AMBA AXI Stream

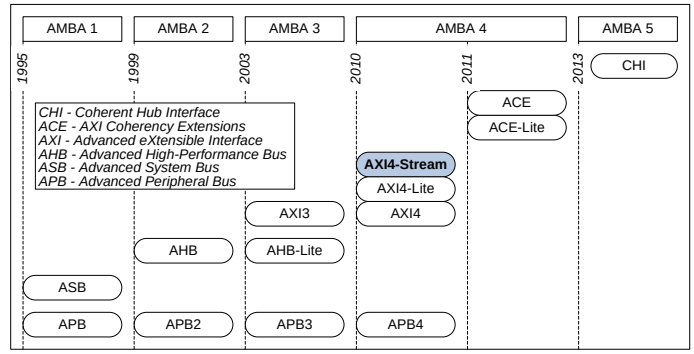
2015 – 2016 - 2017

Ando Ki, Ph.D.
(adki@future-ds.com)

Agenda

- Introduction
  - Evolution of AMBA Standards
  - AXI Interfaces
  - Memory mapped v.s. stream
  - Typical examples of stream interface
- AMBA AXI-Stream
  - VALID/READY handshake mechanism
  - AMBA AXI-Stream
  - Data streams
  - Interface signals
  - Data signaling
  - Byte qualifiers
- Example using TEA
  - Simple case
  - AXI stream case project

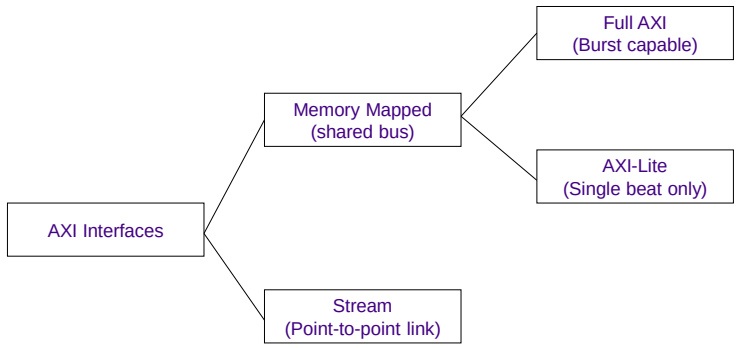
Evolution of AMBA Standards



- CHI - Coherent Hub Interface - The highest performance, used in networks and servers
- ACE - AXI Coherency Extensions - Used in big.LITTLE™ systems for smartphones, tablets, etc.
- AXI - Advanced eXtensible Interface - The most widespread AMBA interface. Connectivity up to 100's of Masters and Slaves in complex SoC's
- AHB - Advanced High-Performance Bus - The main system bus in microcontroller usage
- APB - Advanced Peripheral Bus - Minimal gate count for peripherals
- ATB - Advanced Trace Bus - For moving trace data around the chip

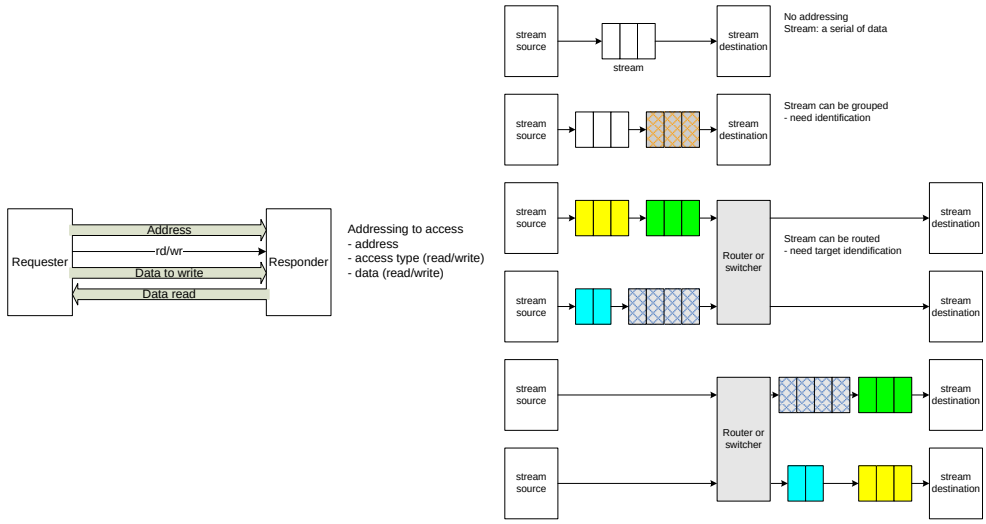
refer to “Ashley Stevens, Introduction to AMBA® 4 ACE™ and big.LITTLE™ Processing Technology, July 2013.”

