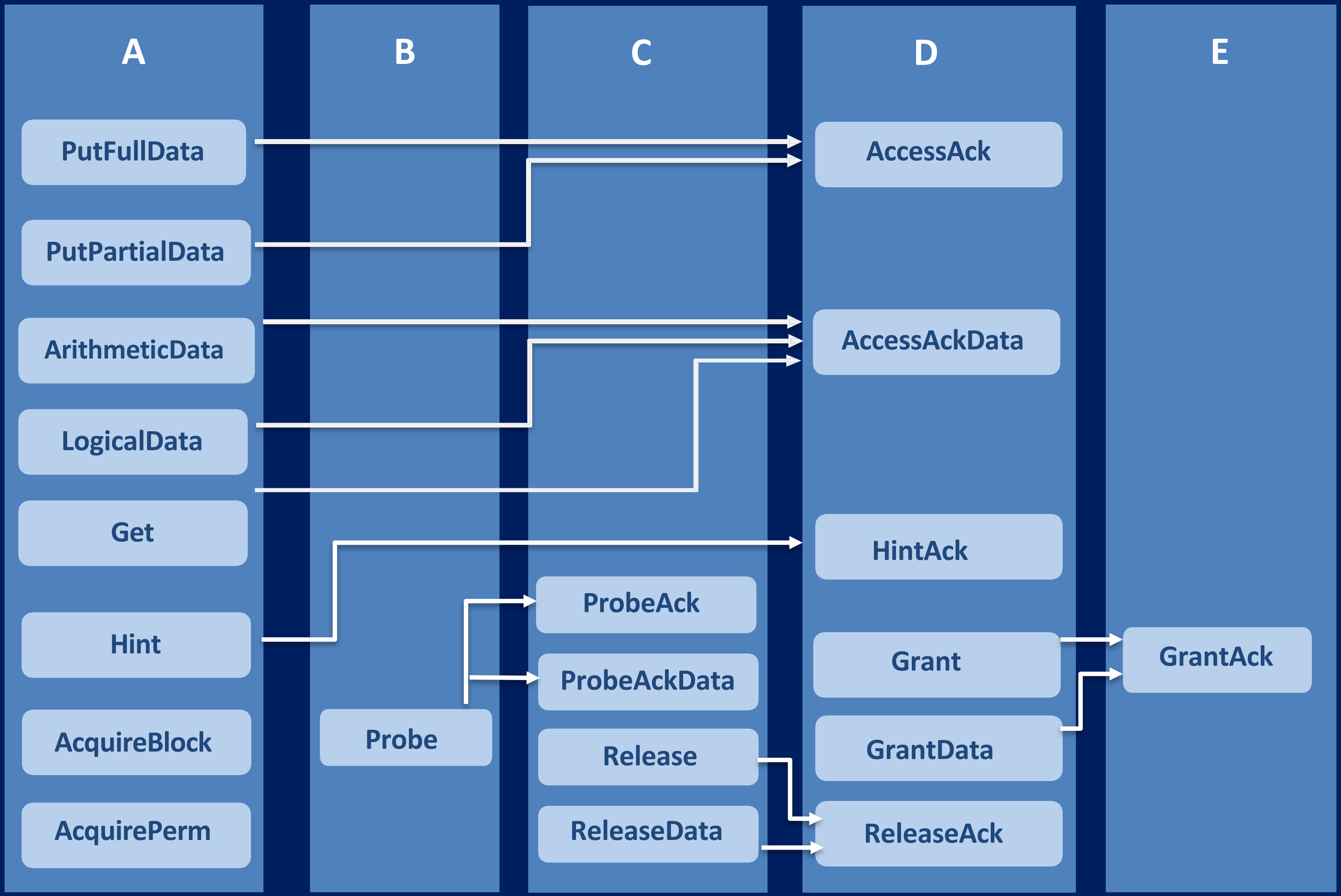
Tilelink Uncached Heavyweight Messages



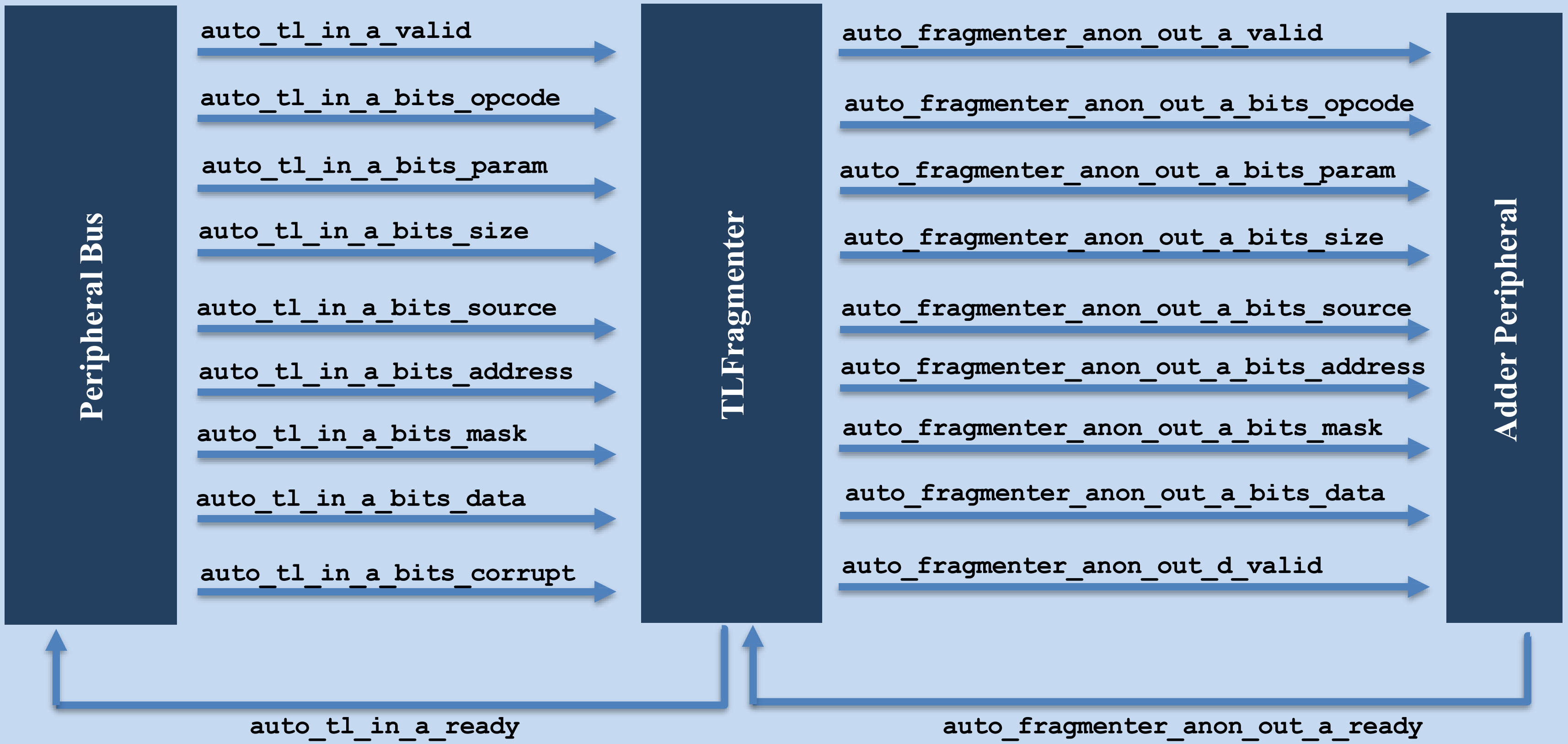
Tilelink Cached.

Channel	Signal	Description
A	a_valid, a_ready, a_opcode, a_param, a_size, a_address, a_data, a_mask, a_source	Supports all request types, including cache coherence and atomic operations.
B	b_valid, b_ready, b_opcode, b_param, b_size, b_address, b_source, b_mask	Used for probes, enabling the slave to check and invalidate caches if needed.
C	c_valid, c_ready, c_opcode, c_param, c_size, c_address, c_data, c_source	Used for release operations, allowing caches to voluntarily evict or write back data.
D	d_valid, d_ready, d_opcode, d_param, d_size, d_data, d_source, d_sink, d_denied, d_corrupt	Used for responses to requests, including error and coherence status information.
E	e_valid, e_ready, e_sink	Used for grant acknowledgments to ensure coherence.

Tilelink Cached Messages



REQUEST – CHANNEL A



Peripheral Bus

auto_tl_in_a_valid

auto_tl_in_a_bits_opcode

auto_tl_in_a_bits_param

auto_tl_in_a_bits_size

auto_tl_in_a_bits_source

auto_tl_in_a_bits_address

auto_tl_in_a_bits_mask

auto_tl_in_a_bits_data

auto_tl_in_a_bits_corrupt

TLFragmenter

auto_fragmenter_anon_out_a_valid

auto_fragmenter_anon_out_a_bits_opcode

auto_fragmenter_anon_out_a_bits_param

auto_fragmenter_anon_out_a_bits_size

auto_fragmenter_anon_out_a_bits_source

auto_fragmenter_anon_out_a_bits_address

auto_fragmenter_anon_out_a_bits_mask

auto_fragmenter_anon_out_a_bits_data

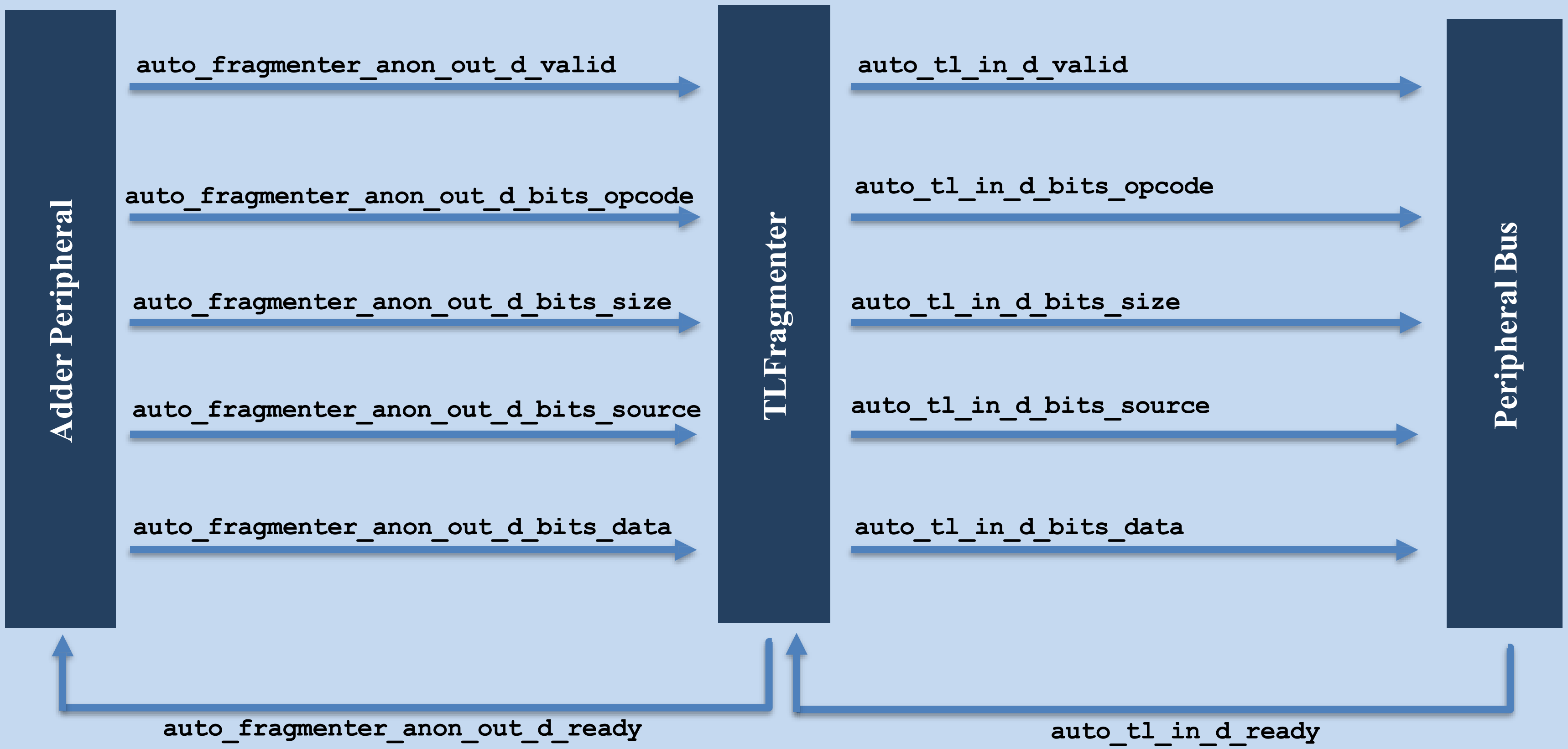
auto_fragmenter_anon_out_d_valid

Adder Peripheral

auto_tl_in_a_ready

auto_fragmenter_anon_out_a_ready

RESPONSE – CHANNEL D



Adder Peripheral

TLFragmenter

Peripheral Bus

auto_fragmenter_anon_out_d_valid

auto_fragmenter_anon_out_d_bits_opcode

auto_fragmenter_anon_out_d_bits_size

auto_fragmenter_anon_out_d_bits_source

auto_fragmenter_anon_out_d_bits_data

auto_tl_in_d_valid

auto_tl_in_d_bits_opcode

auto_tl_in_d_bits_size

auto_tl_in_d_bits_source

auto_tl_in_d_bits_data

auto_fragmenter_anon_out_d_ready

auto_tl_in_d_ready