

# AHB总线协议学习

参考资料：

[AMBA总线—AHB总线协议详解\\_ahb协议-CSDN博客](#)

[AMBA AHB总线协议详细说明（AHB5协议）\\_ahb总线协议可以自定义吗-CSDN博客](#)

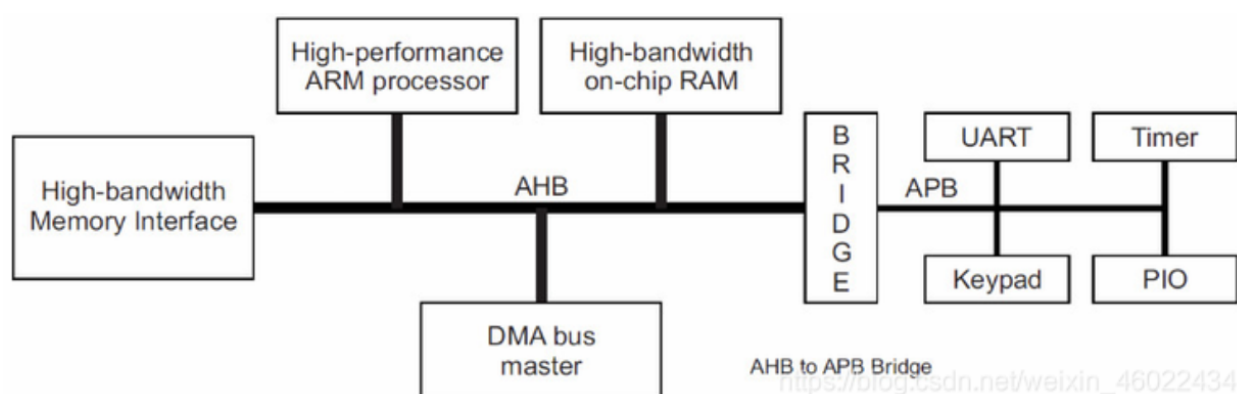
- 一、AHB总线介绍
  - 1、AHB总线的定义
  - 2、典型的AHB总线系统介绍
  - 3、基本传输阶段
- 二、AHB总线信号
  - 1、全局信号
  - 2、Master信号
  - 3、Slave/Multiplexor信号
  - 4、Decoder信号
- 三、AHB总线传输
  - 1、基本传输
    - 1.1. with no wait states
    - 1.2. with wait states
  - 2、传输类型
    - 2.1. HTRANS[1:0]含义
    - 2.2. 四种传输的组合
  - 3、Burst传输
    - 3.1. HBURST的含义
    - 3.2. HBURST的举例
  - 4、等待传输
    - 4.1. 传输类型改变
    - 4.2. 地址改变
- 五、面试问答

## 一、AHB总线介绍

### 1、AHB总线的定义

1. AHB全称：Advanced High-performance BUS，即**先进高性能总线**。
2. AHB总线一般用于 **高速且高性能** 的外设，例如其可将微控制器（MCU）、高带宽的片上RAM、高带宽的外部存储器、DMA总线Master、各种拥有AHB接口的外设连接起来构成一

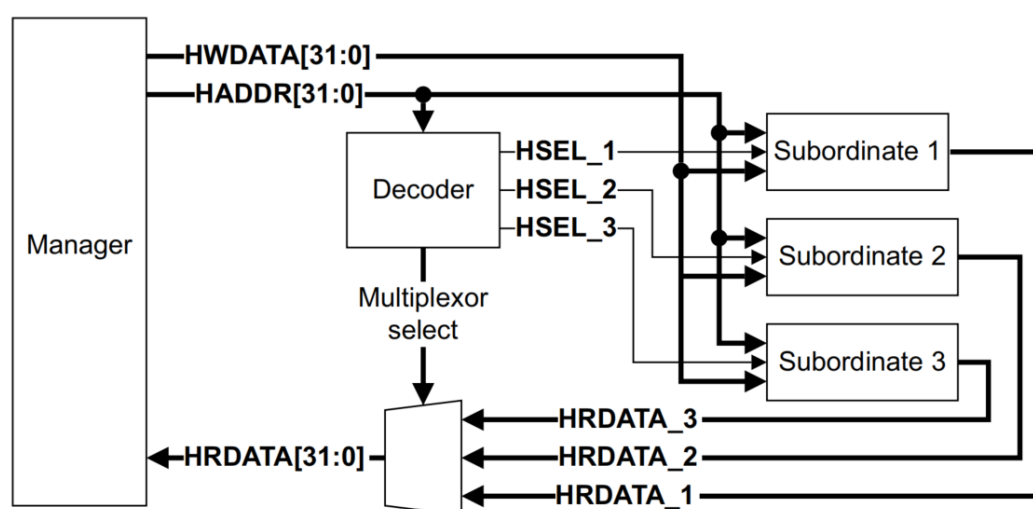
个独立的SOC系统，同时可以通过AHB-to-APB Bridge将 低速且低功耗的外设连接起来。



3. **AHB经历历程**：AHB——>AHB2+lite（简化了AHB的复杂设计）——>AHB5+lite（在AHB2的基础上新增了接口信号）。

## 2、典型的AHB总线系统介绍

### 1. AHB总线组成：

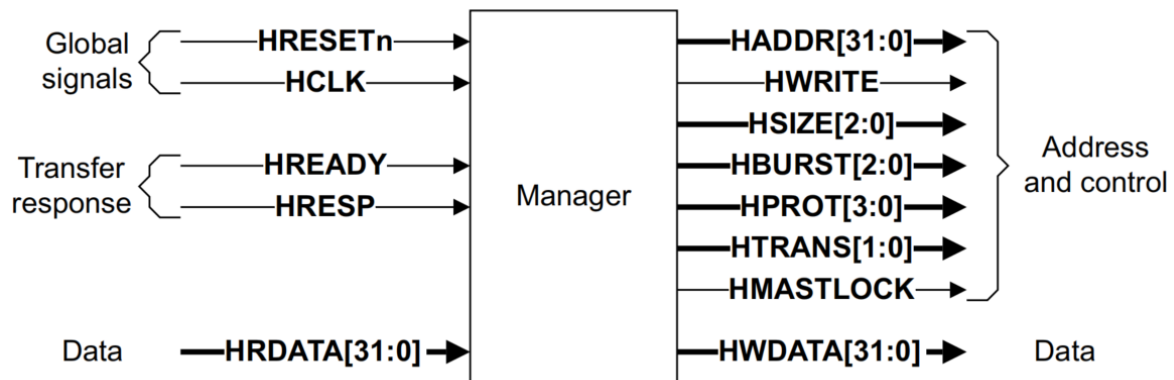


该图只包含一个Master，实际可以多Master对多Slave

- a. **主设备 (Master)**：发起一次读/写操作，某一时刻只允许一个主设备执行操作；
- b. **从设备 (Slave)**：响应一次读/写操作，通过地址映射选中某一个从设备的 HSELx 信号；
- c. **仲裁器 (Arbiter)**：用于 multi-Master 系统，允许某一个主设备控制总线；
- d. **译码器 (Decoder)**：通过地址译码决定选择哪一个从设备，并为选择器提供控制信号；
  - i. 假如有16个从设备，共需要 4bit 表示。若每个从设备有64KB存储空间，则需要 16bit 表示。因此在这里Master输出的 32bit 的低 16bit 可以映射至每个 Slave 用于内部地址选择，而高 16bit 可以决定具体访问哪个 Slave；