AXI Interfaces

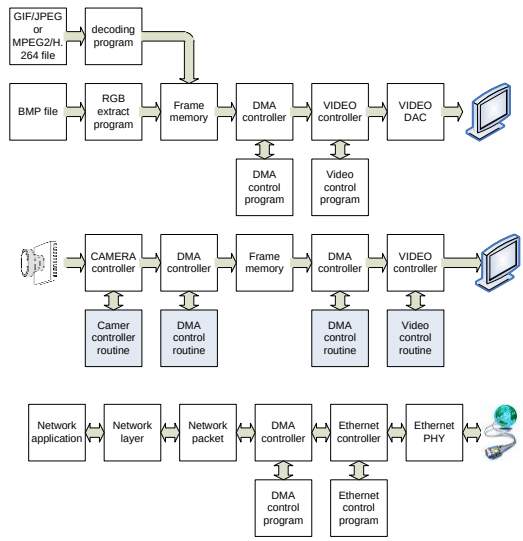


Interface	Features
Full AXI	Memory-mapped Traditional address/data burst
AXI-Lite	Memory-mapped Traditional address/data no-burst
Stream	Data-only burst

Memory mapped v.s. stream



Typical examples of stream interface

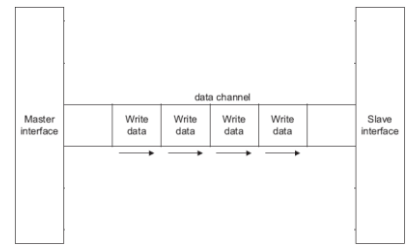


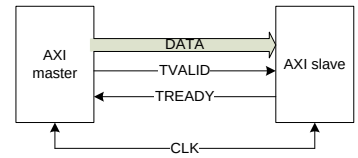
Agenda

- Introduction
  - Evolution of AMBA Standards
  - AXI Interfaces
  - Memory mapped v.s. stream
  - Typical examples of stream interface
- AMBA AXI-Stream
  - VALID/READY handshake mechanism
  - AMBA AXI-Stream
  - Data streams
  - Interface signals
  - Data signaling
  - Byte qualifiers
- Example using TEA
  - Simple case
  - AXI stream case project

AMBA AXI-Stream

- There is only one data channel.
- No address channel, no read and write, always just master to slave (Effectively an AXI4 “write data” channel)
- Unlimited burst length

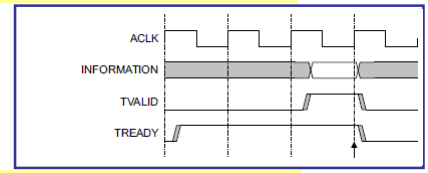




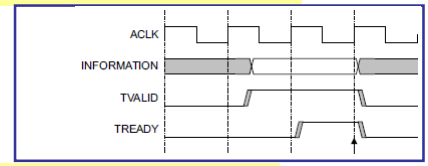
VALID/READY handshake mechanism

- Protocol (basic AXI handshaking)
  - The transmitter asserts TVALID on any clock cycle when it has data and de-asserts when it does not.
  - The receiver asserts TREADY on any cycle when it is ready for data and de-asserts when it is not.
  - In any clock cycle in which TVALID and TREADY are both asserted, data "moves", meaning it is taken by the receiver.
  - Deadlock avoidance scheme is required.
- Two-way flow-control
  - Both mater and slave can control the rate of information movement.
- Dual-ready handshake: two-way TVALID/TREADY handshake

