Must a read after a write to the same address return the newly written data?

Scenario

When there is an AHB write followed by a read from the same address, should the read return the old or the new data when the read address phase is in same cycle of the write data phase?

Answer

The answer to this question is dependent on the design of the slave.

A simple slave will not be buffering any data, so the returned read data will be the latest. A more complex slave could implement buffering for write data (if allowed by HPROT[2]) and so it could "snoop" the write buffer contents before returning read data, or it might just return the previously stored data regardless of what might be buffered.

In conclusion, all the AHB specification requires is that data is returned.

What system support is required if a slave can be powered down or have its clock stopped?

Answer

If a slave access is attempted while that slave is in a power down state or has had its clock stopped, you must ensure that an access will cause the power/clock to be restored, or else configure the AHB decoder up to redirect any such accesses to the dummy slave so that the system does not hang forever when an access to the device is made when it is disabled.

Redirecting the access in this way will ensure that random "IDLE" addresses are treated with the HREADY high and HRESP=OKAY default response, but real accesses (NONSEQ or SEQ) will be detected with an ERROR response.

When can Early Burst Termination occur ?

Answer

Bursts can be early terminated either as a result of the Arbiter removing the HGRANT to a master part way through a burst, or after a slave returns a non-OKAY response to any beat of a burst. Note however that a master cannot decide to terminate a defined length burst unless prompted to do so by the Arbiter or Slave responses.

All AHB Masters, Slaves and Arbiters must be designed to support Early Burst Termination.

Can HTRANS change whilst HREADY is low?

Answer

In general, an AHB master should not change control signals whilst HREADY is low. However it is allowable to change HTRANS in the following conditions:

- **HTRANS=IDLE**
  The AHB master is performing internal operations and has not yet committed to a bus transfer. However during the AHB wait states (HREADY low) the master may determine that a bus transfer is required and change HTRANS on the next cycle to NONSEQ.
- **HTRANS=BUSY**
  HTRANS is being used to give the master time to complete internal operations, which may be entirely independent of HREADY (i.e. wait states on the AHB). Therefore HTRANS can change on the next cycle to any legal value, i.e. SEQ if the burst is to continue, IDLE if the burst has completed, NONSEQ if a separate burst is to begin.
- **HRESP=SPLIT/RETRY**
  As stated in the AHB specification, a master must assert IDLE on HTRANS during the second cycle of the two-cycle SPLIT or RETRY slave response so HTRANS will change value from the first cycle to the second cycle of the response.

- **HRESP=ERROR**
  The master is permitted to change HTRANS in reaction to an ERROR response in the same way as in reaction to a SPLIT/RETRY response and cancel any further beats in the current burst (even if HBURST is indicating a defined-length burst). In this case HTRANS changes to IDLE on the second cycle of the response. Alternatively, the master is permitted to continue with the current transfers

Can a BUSY transfer occur at the end of a burst?

Answer

A BUSY transfer can only occur at the end of an undefined length burst (INCR). A BUSY transfer cannot occur at the end of a fixed length burst (SINGLE, INCR4, WRAP4, INCR8, WRAP8, INCR16, WRAP16).

Can a master change the address/control signals during a waited transfer?

Answer

Yes. If the address/control signals are indicating an IDLE transfer then the master can change to a real transfer (NONSEQ) when HREADY is low.

However, if a master is indicating a real transfer (NONSEQ or SEQ) then it cannot cancel this during a waited transfer unless it receives a SPLIT, RETRY or ERROR response.

Can an AHB master be connected directly to an AHB slave?

Answer

Any slave which does not use SPLIT responses can be connected directly to an AHB master. If the slave does use SPLIT responses then a simplified version of the arbiter is also required.

If an AHB master is connected directly to an AHB slave it is important to ensure that the slave drives HREADY high during reset and that the select signal HSEL for the slave is tied permanently high.

Do all slaves have to support the BUSY transfer type?

Answer

Yes. All slaves must support the BUSY transfer type to ensure they are compatible with any bus master.

Does the address have to be aligned, even for IDLE transfers?

Answer

Yes. The address should be aligned according to the transfer size (HSIZE) even for IDLE transfers. This will prevent spurious warnings from bus monitors used during simulation.

How many masters can there be in an AHB system?

Answer

The AHB specification caters for up to 16 masters. However, allowing for a dummy bus master means the maximum number of real bus masters is actually 15. By convention bus master number 0 is allocated to the dummy bus master.

Is HREADY an input or an output from slaves?

Answer

An AHB slave must have the HREADY signal as both an input and an output.

HREADY is required as an output from a slave so that the slave can extend the data phase of a transfer.

HREADY is also required as an input so that the slave can determine when the previously selected slave has completed its final transfer and the first data phase transfer for this slave is about to commence.

Each AHB Slave should have an HREADY output signal (conventionally named HREADYOUT) which is connected to the Slave-to-Master Multiplexer. The output of this multiplexer is the global HREADY signal which is routed to all masters on the AHB and is also fed back to all slaves as the HREADY input.

Is a default slave really necessary?

Answer

If the entire 4 gigabyte address space is defined then a default slave is not required. If, however, there are undefined areas in the memory map then it is important to ensure that a spurious access to a non-existent address location will not lock up the system. The functionality of the default slave is extremely simple and it will often make sense to implement this within the decoder.

Is a dummy master really necessary?

Answer

A dummy master is necessary in any system which has a slave that can give SPLIT transfer responses. The dummy master is required so that something can be granted the bus if all the other masters have received a SPLIT response.

No logic is required for the dummy master and it can be implemented by simply tying off the inputs to the master address/control multiplexer for the dummy master position. The requirements for a dummy master are that HTRANS is driven to IDLE, HLOCK is driven low, and all other master outputs are driven to legal values.

Is it legal for a master to change HADDR when a transfer is extended?

Answer

If a master is indicating that it wants to do a NONSEQ, SEQ or BUSY transfer then it cannot change the address during an extended transfer (when HREADY is low) unless it receives an ERROR, RETRY or SPLIT response. If the master is indicating that it wants to do an IDLE transfer then it may change the address.

Is it specified that HPROT, HSIZE and HWRITE remain constant throughout a burst?

Answer

Yes, the control signals must remain constant throughout the duration of a burst.

The specification recommends that only 16 wait states are used. What should you do if more than 16 cycles are needed?

Answer

For some slaves it is acceptable to insert more than 16 wait states. For example, a serial boot ROM which is only ever accessed at initial power up could insert a larger number of wait states and it would not affect the calculation of the system performance and latency once system power up has been completed.

For other slaves a number of options exist. A SPLIT or RETRY response could be used to indicate that the slave is not yet able to perform the requested data transfer, or the slave could be accessed either in response to interrupts or after polling a status register, in either case indicating that the slave is now able to respond in an acceptable number of cycles.

What are the different bursts used for?

Answer

Typically a master would use wrapping bursts for cache line fills where the master wants to access the data it requires first and then it completes the burst to fetch the remaining data it requires for the cache line fill. