ESWIN

E-Trace Encoder 设计说明

E-Trace Encoder Design Specification

Rev 1.0

2024/05/23

版权声明

Copyright

许可协议声明

License

免责声明

Disclaimer

修订记录：

Revision History：

文档版本 Revision	日期 Date	说明 Description
Rev 1.0	2025/05/23	初始版本 Initial version

目 录
Contents

表目录

Tables

图目录

Figures

1. E-Trace Encoder 系统方案 E-Trace Encoder System Scheme

1.1 E-Trace Encoder 系统方案介绍 E-Trace Encoder System Scheme Introduce

高效追踪编码器（Efficient Trace Encoder，E-Trace Encoder）整体由接口（Interface）、编码器（Encoder）两部分组成。

The E-Trace Encoder consists of two parts: Interface and Encoder.

接口：从核（Core）中引出相关的信号到 Encoder 中来实现对指令的跟踪（Trace），每组核（Core）需要引出三组接口（Interface）。

Interface: Introduce relevant signals from the Core to the Encoder to implement Trace for instructions, and each Core needs to introduce three sets of interfaces.

编码器：根据接口（Interface）中信号类型以及指令行为进行数据组包，通过 ATB 协议对数据包进行传输。

Encoder: Packets data based on signal type and instruction behavior in the interface, transmits data packets through the ATB protocol.

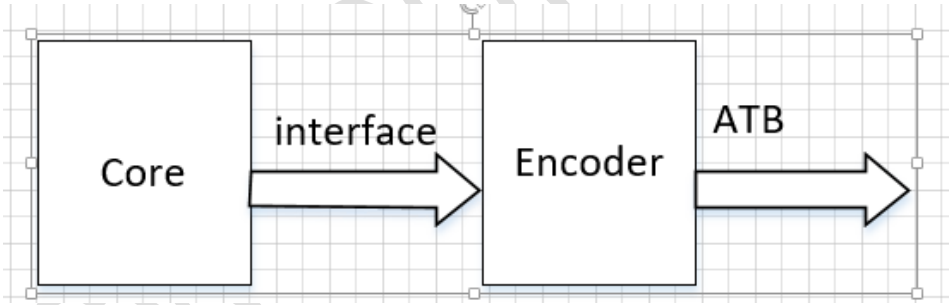

图 1-1 E-Trace Encoder 整体方案框图

Figure 1-1 Overall scheme diagram of E-Trace Encoder

1.2 编码器系统接口 Encoder System Interface

表 1-1 E-Trace Encoder 接口信号

Table 1-1 E-TRACE Encoder Interface Signal

信号 Singal	描述 Description
APB interface	
tr_pclk_i	APB slave: clock
tr_prstn_i	APB slave : reset
tr_psel_i[4:0]	APB slave : select

信号 Singal	描述 Description
tr_penable_i	APB slave : enable
tr_pwrite_i	APB slave : write
tr_paddr_i[31:0]	APB slave : address
tr_pwdata_i[31:0]	APB slave : write data
tr_prdata_o[31:0][4:0]	APB slave : read data
tr_pready_o[4:0]	APB slave : ready
tr_pslverr_o[4:0]	APB slave : error response
ATB interface	
tr_atclk_i	Global Signal:Global ATB clock
tr_atresetn_i	Global Signal:ATB interface reset
tr_atbytes_o[3:0]	Data Signal:ATB master : number of bytes on ATDATA
tr_atdata_o[127:0]	Data Signal:ATB master: trace data
tr_atid_o[6:0]	Data Signal:ATB master : ID
tr_atvalid_o	Data Signal:ATB master : transfer is valid
tr_atready_i	Data Signal:ATB master : slave ready
tr_afready_o	Flush Signal:ATB master : flush acknowledge
tr_afvalid_i	Flush Signal:ATB master : flush sigal
tr_atclk_i	Global Signal:Global ATB clock
tr_atresetn_i	Global Signal:ATB interface reset
tr_atbytes_o[2:0]	Data Signal:ATB master : number of bytes on ATDATA
Core signal	
Inter_uopx_x_i	Interface between encoder and core
Trenc_async_expt_vld_i[3:0]	Asynchronous exception valid
Trenc_core_id_i[2:0][4]	Core id
Trenc_clk_i	Cpu clk
Trenc_itime_i	Timestamp value

2. E-Trace Encoder 接口 E-Trace Encoder Interface

跟踪接口包括以下几类：（1）核（Core）和编码器（Encoder）之间的接口；（2）编码器（Encoder）数据传输的 ATB 协议；（3）寄存器读写的 APB 协议。

Trace Interface includes the following categories: (1) Interface between Core and Encoder; (2) ATB protocol for Encoder data transmission; (3) APB protocol for register read and write.

2.1 CPU 和追踪编码器间的接口 Interface between CPU and Encoder

表 2-1 CPU 和编码器间的接口信号

Table 2-1 Interface Signal between CPU and Encoder

信号 Singal	描述 Description	重置 Reset
Trenc_x_Retn_i	Async reset, low active	
Trenc_x_Clk_i	Clock input.	

信号 Singal	描述 Description	重置 Reset
Trenc_x_Iaddr_i[38:0]	The address of the instruction retired in this block.	0
Trenc_x_Itype[3:0]	Type of instruction	0
Trenc_x_Time[15:0]	Time Generator by the core	0
Trenc_x_Priv_i[1:0]	Privilege level for all instructions retired on this cycle	0
Trenc_x_Cause_i[1:0]	Exception or interrupt cause (ucause/scause/mcause)	0
Trenc_x_Tval_i[39:0]	Epc	0
Trenc_x _Cont_i[19:0]	Domain ID&ASID	0
Trenc_x _core_id_i[1:0]	Core id	0
Trenc_x _iretire[2:0]	Number of halfwords represented by instructions retired in this block.	0
Trenc_x _ilastsize	The size of the last retired instruction is $2^{ilastsize}$ half-words	0
Trenc_x_inst_num	Number of instruction in this entry	0
Trenc_async_expt_vld	Asynchronous exception valid	0

*x= 0，1，2；

2.2 ATB 接口 ATB Interface

表 2-2 ATB 接口信号
Table 2-2 ATB Interface Signal

信号 Singal	描述 Description
Trenc_atclk_i	Global Signal:Global ATB clock
Trenc_atresetn_i	Global Signal:ATB interface reset
Trenc_atbytes_o[2:0]	Data Signal:ATB master : number of bytes on ATDATA
Trenc_atdata_o[63:0]	Data Signal:ATB master: trace data
Trenc_atid_o[6:0]	Data Signal:ATB master : ID
Trenc_atvalid_o	Data Signal:ATB master : transfer is valid
Trenc_atready_i	Data Signal:ATB master : slave ready
Trenc_afready_o	Flush Signal:ATB master : flush acknowledge
Trenc_afvalid_i	Flush Signal:ATB master : flush sigal

2.3 APB 接口 APB Interface

表 2-3 APB 接口信号
Table 2-3 APB Interface Signal

信号 Singal	描述 Description
Trenc_x _pclk_i	APB slave: clock
Trenc_x _prstn_i	APB slave : reset
Trenc_x _psel_i	APB slave : select
Trenc_x _penable_i	APB slave : enable

信号 Singal	描述 Description
Trenc_x _pwrite_i	APB slave : write
Trenc_x _paddr_i[32:0]	APB slave : address
Trenc_x _pwdata_i[32:0]	APB slave : write data
Trenc_x _prdata_o[32:0]	APB slave : read data
Trenc_x _pready_o	APB slave : ready
Trenc_x _pslverr_o	APB slave : error response

3. 编码器介绍 Encoder Introduce

编码器用于对核上执行的指令进行编码。如下图所示，编码器主要由 5 部分构成：(1) 寄存器映射；(2) 跟踪 Payload 生成器；(3) 跟踪包生成器；(4) 数据包缓冲区；(5) ATB 总线。

Encoder is used to encode instructions executed on the Core. As shown in the following figure, Encoder mainly consists of 5 parts: (1) Register map; (2) Trace Payload Generator; (3) Trace Packet Generator (4) Packet buffer; (5) ATB (Advanced Trace Bus).

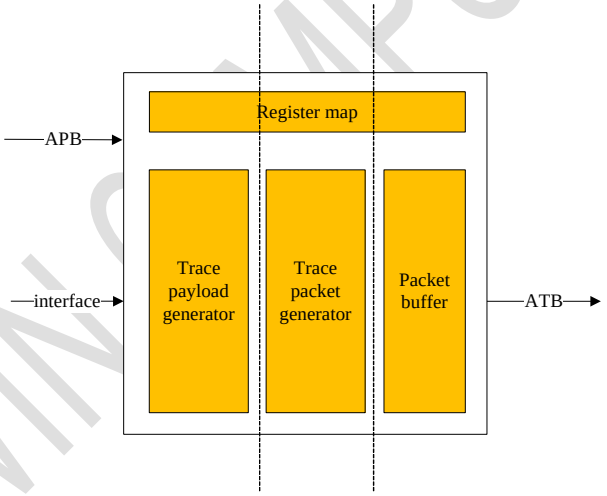

图 3-1 编码器整体方案框图

Figure 3-1 Encoder Overall Scheme Diagram

3.1 寄存器映射 Register Map

寄存器映射实现对编码器的控制，参考推荐的寄存器映射[1]并且结合实际实现功能，确定如下的寄存器功能。

Register Map is used to control the Encoder. Referring to the recommended Register Map and combining it with actual implementation functions, the following register functions are determined.

[1] https://github.com/riscv-non-isa/tg-nexus-trace/blob/master/docs/RISC-V-Trace-Control-Interface.adoc#register-map

表 3-1 编码器寄存器列表

Table 3-1 Encoder Register List

地址偏移量 Address offset	跟踪编码器 Trace Encoder	描述 Description
0x000	teControl	[31:0] TE Control Register
0x004	teImpl	[31:0] TE Implementation informaion
0x008	teInstFeature	[31:0] Extra instruction trace encoder features
0x040	tsControl	[31:0] Timestamp control register
0x41c	teTime	[31:0] Expire Time

注意：

Note：

i. 仅当禁用跟踪时，才允许写入寄存器。

Only when Trace disabled，write register allowed.

（1） 0x000 teControl 跟踪编码器控制寄存器

0x000 teControl Trace Encoder Control Register

表 3-2 teControl 寄存器

Table 3-2 teControl Register

位 Bits	字段 Field	描述 Description	读写 RW	重置 Reset
0	teActive	Master enable for given TE. 0 resets the TE When teActive=0,	RW	0
1	teEnable	1=TE enabled. Allows teTracing to turn all tracing on and off.	RW	0
2	teTracing	1 trace is being generated.	R	0
3	teEmpty	1 trace can transmission data which generate	R	0
6-4	teInstMode	Main instruction trace generation mode 0 = Branch trace is support Other Reserve	RW	0
15-7		Reserve		0
17-16	teSyncMode	Select periodic synchronization mechanism. At least one non-zero mechanism must be implemented. 0 = Reserve 1 = Reserve 2 = Count clock cycles && retire instruction valid 3 = Reserve	R	0
23-18	Reserved			
26-24	teFormat	Trace recording format 0 = Format defined by E-Trace Specification 1-7= Reserved for future formats	R	0
31-27	teSink	Which sink to send trace to. others = Reserved 5 = ATB Sink	R	0

（2） 0x004 teImpl 跟踪编码器实现寄存器

0x004 teImpl Trace Encoder Implementation Register

表 3-3 teImpl 寄存器

Table 3-3 telmpl Register

位 Bits	字段 Field	描述 Description	读写 RW	重置 Reset
3-0	teVersion	1 = Current Version.Control interface version	R	0
4	hasSRAMSink	1 if this TE has an on-chip SRAM sink. 0 if not（this version do not has SRAM sink）	R	0
5	hasATBSink	1 if this TE has an ATB sink. 0 if not（this version has ATB sink）	R	0
6	hasPIBSink	1 if this TE has an off-chip trace port via a Pin Interface Block (PIB). 0 if not（this version do not has PIB sink）	R	0
7	hasSBASink	1 if this TE has an on-chip system memory bus master trace sink. （this version do not has SBA sink）	R	0
8	hasFunnelSink	1 if this TE feeds into a trace funnel device.（this version has Funnel sink）	R	0
19-9		Reserved for future sink types	R	0
23-20	teSrcID	This TE's source ID. （this version value is 4'b1111）	R	0
31-24		Reserved		0

（3） 0x008 telnstfeature 功能寄存器

0x008 telnstfeature function register

表 3-4 telnst 寄存器

Table 3-4 telnst Register

位 Bits	字段 Field	描述 Description	读写 RW	重置 Reset
0	telnstNoAddrDiff	Reserve		0
1	telnstNoExceptAdd	Reserve		0
2	telnstEnaSequentialJump	Reserve		0
3	telnstEnaImplicitReturn	Reserve		0
4	telnstEnaBranchPrediction	Reserve		0
5	telnstEnaJumpTargetCach	Reserve		0

（4） 0x040 teControl 时间戳控制寄存器

0x040 teControl timestamp control register

表 3-5 teControl 寄存器

Table 3-5 teControl Register

位 Bits	字段 Field	描述 Description	读写 RW	重置 Reset
0	tsActive	Enable for timestamp unit	RW	0

位 Bits	字段 Field	描述 Description	读写 RW	重置 Reset
31-1	tsWidth	Reserve		

（5） 0x41c teTime 终止时间寄存器

0x41c teTime termination time register

表 3-6 teTime 寄存器

Table 3-6 teTime Register

位 Bits	字段 Field	描述 Description	读写 RW	重置 Reset
0-15	Tetime	Expire time value	RW	0
31-16	tsWidth	Reserve		

3.2 跟踪 Payload 生成 Trace Payload Generator

跟踪编码输出数据包，基本的数据包格式由表示 Payload 长度的帧头、Core ID、时间戳标志位、时间戳以及 Payload 组成。

The E-Trace Encoder outputs a packet, and the basic data frame format consists of a frame header representing the Payload length, Core ID, timestamp flags, timestamp, and Payload.

3.2.1 Payload 数据格式 Payload Data Format

E-Trace Encoder 负载支持三种数据格式，分别为：Format3、Format2、Format1，以下为 Payload 的三种具体格式内容。

The E-Trace Encoder Payload supports three data formats: Format3 Format2、Format1. The following are three specific formats of Payload content.

3.2.1.1 格式 3 Format3

（1） 格式 3 子格式 0（Format3 Subformat 0）：同步数据包。发送条件：a）第一条 Trace 指令；b）Resynchronisation 之后。

Format3 Subformat 0: synchronize packets. Sending condition: a) the first Trace instruction; b) after Resynchronization.

表 3-7 格式 3 子格式 0

Table 3-7 Format3 Subformat0

格式 3 子格式 0 同步 Format 3 Subformat0 Synchronisation		
域名 Field Name	**位 Bits**	**描述 Description**
Format	2	11(sync)：synchronisation.
subformat	2	00(start)：start of tracing or resync.
Branch	1	0: instr is branch and branch was taken.1:instr is not branch or instr is branch but not taken.
privilege	2	The privilege of instr.
Time	16	The time value from uncore.
context	20	The domain id[3:0] & asid[19:4]
address	39	The address if instr. This address has been left shift.

（2） 格式 3 子格式 1（Format3 Subformat1）：异常数据包。发送条件：在指令异常之后发送该数据包。

Format3 Subformat1: exception packet. Sending condition: send the data packet after the instruction exception.

表 3-8 格式 3 子格式 1
Table 3-8 Format3 Subformat1

格式 3 子格式 1 异常 Format3 Subformat1 Trap		
域名 Field Name	**位 Bits**	**描述 Description**
format	2	11(sync)：synchronization
subformat	2	01(trap)：Exception or interrupt cause and trap handler address.
branch	1	0:instr is a branch and branch was taken. 1:instr is not a branch or instruction is a branch but not taken
privilege	2	The privilege level of instr
time	16	The time value.
context	20	The domain id[3:0] & asid[19:4]
ecause	6	The exception or interrupt cause
interrupt	1	Interrupt
thaddr	1	1：address point to trap handler. 0: address point to address
address	39	The address of instr. This address has been left shift.
tval	40	EPC

（3） 格式 3 子格式 3（Format3 Subformat3）：Support 数据包。发送条件：a）Trace 使能或者无效时；b）操作模式发生改变时；c）Packet 发送失败时，例如发生反压时。

Format3 Subformat3: support packet. Sending condition: a) when Trace is enabled or invalid; b) when the operating mode changes; c) when the packet fails to be sent, such as when backpressure occurs.

表 3-9 格式 3 子格式 3
Table 3-9 Format3 Subformat3

格式 3 子格式 3 支持 Format3 Subformat3 Support		
域名 Field Name	位 Bits	描述 Description
format	2	11(sync)：synchronization
subformat	2	11(support)：Supporting information for the decoder
ienable	1	Indicate trace encoder is enable
encoder_mode	1	0: support branch trace
qual_status	2	00:no change 01:trace disable. 10:trace packets lost 11:not used
ioptions	2	10:full address mode

3.2.1.2 格式 2 Format 2

发送条件：指令的地址需要被上报的时候。

Sending condition: when the address of the instruction needs to be reported.

表 3-10 格式 2
Table 3-10 Format2

格式 2 Format2		
域名 Field Name	位 Bits	描述 Description
format	2	10
address	39	The address of instr. This address has been left shift.
notify	1	If the value of this bit is different from the MSB of address, it indicates that this packet is reporting an instruction that is not the target of an un-inferable discontinuity.
updiscon	1	If the value of this bit is different from notify, it indicates that this packet is reporting the instruction following an un-inferable discontinuity and is also the instruction is marked an async_exception or interrupt。

3.2.1.3 格式 1 Format 1

（1） 格式 1：地址，分支列表（Format 1：Address，Branch Map）。发送条件：a）分支信息需要被上报；b）指令的地址需要被上报；c）自从上一个数据包以来至少存在一个分支指令。

Format 1: Address, Branch map. Sending conditions: a) branch information needs to be reported; b) the address of the instruction needs to be reported; c) there has been at least one branch instruction since the previous packet.

表 3-11 带地址的 Format1

Table 3-11 Format1 with Address

格式 1 带有地址的分支列表 Format1 Branch Map with Address		
域名 Field Name	位 Bits	描述 Description
format	2	01
branches	5	Number of branches in history(0 cannot occur in this format).
branch_map	31	An array of bits indicating whether branches are taken or not. 0：branch taken 1：branch not taken
address	39	The address of instr. This address has been left shift.
notify	1	If the value of this bit is different from the MSB of address, it indicates that this packet is reporting an instruction that is not the target of an un-inferable discontinuity.
updiscon	1	If the value of this bit is different from notify, it indicates that this packet is reporting the instruction following an un-inferable discontinuity or is also the instruction is marked an async_exception or interrupt

（2） 格式 1：无地址，分支列表（Format 1：No Address，Branch Map）。发送条件：a）分支信息需要被上报；b）指令的地址需要被上报；c）自从上一个数据包以来至少存在一个分支指令。格式 0 子格式 0（Format0 Subformat0）：不支持。

Format 1: No Address, Branch map. Sending condition: a) branch information needs to be reported; b) the address of the instruction needs to be reported; c) there has been at least one branch instruction since the previous packet. Format0 Subformat0: not supported.

表 3-12 不带地址的格式 1

Table 3-12 Format1 with no Address

格式 1 不带地址的分支列表 Format1 Branch Map with no Address		
域名 Field Name	位 Bits	描述 Description
format	2	01
branches	5	0
branch_map	31	An array of bit indicating whether branches are taken or not. 0：branch taken 1：branch not taken

3.2.2 编码算法实现 Implementation of Encoding Algorithm

3.2.2.1 E-Trace Encoder 算法设计 E-Trace Encoder Algorithm Design

E-Trace Encoder 算法实现如下图所示。

The implementation of the E-Trace Encoder algorithm is shown in the following figure.

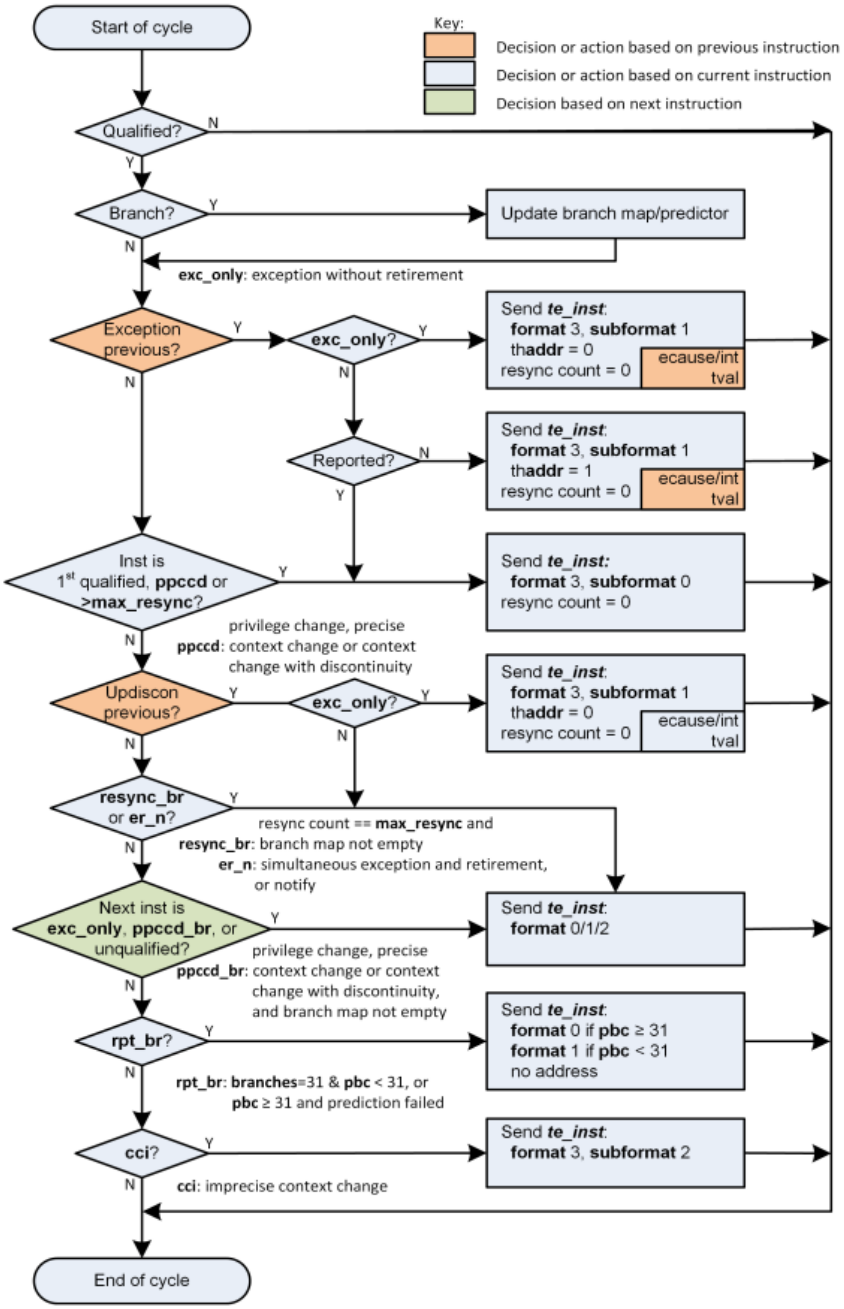

图 3-2 数据包格式选择算法

Figure 3-2 Packet Format Selection Algorithm

3.2.2.2 设计方案 Design Scheme

（1） 参考算法设定的场景一个周期退休一条指令，需要当前（Current）指令的下一条（Next）指令和上一条（Previous）指令来确定发送的数据包格式，E-Trace Encoder 里面采用和推荐算法相同实现的流程。

In the scenario set by the reference algorithm, one cycle retires one instruction, and the Next and Previous instructions of the Current instruction are required to determine the format of the sent data packet. The E-Trace Encoder adopts the same implementation process as the recommendation algorithm.

（2） 同时根据每一个周期中退休的有效指令数来实现对算法槽中有效指令的判断。

Simultaneously, based on the number of valid instructions retired in each cycle, the judgment of valid instructions in the algorithm slot is achieved.

（3） 对于第一条指令（First Instruction），根据 Spec 协议会发送启动（Strat）数据包，上报第一条指令的地址。

For the first instruction, a start packet will be sent according to the spec protocol, reporting the address of the first instruction.

（4） 对于最后一条指令（Last Instruction），根据 Spec 协议会发送格式 2（Format2）或者格式 1（Format1）数据包，报出最后一条指令的地址。

For the last instruction, according to the spec protocol, format2 or format1 packets will be sent, reporting the address of the last instruction.

（5） 对于正常退休的异常指令，会上报该指令的地址。

For abnormal instructions for normal retirement, the address of the instruction will be reported.

（6） 如果由于下游 Trace 组件 Atready 长时间未准备好或者退休指令短时间发出大量数据包等原因导致 E-Trace Encoder 中缓冲区（Buffer）发生“写满”的情况，E-Trace Encoder 中会清空缓存（Buffer）以及流水线（Pipeline），并且在清空之后发送支持（Support）以及启动（Strat）数据包，该启动（Strat）数据包的地址为“当前（Current）移位槽”中第一个有效指令。

If the buffer in the E-Trace Encoder becomes "write full" due to reasons such as the downstream Trace component Atrady not being ready for a long time or retirement instructions sending a large number of packets for a short period of time, the E-Trace Encoder will clear the buffer and pipeline, and send support and start packets after clearing. The address of the start packet is the first valid instruction in the "Current shift slot".

（7） 如果在关闭跟踪之后出现溢出，此时最后一个数据包为指示发生溢出的支持数据包，如果跟踪溢出和关闭跟踪同时发生，可能会出现“支持（Pkt_lost）+支持（Trace_disable）”类似的数据包组合形式。

If an overflow occurs after closing trace, the last packet is the support packet indicating the overflow. If trace overflow and closing trace occur simultaneously, a combination of packets similar to support “(pktlost)+support (trace_disable)” may occur.

（8） 在调试（Debug）模式下不会对指令进行 Trace。

In debug mode, no trace will be applied to instructions.

3.2.2.3 算法实现框图 Algorithm Implementation Block Diagram

如下图所示：使用 6 个移位槽作为算法实现的主体结构。其中移位槽 0 代“下一条指令（Next Instruction）”，移位槽 1，2，3，4 表示“当前指令（Current Instruction）”，移位槽 5 表示“上一条指令（Previous Instruction）”。对移位槽 1，2，3，4 中的每条指令进行算法判断，根据算法得到该指令需要发送的数据包。同时根据每一周期进入跟踪系统的有效指令来决定指令在移位槽中的状态以及移位槽中有效的指令数。

As shown in the figure below, 6 shift slots are used as the main structure of the algorithm implementation. Shift slot 0 represents "Next instruction", shift slots 1, 2, 3, and 4 represent "Current instruction", and shift slot 5 represents "Previous instruction". Perform algorithm judgment on each instruction in shift slots 1, 2, 3, and 4, and obtain the data packet that needs to be sent for that instruction based on the algorithm. At the same time, the state of the instruction in the shift slot and the number of valid instructions in the shift slot are determined based on the valid instructions entering the Trace system in each cycle.

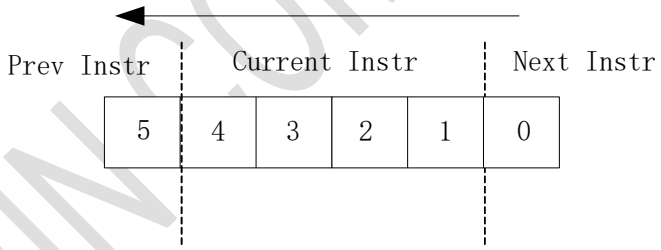

图 3-3 数据包选择算法实现

Figure 3-3 Implementation of Packet Selection Algorithm

如第一周期（CLK1）有效指令数为 3 条指令（a，b，c）第二周期（CLK2）有效指令数分别为 4，3，2，1，0 时，指令在“移位槽”中的存在如下图所示：

When the number of valid instructions in the first cycle (CLK1) is 3 (a, b, c) and the number of valid instructions in the second cycle (CLK2) is 4, 3, 2, 1, and 0 respectively, the existence of instructions in the "shift slot" is shown in the following figure:

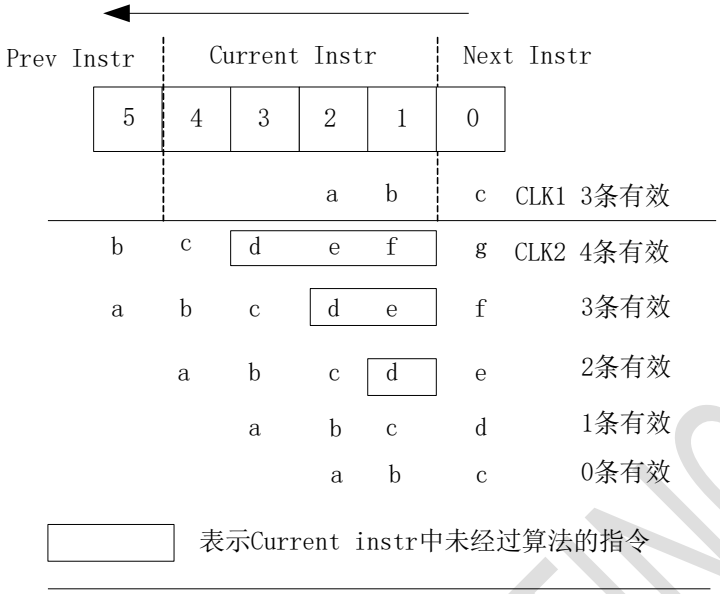

图 3-4 指令在“移位槽”中的移动情况

Figure 3-4 The Movement of Instructions in the "shift slot"

3.3 跟踪数据包生成器 Trace Packet Generator

跟踪数据包生成器主要完成对生成的 Payload 按照特定的格式进行数据打包形成数据包（Packet），如下图所示。每个数据包由包头、核编号（Core_id）、时间戳标志位、时间戳以及 Paylaod 组成，其中包头（5 bit）表示数据包中负载的长度；核编号（Core_id 2 bit）表示跟踪系统运行的核（Core）；时间戳标志位（1bit）表示数据包里面是否包含时间戳（可以由寄存器配置）；时间戳（16 bit）表示生成该数据包时对应的时间；Payload 表示指令经过编码算法之后生成的数据包。

The tracking packet generator mainly completes the data packaging of the generated Payload in a specific format to form a packet, as shown in the following figure. Each packet consists of a header, core_id, timestamp flag, timestamp, and Payload, where the header (5 bit) represents the length of the Payload in the packet; Core id (2bit) represents the core of the tracking system's operation; The timestamp flag bit (1 bit) indicates whether the packet contains a timestamp (which can be configured by registers); The timestamp (16 bit) represents the time corresponding to the generation of the data packet; Payload represents the data packet generated by an instruction after being encoded by an algorithm.

（1） 时间戳可以通过 tsControl 寄存器进行配置，若存在时间戳，则时间戳标志位为 1，若不存在时间戳，时间戳标志位为 0。

The timestamp can be configured through the tsControl register. If there is a timestamp, the timestamp flag is 1. If there is no timestamp, the timestamp flag is 0.

（2） Encoder 中第一条数据包为支持（Support）数据包，若存在时间戳，此时时间戳为 0。

The first data packet in the encoder is a support data packet, and if there is a timestamp, the timestamp is 0.

（3） Encoder 最后一条数据包为支持（Support）数据包，若存在时间戳，此时时间戳为 0。

The last data packet in the Encoder is a support data packet, and if there is a timestamp, the timestamp is 0.

（4） Trace 系统发生溢出后的第一个数据包为支持（Support）数据包，若存在时间戳，此时时间戳为 0。

The first packet after overflow in the trace system is the support packet, and if there is a timestamp, the timestamp is 0.

（5） Payload 补齐的形式：使用 Payload 的 LSB 位作为补齐位，补齐到 Byte 对齐。

The form of Payload completion: Use the LSB bit of Payload as the completion bit, and complete it to byte alignment.

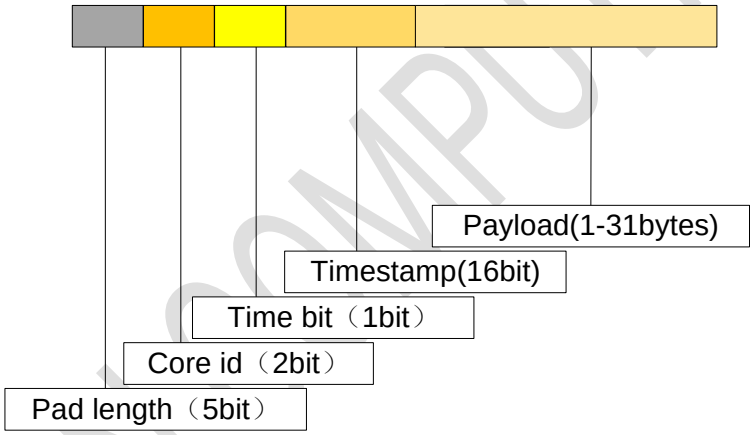

图 3-5 数据包组成形式

Figure 3-5 Packet Composition Form

3.4 数据包缓冲区 Packet Buffer

Trace 系统中对于每一个核（Core）外接独立的 E-Trace Encoder，其中每一个 E-Trace Encoder 使用独立的数据包缓冲区进行跟踪数据缓存。四个 E-Trace Encoder 中数据包缓冲区的设计相同。

In the Trace system, each Core is externally connected to an independent E-Trace Encoder, where each E-Trace Encoder uses an independent Packet Buffer for Trace data caching. The design of Packet Buffer is the same among the four E-Trace Encoders.

以程序中出现分支的概率为 1/4，每条分支发生跳转的概率为 1/2，以及综合实际 CPU 退休情况，设置缓冲区的最大深度为 8。根据最大数据包长度确定缓冲区写口宽度为 168 bit，FIFO（First in First out）读出数据口的最小宽度设置为 64 bit。FIFO 的基本参数如下：

The probability of branches appearing in the program is 1/4, the probability of Takens

occurring in each branch is 1/2, and considering the actual CPU retirement situation, the maximum depth of the buffer is set to 8. The width of the buffer write port is determined to be 168 bit based on the maximum packet length, and the minimum width of the FIFO read data port is set to 64 bit. The basic parameters of FIFO are as follows:

表 3-13 FIFO 参数
Table 3-13 FIFO Parameters

写数据位宽 Write Data Bit Width	读数据位宽 Read Data Bit Width	FIFO 深度 FIFO Depth
168 bit	64 bit	8

写数据位宽=最大数据包长度+可表示最大数据包长的 bit 位=最大 packet 的长度。

Write data bit width

= maximum packet length + bit that can represent the maximum packet length

= maximum packet length.

3.5 ATB 发送数据 ATB Send Data

ATB Send 完成读取数据包缓冲区中的数据，按照 ATB 协议发送至 ATB Upsizer 模块。基本时序如下图所示：

ATB Send completes reading the data from the Packet Buffer and sends it to the ATB Upsizer module according to the ATB protocol. The basic timing is shown in the following figure:

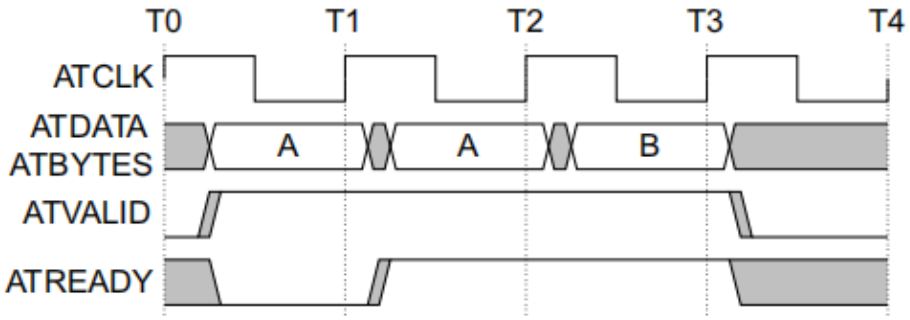

图 3-6 ATB 协议时序
Figure 3-6 ATB Protocol Timing

若数据包长度为 104 bit，需要两次握手才可将全部数据送出，第一次握手送出 64 bit，第二次握手送出 40 bit。

If the packet length is 104 bit, two handshakes are required to send all the data. The first handshake sends 64 bit, and the second handshake sends 40 bit.

在禁用跟踪系统后拉高 Afvalid，可完成读出跟踪（Trace）产生的所有数据。

After disabling the Trace system, raising the afvalid can complete the reading of all data generated by the Trace.

ATDATA：表示 E-Trace Encoder 输出的数据；ATBYTES：表示 ATDATA 的有效字节数。不同的 ATID 值表示不同的 Core。ATID 和 Core 之间的对应关系如下表：

ATDATA: represents the data output by the tracking encoder; ATBYTES: represents the number of valid bytes of ATDATA. Different ATID values represent different cores. The correspondence between ATID and Core is shown in the table below:

表 3-14 ATID 和 Core 之间的对应关系
Table 3-14 The Correspondence between ATID and Core

核编号 Core_ID	ATID
00	0F
01	1F
02	2F
03	3F

4. 编码器配置 Encoder Configure

通过打开/关闭 E-Trace Encoder 寄存器，来实现打开/关闭 trace 功能。以下为简单的示例。

By opening/closing the E-Trace Encoder register, the trace function can be turned on/off. The following is a simple example.

4.1 启动 E-Trace Encoder Enable E-Trace Encoder

配置流程：

Configuration process:

表 4-1 启动编码器流程
Table 4-1 Enable Encoder Process

基地址 Base Address	偏移地址 Offset address	配置值 Configuration Value
0x4001000	0x000	FFFF FF01
0x4001000	0x004	FFFF FFFF
0x4001000	0x040	FFFF FFFF
0x4001000	0x41c	须小于 FFFF FFFF
0x4001000	0x000	FFFF FF03

在所有寄存器配置完成之后，跟踪编码器才会开始工作。

Trace will only start working after all register configurations are completed.

4.2 禁用 E-Trace Encoder Disable E-Trace Encoder

配置流程：

Configuration process:

表 4-2 禁用编码器流程

Table 4-2 Disable Encoder Process

基地址 Base Address	偏移地址 Offset address	配置值 Configuration Value
0x4001000	0x000	FFFF FF00

5. 参考说明 Reference Specification

[1] ATB Protocol：AMBA ATB Protocol Specification[2].

[2] APB Protocol：AMBA APB Protocol Version:2.0[3].

[3] Trace Protocol：Efficient Trace for RISC-V Version 1.1.3-Frozen[4].

[2] https://developer.arm.com/documentation/ihi0032/latest/
[3] https://documentation-service.arm.com/static/60d5b505677cf7536a55c245?token=
[4] https://tools.cloudbear.ru/docs/riscv-trace-spec-1.1.3-20211102.pdf