- ii. 注意：这里的高 16bit 必须不重叠，保证每个从设备的选择地址是独一无二的。
  - iii. 当HREADY=1时才会对HADDR、控制信号、HSELx信号采样。
  - iv. 给不存在的地址执行操作时（NONSEQ/SEQ，默认Slave响应错误，IDLE/BUSY，默认零等待）。
- e. **选择器 (Multiplexor)**：选择合适的从设备的输出信号（数据、响应信号）给主设备。

## 2. AHB Master的组成：

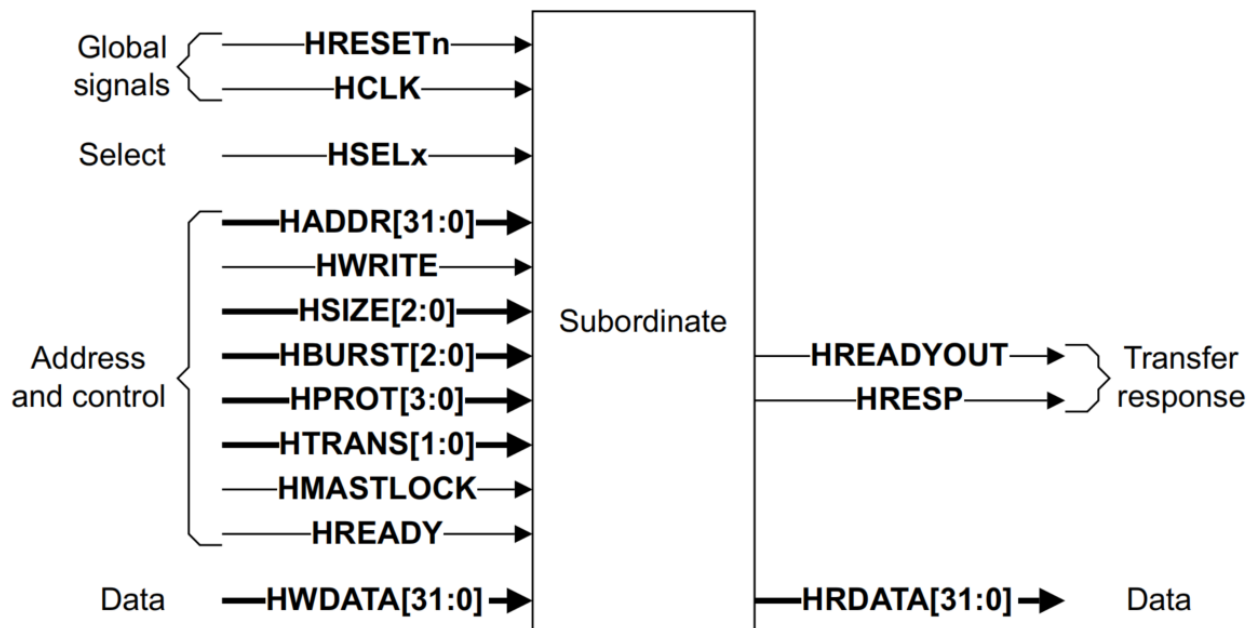


该图未显示 AHB5 新增信号

从信号名称上看与APB高度类似：

- a. 名称均以 “H” 开头，代表AHB；
- b. **新增四组信号：HSIZE、HBURST、HTRANS、HMASTLOCK；**
- c. HRESP对应APB的PSLVERR，以表示传输错误与否。

## 3. AHB Slave的组成：



- a. 信号方向与Master相反，与Master类似新增四组信号；
- b. HSELx信号拉高以表示选中该从设备。

### 3、基本传输阶段

- 1. **Address phase**：通常需要1个时钟周期，HWRITE、HADDR等控制信号同一拍有效，随后HREADY拉高表示当前地址已生效。
- 2. **Data phase**：通过需要1个或多个时钟周期（HREADYOUT和传输数据量大小等决定），HWDATA、HRDATA等和数据相关的信号有效，表示数据读写的开始。

## 二、AHB总线信号

### 1、全局信号

信号名	I/O	源	位宽	描述
HCLK	I	时钟源	1	AHB总线时钟。规定在时钟上通常输出信号在时钟上升沿之
HRESETn	O	复位源	1	AHB总线复位。低有效。

- 1. **关于HCLK**：AHB5规定了 Stable\_Between\_Clock 属性，其为True时需要保证输入信号在时钟上升沿到来之后保持稳定，不能有“毛刺”的出现；反之允许有毛刺的出现。
- 2. **关于HRESETn**：
  - a. 复位期间，所有Master的数据、地址信号需有效，且HTRANS[1:0]需为IDLE状态；
  - b. 复位期间，所有Slave的HREADYOUT信号需为高电平。

### 2、Master信号

注：这里只写了常用的信号

信号名	I/O	目的地	位宽	描
HADDR	O	Slave/Decoder	ADDR_WIDTH	AHB地址总线。
HWDATA	O	Slave	DATA_WIDTH	写数据总线。
HBURST	O	Slave	0/3	规定Burst传输次数（方式。
HSIZE	O	Slave	3	规定每次传输的数据大

<b>HWRITE</b>	O	Slave	1	传输方向。1-写，0-读。
<b>HTRANS</b>	O	Slave	2	规定传输类型。 IDLE、BUSY、NONSEQUENTIAL。
<b>HPROT</b>	O	Slave	0/4/7	规定数据传输的性质。
<b>HMASTLOCK</b>	O	Slave	1	锁定Master传输，锁断。

**a. 关于HBURST和HSIZE:**

- i. 在APB中，一次只能扔出一个数据，且该数据的位宽由PSTRB决定；
- ii. 在AHB中，一次可以扔出多个数据，其由HBURST决定，且该数据的位宽由HSIZE决定。

**b. 关于HTRANS:**

- i. 在APB中，由PSELx和PENABLE指示数据的读/写过程；
- ii. 在AHB中，由HTRANS指示数据的读/写过程，可以简单理解为：PSELx=1, PENABLE=0——>NONSEQUENTIAL; PSELx=1, PENABLE=1——>SEQUENTIAL;
- iii. 这里的BUSY状态可以置于NONSEQUENTIAL和SEQUENTIAL之间，若在一个数据的发送过程中出现了暂时无法继续发送的状态，可以置HTRANS为BUSY，待恢复为正常状态以后再置为SEQUENTIAL。

**c. 关于HSIZE:**

<b>HSIZE[2]</b>	<b>HSIZE[1]</b>	<b>HSIZE[0]</b>	<b>Size</b>	<b>描述</b>
<b>0</b>	<b>0</b>	<b>0</b>	<b>8</b>	<b>byte</b>
<b>0</b>	<b>0</b>	<b>1</b>	<b>16</b>	<b>halfword</b>
<b>0</b>	<b>1</b>	<b>0</b>	<b>32</b>	<b>word</b>
<b>0</b>	<b>1</b>	<b>1</b>	<b>64</b>	<b>doubleword</b>
<b>1</b>	<b>0</b>	<b>0</b>	<b>128</b>	-
<b>1</b>	<b>0</b>	<b>1</b>	<b>256</b>	-
<b>1</b>	<b>1</b>	<b>0</b>	<b>512</b>	-
<b>1</b>	<b>1</b>	<b>1</b>	<b>1024</b>	-

- i. 前四种常用，**HSIZE代表的数据位宽应小于等于读/写数据总线宽度**。例如DATA\_WIDTH=64，则HSIZE只能为0、1、2、3。

### 3、Slave/Multiplexor信号

信号名	I/O	目的地	位宽	
HRDATA	O	Multiplexor/Master	DATA_WIDTH	读数据总线。
HRESP	O	Multiplexor/Master	1	指示传输错误与
HREADYOUT	O	Multiplexor/Master	1	准备就绪信号，

### 4、Decoder信号

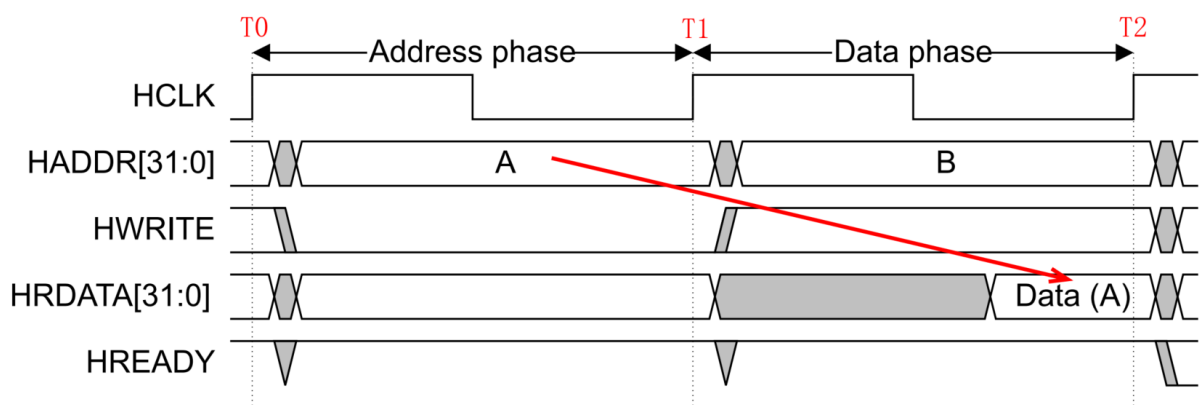
信号名	I/O	目的地	位宽	
HSELx	O	Slave/Multiplexor	1	从设备选择信号，非IDLE传输时，控制信号同一拍

## 三、AHB总线传输

### 1、基本传输

#### 1.1. with no wait states

##### 1. 读操作：

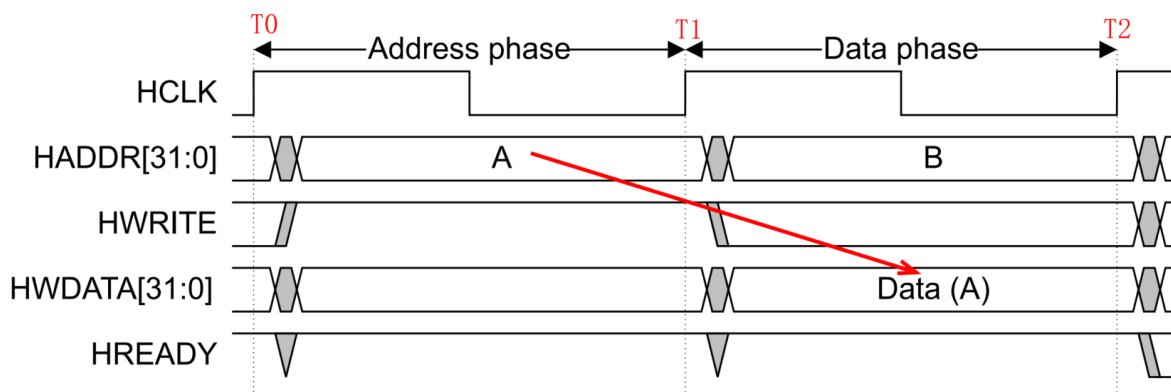


- a. **T0**: Master在时钟上升沿将HADDR、HWRITE驱动至Slave的输入端口（图中有一定的延迟滞后，不是一定在时钟上升沿发生变化）；
- b. **T1**: Slave在此时刻采样HADDR、HWRITE信号，判断已做好读出数据的准备，由此在时钟上升沿将HREADY信号拉高，且内部可能将HADDR存放的数据放置读数据总线

HRDATA上;

- c. **T2**: Master在时钟上升沿采样HREADY信号为1, 即表示可以接收数据, 由此采样HRDATA上的数据 “DATA(A)” ;
- d. 可以看出**地址信号与数据信号错开了相应的时钟周期**, 且在HREADY拉高时HADDR、HWRITE不需要维持原有的状态, 因此**AHB适合流水线操作**。

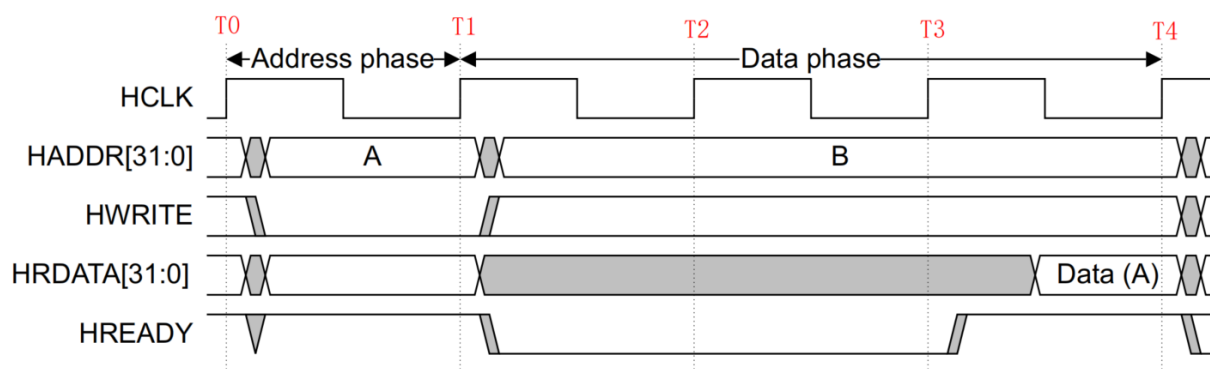
## 2. 写操作:



- a. **T0**: 与读操作不同的是, 此时HWRITE=1, 表示写操作;
- b. **T1**: 与读操作不同的是, 此时由Master根据HREADY的状态驱动数据Data(A)至写数据总线HWDATA上;
- c. **T2**: Slave在时钟上升沿将HWDATA上的数据写入地址空间HADDRA中;
- d. 同样这里址信号与数据信号错开了相应的时钟周期。

## 1.2. with wait states

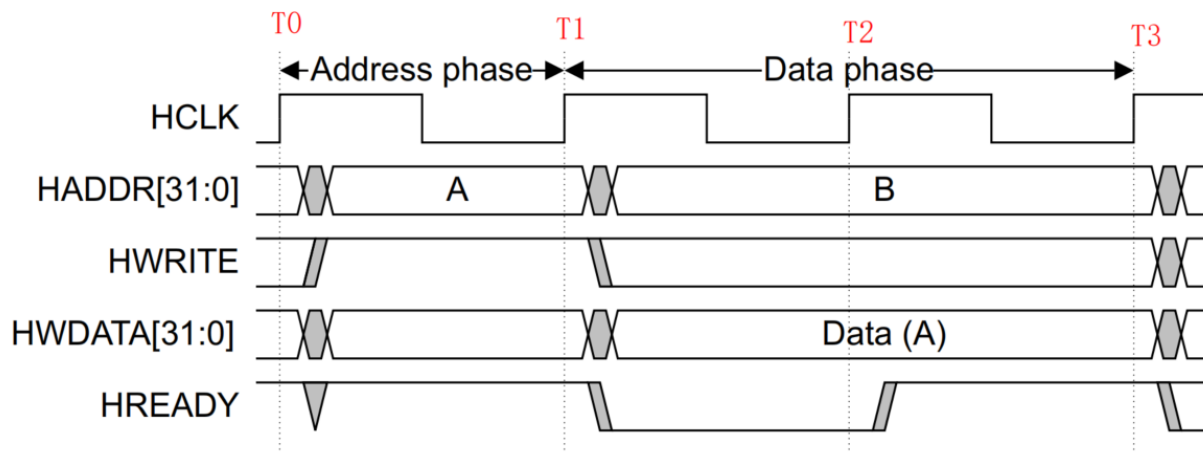
### 1. 读操作:



- a. **T0**: 同上述with no wait states;
- b. **T1**: Slave在此阶段采样到了Master发送来的HADDR、HWRITE信号, 但是判断自身暂无能力读出数据, 因此在时钟上升沿驱动HREADY信号为低电平 (有一定的滞后时间) ;
- c. **T2**: Master在此阶段采样到了HREADY为低电平, 因此得知Slave未准备好读出数据, 暂时不接收HRDATA上的数据 (这里可能输入进了Master, 但是其可能不保存下来) ;

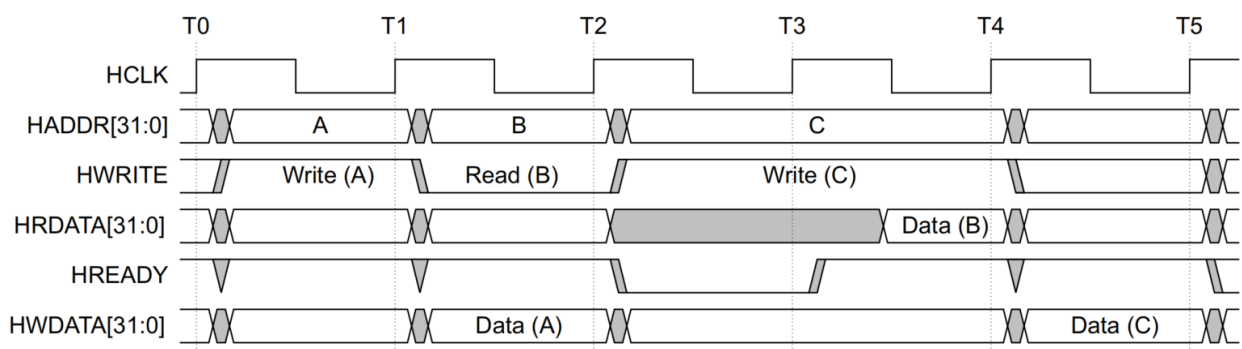
- d. **T3**: Slave在此阶段驱动HREADY信号为高电平，表示准备好读出数据，同时将地址A上的数据驱动至HRDATA上；
- e. **T4**: Master在此阶段采样到了HREADY为高电平，由此采样HRDATA上的数据 Data(A)。
- f. 疑问：感觉这里在得到了要读出的地址A后，HADDR上的值可以随意更改，不一定一直维持A不变，因为此时Slave设置了要读的地址为A，直到此次传输结束才将地址更改为下一次要传输的地址。——>AHB协议有副作用，当产生这样的延迟时，下一次要传输的地址也会延迟。

## 2. 写操作：



- a. **T0**: 同上述with no wait states;
- b. **T1**: 同上，Slave将HREADY置为低电平；但是这里Data(A)还是由Master正常驱动，但是Slave并未将该数据写入；
- c. **T2**: Slave将HREADY置为高电平，表示准备好接收数据；但是**此时Master下一个时钟周期才能采样到HREADY信号的变化，因此保持HWDATA上的数据不变**；
- d. **T3**: Master采样到HREADY为高电平，由此将Data(A)写入地址A中。

## 3. 多次传输：



### a. T0:

- i. Master: 采样HREADY=1——>驱动HADDR、HWRITE，执行向地址A写数的操作 (Address phase A) ；



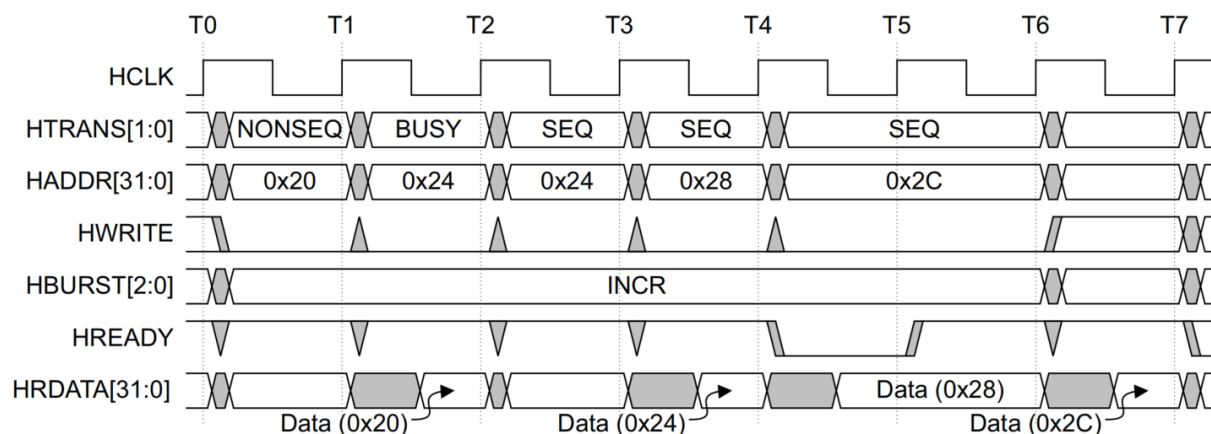
- b. T1:
  - i. Slave: 采样HADDR、HWRITE——>驱动: HREADY信号无变化即准备好写入数据 (Data phase A) ;
  - ii.Master: 采样HREADY=1——>驱动: HWDATA为要写入的数据; 同时驱动 HADDR、HWRITE, 执行读地址B的操作 (Address phase B) ;
- c. T2:
  - i. Slave: 将数据写入地址A, 完成第一次操作; 同时对HADDR、HWRITE信号进行采样, 将HREADY信号拉低, 告知Master还未准备好执行下一次操作 (Data phase B) ;
  - ii.Master: 驱动HADDR、HWRITE, 执行写地址C的操作 (Address phase C) ;
- d. T3:
  - i. Slave: 驱动HREADY信号为高, 将数据Data(B)驱动至HRDATA上 (Data phase B) ;
  - ii.Master: 此时还未检测到HREADY信号的变化, 因此维持HADDR、HWRITE信号不变 (Address phase C) ;
- e. T4:
  - i. Slave: 对HADDR、HWRITE信号进行采样, 且HREADY信号无变化即准备好写入数据 (Data phase C) ;
  - ii.Master: 采样到HREADY信号为高, 读取HRDATA上的数据, 完成第二次操作; 同时将Data(C)放置HWDATA上;
- f. T5:
  - i. Slave: 将数据写入地址C, 完成第三次操作。
- g. **总结: Master对HRDATA、HPREADY进行采样, HWDATA、HWRITE、HADDR进行驱动; Slave反之。Master根据HPREADY决定是否执行下一次操作。**

## 2、传输类型

### 2.1. HTRANS[1:0]含义

HTRANS	类型	描述
00	IDLE	M: 无数据需要传输。 S: 忽略地址与其他控制信号, 并以零等待状态响应
01	BUSY	M: 在burst传输过程中, 下一个数据无法立即传输地址及其他控制信号需要反映下一次操作。 S: 忽略地址与其他控制信号, 并以零等待状态响应
10	NONSEQUENTIAL	M: 单笔数据或burst传输的第一笔数据。

## 2.2. 四种传输的组合



1. T0: M发现有4-beat 32bit的传输要执行, 且采样HREADY=1, 则执行该操作, 驱动HTRANS=NONSEQ, HADDR=0X20, 表明执行0X20地址的读出操作。
2. T1: S完成0X20地址的读出操作。M此时插入BUSY, 以表明不能完成第二次操作, 但仍置HADDR=0X24 (满足要求)。【地址+4, 32bit/8bit=4】
3. T2: S忽略地址及其余控制信号。M准备好则置HTRANS=SEQ, 保持HADDR不变。
4. T3: S完成0X24地址的读出操作。M执行第三次操作, 置HADDR=0X28。
5. T4: S置HREADY=0, 表明此时未准备好读出0X28地址。M执行第四次操作, 置HADDR=0X2C。
6. T5: S置HREADY=1, 完成0X28地址的读出操作。M采样HREADY=0, 保持不变。
7. T6: S完成0X2C地址的读出操作。M完成整个操作。
8. **注意: T0 M发起操作——>T1 S完成操作——>T2 M得到值。这里传输实际需要8个clock (1发起+4个数+2延迟+1得到)。**

## 3、Burst传输

### 3.1. HBURST的含义

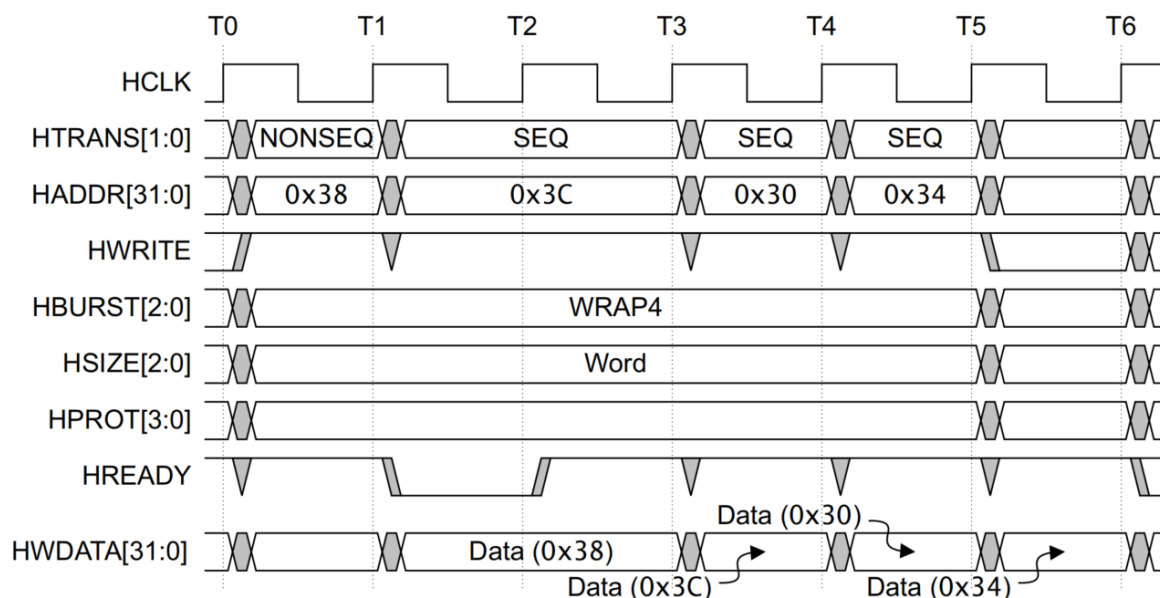
HBUSRT[2:0]	类型	描述
000	SINGLE	单个传输
001	INCR	不定长burst传输
010	WRAP4	4-beat wrap 传输
011	INCR4	4-beat 地址递增传输

<b>100</b>	WRAP8	8-beat wrap 传输
<b>101</b>	INCR8	8-beat 地址递增传输
<b>110</b>	WRAP16	16-beat wrap 传输
<b>111</b>	INCR16	16-beat 地址递增传输

1. **WRAP**: 地址具有边界。例如, 定义word 4-beat传输 (16-byte), 起始地址为0X34, 则依次为0X38、0X3C、0X30。适于Cache。
2. **INCR**: 地址递增。如, 定义word 4-beat传输 (16-byte), 起始地址为0X34, 则依次为0X38、0X3C、0X40。
3. HBURST规定了数据传输的个数, 若要计算总数据量, 则要根据HSIZE计 (规定每个数据的位宽) 算。
4. 注意地址一般由主机计算出来给到从机。菜鸟询问关于AHB总线中Burst传输的问题 - 数字IC设计讨论(IC前端|FPGA|ASIC) - EETOP 创芯网论坛 (原名: 电子顶级开发网) -
5. 其他:
  - a. **SINGLE**传输后需跟**NONSEQ**/IDLE;
  - b. **INCR**后可以跟**BUSY**/**NONSEQ**/IDLE结束传输;
  - c. 其余后需跟**SEQ**结束传输。
6. 发生错误的情况:
  - a. Slave发出failure response, Master可以取消或者不取消本次传输。若取消则需恢复HTRANS=IDLE。
  - b. multi-masters时, slave需终止上一次的错误, 以合适响应当前master的操作。

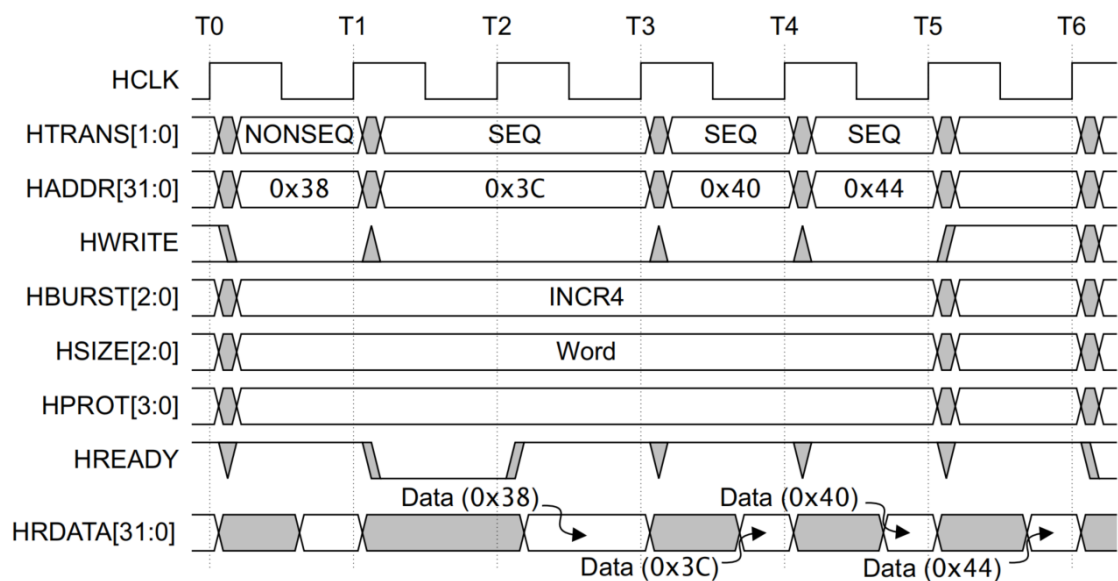
## 3.2. HBURST的举例

### 1. WRAP4写



- a. **T0**, M采样HREADY=1, 执行4-beat word写的操作, 置HTRANS=NONSEQ, HADDR=0X38, 执行1st-beat的地址 phase 操作。
- b. **T1**, S驱动HREADY=0, 未准备好。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X3C, 执行2nd-beat的地址 phase 操作; 置HWDATA=Data(0X38), 执行1st-beat的数据 phase 操作。
- c. **T2**, S驱动HREADY=1, 准备好。M采样HREADY=0, 维持HTRANS、HADDR、HWDATA不变。
- d. **T3**, S完成1st-beat的写入操作。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X30, 执行3th-beat的地址 phase 操作; 置HWDATA=Data(0X3C), 执行2nd-beat的数据 phase 操作。
- e. **T4**, S完成2nd-beat的写入操作。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X34, 执行4th-beat的地址 phase 操作; 置HWDATA=Data(0X30), 执行3th-beat的数据 phase 操作。
- f. **T5**, S完成3th-beat的写入操作。M采样HREADY=1, 恢复; 置HWDATA=Data(0X34), 执行4th-beat的数据 phase 操作。
- g. **T6**, S完成4th-beat的写入操作。
- h. **其他**: 1 (得到) + 4 (4个) + 1 (等待) + 1 (最后写入) 个时钟周期。

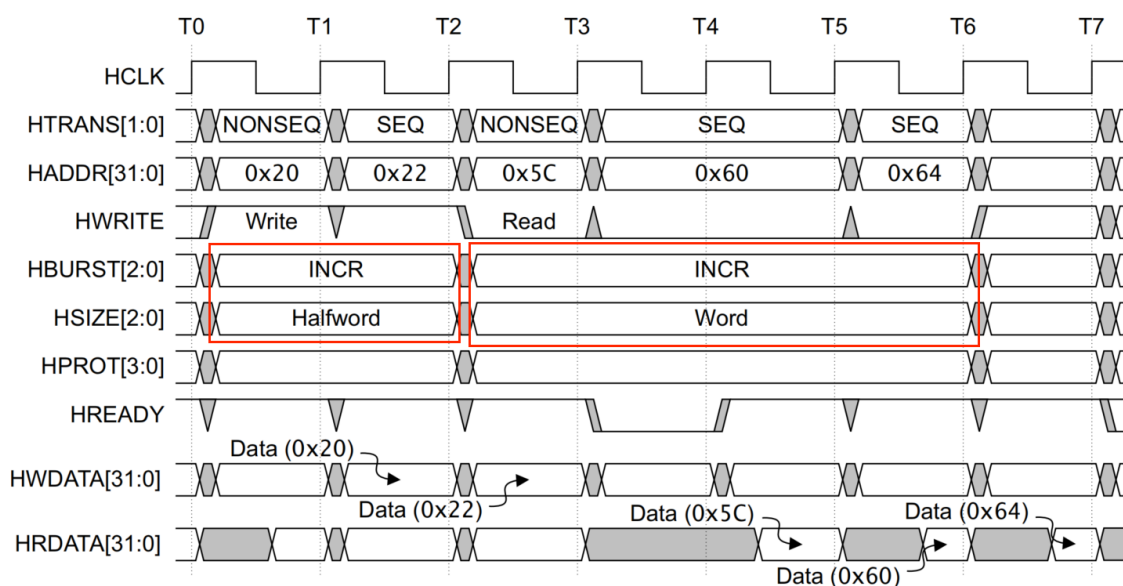
## 2. INCR4读



- a. **T0**, M采样HREADY=1, 执行4-beat word读的操作, 置HTRANS=NONSEQ, HADDR=0X38, 执行1st-beat的地址 phase 操作。
- b. **T1**, S驱动HREADY=0, 未准备好。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X3C, 执行2nd-beat的地址 phase 操作。
- c. **T2**, S驱动HREADY=1, 准备好, 完成1st-beat的读出操作。M采样HREADY=0, 维持HTRANS、HADDR不变。
- d. **T3**, S完成2nd-beat的读出操作。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X40, 执行3th-beat的地址 phase 操作。

- e. **T4**, S完成3th-beat的读出操作。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X44, 执行4th-beat的地址 phase操作。
- f. **T5**, S完成4th-beat的读出操作。M恢复。

### 3. INCR



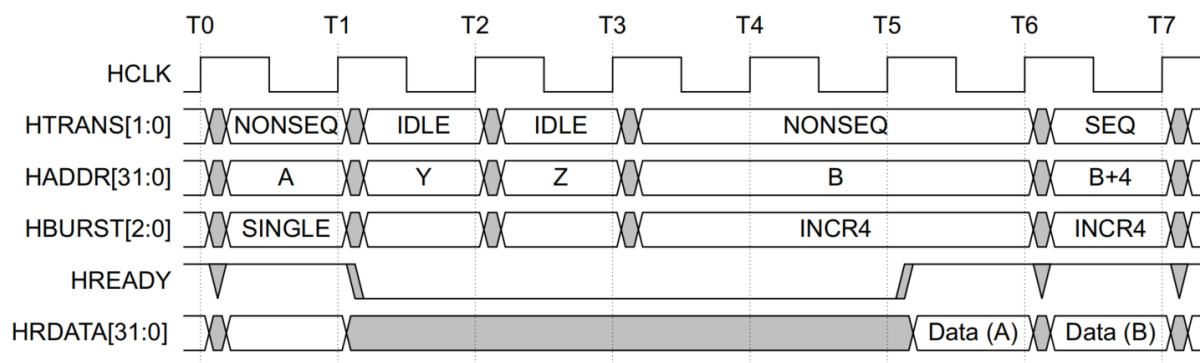
- a. **T0**, M采样HREADY=1, 执halfword写的操作, 置HTRANS=NONSEQ, HADDR=0X20, 执行1st-beat的地址 phase操作。
- i. **T1**, M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X22, 执行2nd-beat的地址 phase操作; 置HWDATA=Data(0X20), 执行1st-beat的数据 phase操作。
- b. **T2**, S完成1st-beat的写入操作。M采样HREADY=1, 执行word读操作, 置HTRANS=NONSEQ, HADDR=0X5C, 执行1st-beat的地址 phase操作; 置HWDATA=Data(0X22), 执行2nd-beat的数据 phase操作。
- c. **T3**, S完成2nd-beat的写入操作, 驱动HREADY=0, 暂缓下次操作。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X60, 执行2nd-beat的地址 phase操作。
- d. **T4**, S驱动HREADY=1, 完成1st-beat的读出。M采样HREADY=0, 维持HTRANS、HADDR不变。
- e. **T5**, S完成2nd-beat的读出。M采样HREADY=1, 置HTRANS=SEQ, HADDR=0X64, 执行3th-beat的地址 phase操作。
- f. **T6**, S完成3th-beat的读出。M恢复。

## 4、等待传输

注: HREADY实现等待传输。

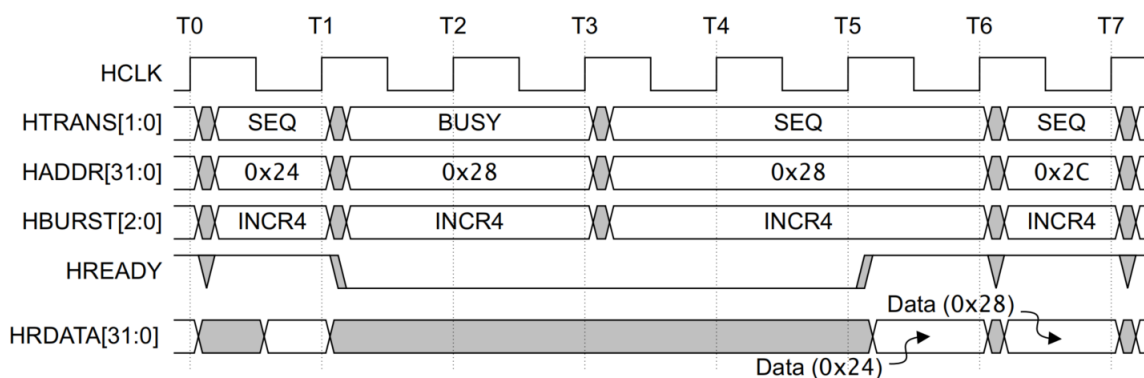
### 4.1. 传输类型改变

#### 1. 情况一: IDLE——>NONSEQ



- T0, M采样HREADY=1, 执行SINGLE burst读操作 (送出地址A)。
- T1, M采样HREADY=1, 插入IDLE传输 (送出地址Y) ; S驱动HREADY=0, 未准备好读出地址A的数据。(SIGNLE后面可以是IDLE/NONSEQ)
- T2, M插入IDLE传输 (改变地址Y为Z) 【在这里HREADY=0, 不对HADDR、HSELx、控制信号进行采样, 因此HADDR的值可以自行更改】。
- T3, M插入INCR4 burst读操作 (IDLE——>NONSEQ) (送出地址B)。**
- T4, M采样HREADY=0, 保持HTRANS、HADDR不变。**
- T5, S驱动HREADY=1, 读出数据Data(A), 地址B送入S。 **M采样HREADY=0, 保持HTRANS、HADDR不变。**
- T6, M采样HREADY=1, 执行读地址B+4的操作。S读出数据Data(B)。
- 注意: 这里当waited states时, IDLE改变为NONSEQ后, NONSEQ需保持不变直到HREADY=1(T6改变)。**

## 2. 情况二: 定长burst, BUSY——>SEQ (相当于插入busy缓冲传输)

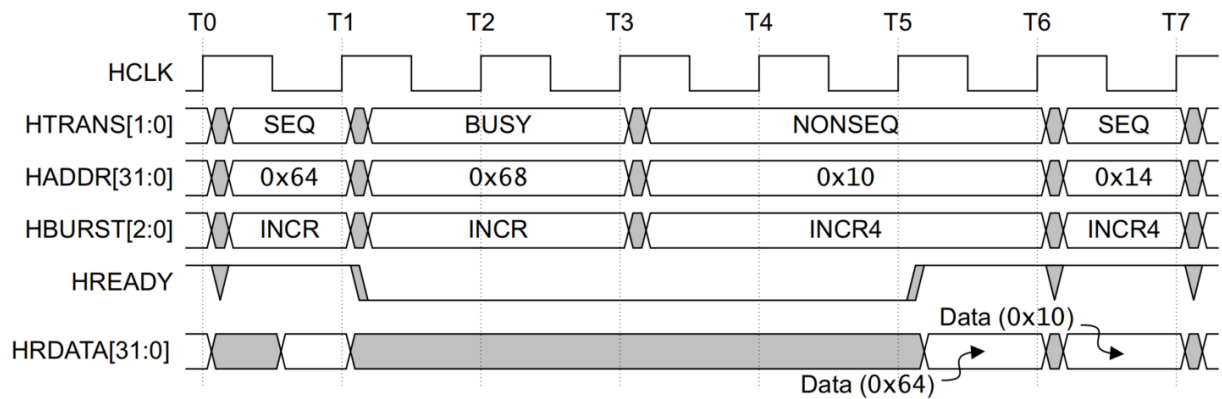


- T0, M采样HREADY=1, 执行INCR4 burst读操作 (送出地址0X24)。
- T1, M插入BUSY, 送出地址0X28 (送出下一次要执行的操作)。S驱动HREADY=0, 未准备好读出地址0X24的数据。
- T2, M和S维持原有状态。
- T3, M插入执行地址0X28的读出操作, HTRANS=SEQ(判断自身已准备好)。**
- T4, M采样HREADY=0, 保持HTRANS、HADDR不变。**



- f. T5, **M采样HREADY=0, 保持HTRANS、HADDR不变**。S驱动HREADY=1, 读出数据Data(0X24), 地址0X28送入S。
- g. T6, M采样HREADY=1, 执行地址0X2C的读出操作。S读出数据Data(0X28)。
- h. **注意: 这里当waited states时, BUSY改变为SEQ后, SEQ需保持不变直到HREADY=1(T6改变)。**

### 3. 情况三: INCR, BUSY——>其余任何



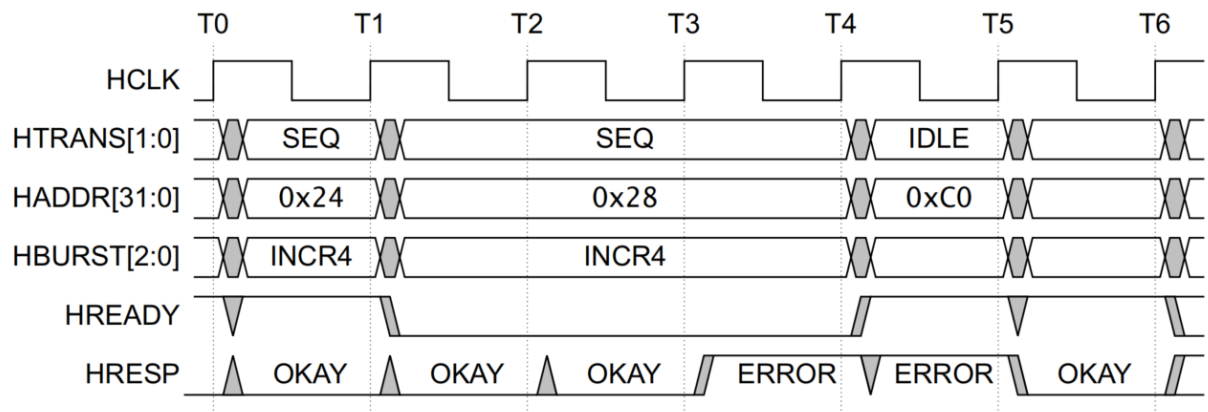
- a. T0, M采样HREADY=1, 执行INCR burst读操作 (送出地址0X64) 。
- b. T1-T2, M插入BUSY, 送出地址0X68。S驱动HREADY=0, 未准备好读出地址0X64的数据。
- c. T3, M插入执行地址0X10 (1st-beat) (INCR4 burst) (NONSEQ, 结束上一次burst) 的读出操作。
- d. T4, **M采样HREADY=0, 保持HTRANS、HADDR不变**。
- e. T5, **M采样HREADY=0, 保持HTRANS、HADDR不变**。S驱动HREADY=1, 读出数据Data(0X64), 地址0X10送入S。
- f. T6, M采样HREADY=1, 执行地址0X14的读出操作。S读出数据Data(0X10)。
- g. **注意: 这里当waited states时, BUSY改变为NONSEQ后, NONSEQ需保持不变直到HREADY=1(T6改变)。此外, 有BUSY变为NONSEQ后, 表示上一次burst结束, 若变为IDLE/SEQ则继续执行上一次burst。**

## 4.2. 地址改变

### 1. 情况一: IDLE期间

如上述 **IDLE——>NONSEQ** 情况, 在等待期间, HTRANS由SEQ变为IDLE时, 地址开始为Y之后可以变为其他值 (Z)。但是当由IDLE变为NONSEQ时, 地址变为B, 此时地址B不能更改除非HREADY=1 (因此此时还未执行ADDRESS B phase, 等执行完以后可以更改)。但是如果IDLE变为SEQ则不能更改。

### 2. 情况二: Error响应



- T0, M采样HREADY=1, 执行INCR4 burst 某个beat的读操作, 送出地址0X24。
- T1, M采样HREADY=1, 执行下个beat的读操作, 送出地址0X28。S驱动HREADY=0, 未准备好读出地址0X24的数据。
- T2-T3, M和S保持不变。S驱动HRESP=1, 响应执行可能失败。
- T4, M采样HRESP=1, HTRANS=IDLE, 地址变为0XC0。** S驱动HREADY=1, 但是此时地址0X24读出操作可能失败。
- T5, S驱动HRESP=0, 响应此时正常, 可以执行0XC0的读出操作 (保存进0XC0地址)。
- 其他: **AHB需要两个时钟周期响应错误, APB仅需要一个时钟周期即可。**

	HREADY	
HRESP	0	1
0	暂停传输 (等待)	成功传输
1	错误, cycle 1	错误, cycle 2

## 五、面试问答

- AHB为什么适用于高速且高性能的接口?
  - 流水线操作, 几乎每个时钟周期就能完成一次操作 (地址+数据);
  - burst传输, 一次就可以发出多次操作;
  - 仲裁机制, 某一时刻只允许一个Master控制总线。