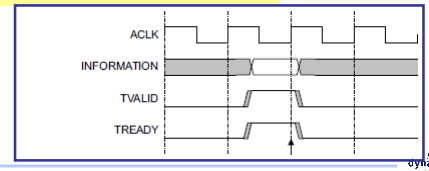
TREADY before TVALID handshake



TVALID before TREADY handshake

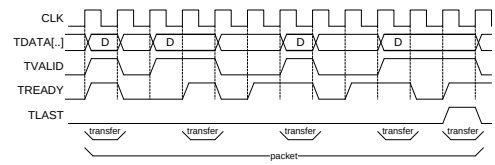


TVALID with TREADY handshake



AMBA AXI-Stream

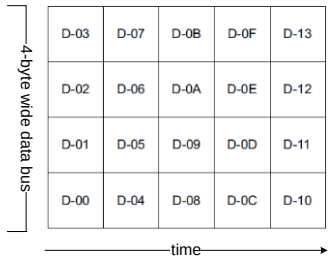
- Terminology
  - Transfer
    - A single transfer of data across an AXI4-Stream interface. A single transfer is defined by a single TVALID, TREADY handshake
  - Packet
    - A group of bytes that are transported together across an AXI4-Stream interface. (similar to burst)
    - A series of transfers shares the same TID and TDEST and ends with TLAST.
  - Frame
    - A frame contains an integer number of packets. (an entire video frame buffer)
  - Data stream
    - The transport of data from one source to one destination (a series of individual byte transfers; a series of byte transfers grouped together in packets)



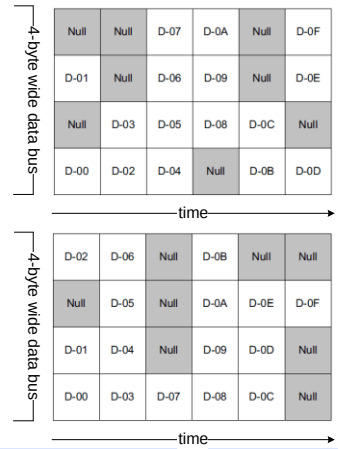
The merging of transfers with different **TID** or **TDEST** values is never permitted.

Data streams (1/2)

- A data stream is a transmission of a number of data bytes.
  - following is a continuous aligned stream, in which every packet has no null/position byte.

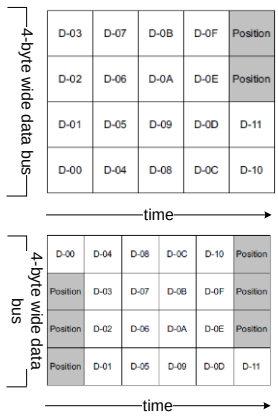


- A data stream can contain null bytes if needed, which have no meaning and can be inserted or removed from the stream.
  - Following two transfer identical information.

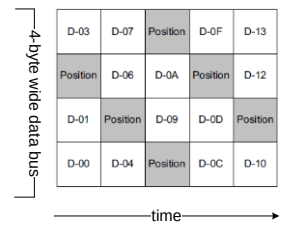


Data streams (2/2)

- Position byte can be used to align start and end of stream, which should be transferred to the destination
- Following two examples shows a continuous unaligned stream.



- A sparse stream is the transmission of a number of data bytes and position bytes.
  - Note that 'position' bytes replaces what has been there.

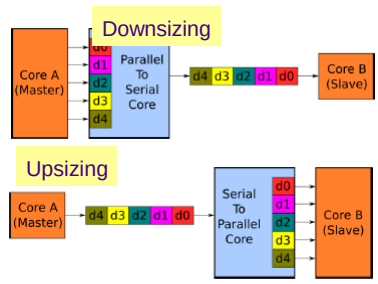


Interface signals

Signal	Source	Description
ACLK	Clock source	The global clock signal. All signals are sampled on the rising edge of **ACLK**.
ARESETn	Reset source	The global reset signal. **ARESETn** is active-LOW.
TVALID	Master	**TVALID** indicates that the master is driving a valid transfer. A transfer takes place when both **TVALID** and **TREADY** are asserted.
TREADY	Slave	**TREADY** indicates that the slave can accept a transfer in the current cycle.
TDATA[(8n-1):0]	Master	**TDATA** is the primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
TSTRB[(n-1):0]	Master	**TSTRB** is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as a data byte or a position byte.
TKEEP[(n-1):0]	Master	**TKEEP** is the byte qualifier that indicates whether the content of the associated byte of **TDATA** is processed as part of the data stream. Associated bytes that have the **TKEEP** byte qualifier deasserted are null bytes and can be removed from the data stream.
TLAST	Master	**TLAST** indicates the boundary of a packet.
TID[(i-1):0]	Master	**TID** is the data stream identifier that indicates different streams of data.
TDEST[(d-1):0]	Master	**TDEST** provides routing information for the data stream.
TUSER[(u-1):0]	Master	**TUSER** is user defined sideband information that can be transmitted alongside the data stream.

Data signaling

- Byte types
  - Data byte
    - valid information
  - Position (placeholder) byte
    - indicates the relative positions of data bytes within the stream.
  - Null byte
    - no valid data



- Adapting data width
  - Merging
    - the process of combining bytes from two different transfers into one transfer
  - Packing
    - the process of removing null bytes from a stream
  - Downsizing
    - converting from a given data bus width to a narrower data bus width
  - Upsizing
    - converting from a given data bus width to a wider data bus width

Byte qualifiers

- TKEEP[x]
  - associated with TDATA[(8x+7):8x]
  - indicates whether the content of the associated byte must be transported to the destination.
    - HIGH: must be transmitted to the destination
    - LOW: a null byte that can be removed from the stream
- TSTRB[x]
  - associated with TDATA[(8x+7):8x]
  - indicates whether the content of the associated byte is a data byte or a position byte.
    - When TKEEP[x] is high
      - TSTRB[x] indicates data or positional
        - HIGH: valid information
        - LOW: not valid information, but positional byte

TKEEP	TSTRB	Data Type	Description
HIGH	HIGH	Data byte	The associated byte contains valid information that must be transmitted between source and destination.
HIGH	LOW	Position byte	The associated byte indicates the relative position of the data bytes in a stream, but does not contain any relevant data values.
LOW	LOW	Null byte	The associated byte does not contain information and can be removed from the stream.
LOW	HIGH	Reserved	Must not be used.

Agenda

- Introduction
  - Evolution of AMBA Standards
  - AXI Interfaces
  - Memory mapped v.s. stream
  - Typical examples of stream interface
- AMBA AXI-Stream
  - VALID/READY handshake mechanism
  - AMBA AXI-Stream
  - Data streams
  - Interface signals
  - Data signaling
  - Byte qualifiers
- Example using TEA
  - Simple case
  - AXI stream case project

Example design

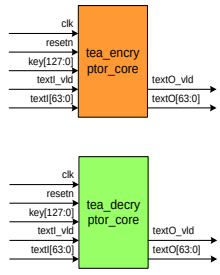
- TEA
  - Tiny Encryption Algorithm, David J. Wheeler and Roger M. Needham, Computer Laboratory, Cambridge University, England.
  - TEA algorithm can be implemented using basic arithmetic operations
    - Required operation : add, shift, xor
      - Simple hardware implementation

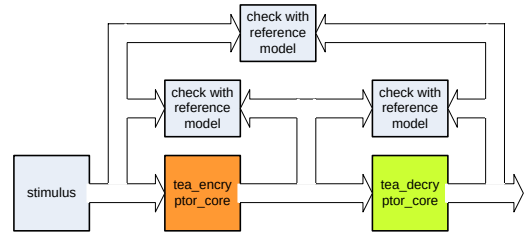

```
void encrypt (long* v, long* k) {
    unsigned long y=v[0],z=v[1], sum=0, /* set up */
    delta=0x9e3779b9, /* a key schedule constant */
    n=32 ;
    while (n-->0) { /* basic cycle start */
        sum += delta ;
        y += ((z<<4)+k[0]) ^ (z+sum) ^ ((z>>5)+k[1]) ;
        z += ((y<<4)+k[2]) ^ (y+sum) ^ ((y>>5)+k[3]) ;
    } /* end cycle */
    v[0]=y ; v[1]=z ;
}
```

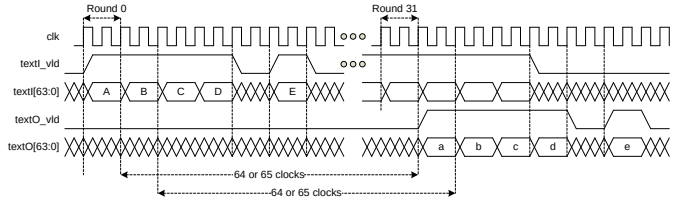
```
void decrypt (long* v, long* k) {
    unsigned long n=32, sum, y=v[0], z=v[1],
    delta=0x9e3779b9 ;
    sum=delta<<5 ;
    while (n-->0) { /* start cycle */
        z -= ((y<<4)+k[2]) ^ (y+sum) ^ ((y>>5)+k[3]) ;
        y -= ((z<<4)+k[0]) ^ (z+sum) ^ ((z>>5)+k[1]) ;
        sum-=delta ;
    } /* end cycle */
    v[0]=y ; v[1]=z ;
}
```

Wheeler, David J.; Needham, Roger M. (1994-12-16). "TEA, a tiny encryption algorithm". Lecture Notes in Computer Science (Leuven, Belgium: Fast Software Encryption: Second International Workshop) 1008: 363–366.

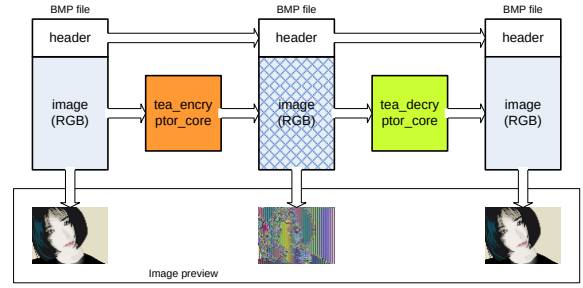
TEA implementation and verification







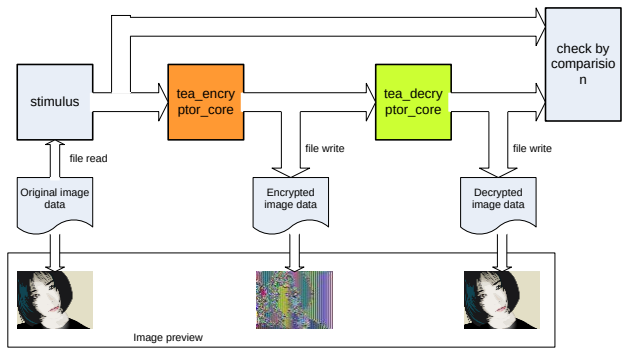
BMP file

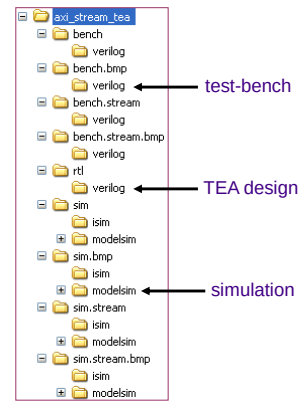


File header	bfType (2)	0	"BM"
	bfSize (4)	2	70
	bfReserved1 (2)	6	0
	bfReserved (2)	8	0
	bfOffBits (4)	10	54 (14+biSize)
Image header	biSize (4)	14	40
	biWidth (4)	18	2
	biHeight (4)	22	2
	biPlanes (2)	26	1
	biBitCount (2)	28	24
	biCompresssion (4)	30	0
	biSizeImage (4)	34	16
	biXpelsPerMeter (4)	38	2835
	biYPelsPerMeter (4)	42	2835
	biClrUsed (4)	46	0
	biClrImportant (4)	50	0
Pixel data	biClrImportant (4)	54	{0x00,0x00,0xFF} RED, pixel 0,1
		57	{0xFF,0xFF,0xFF} WHITE, pixel 1,1
		60	{0x00,0x00} Padding for 4-byte alignment
		62	{0xFF,0x00,0x00} BLUE, pixel 0,0
		65	{0x00,0xFF,0x00} GREEN, pixel 1,0
		68	{0x00,0x00} Padding for 4-byte alignment

'bench.bmp/verilog/bmp_handle.v' contains functions to handle BMP file.

TEA verification project



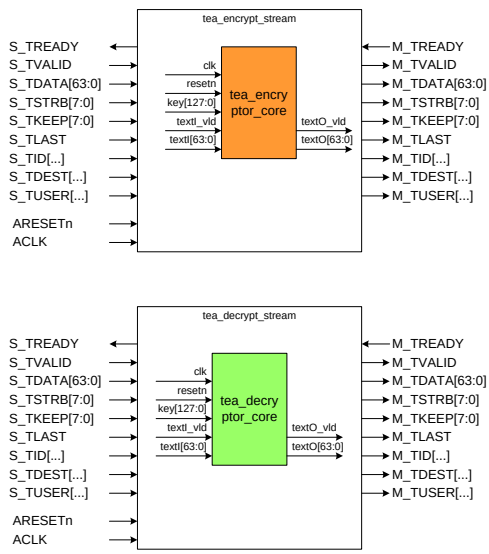


- Go to '.../axi_stream_tea' directory
- Have a look at TEA design in 'rtl/verilog' directory
  - 'tea_encryptor_core.v' and 'tea_decryptor_core.v'
- Have a look at test-bench in 'bench.bmp/verilog' directory
  - 'bmp_handle.v', 'bmp_stimulus_file.v', 'check.v' and 'top.v'
- Run 'make' in 'sim.bmp/modelsim' directory
  - It reads in 'face_320x240.bmp' and generates 'face_320x240_en.bmp' and 'face_320x240_de.bmp'.

Agenda

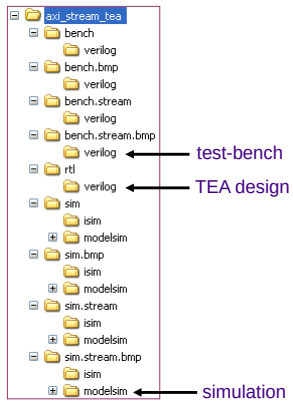
- Introduction
  - Evolution of AMBA Standards
  - AXI Interfaces
  - Memory mapped v.s. stream
  - Typical examples of stream interface
- AMBA AXI-Stream
  - VALID/READY handshake mechanism
  - AMBA AXI-Stream
  - Data streams
  - Interface signals
  - Data signaling
  - Byte qualifiers
- Example using TEA
  - Simple case
  - AXI stream case project

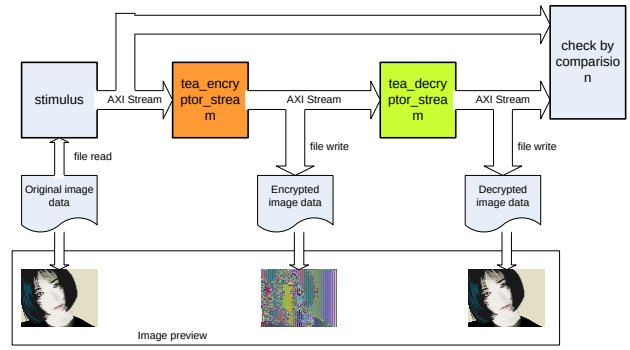
Build your own stream version



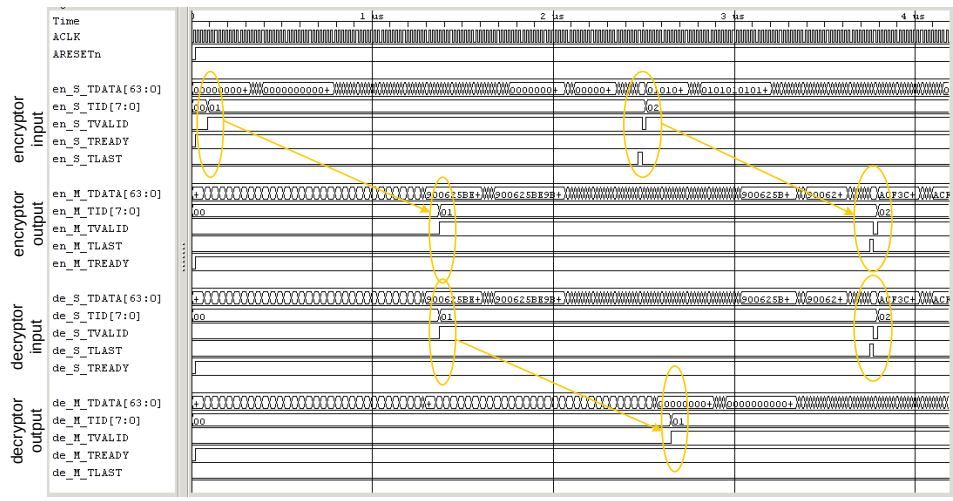
- 'tea_encrypt_stream.v' and 'tea_decryptor_stream.v' in 'rtl/verilog' directory
- Be careful to deal with 'S_TLAST' and 'M_TLAST'.
  - It may need your own plan.
- Some, e.g., 'TKEEP[7:0]' and 'TDEST[...]' can be ignored.

Build your own stream version





Simulation result



Usage of TID and TUSER

- 'TUSER[...]' can carries VSYNC and HSYNC information.
  - 2'b00: data
  - 2'b01: VSYNC
  - 2'b10: HSYNC
- 'TID[...]' can carries line identification

References

- AMBA® 4 AXI4-Stream Protocol Version: 1.0 Specification, IHI 0051A (ID030510), ARM Limited, 2010.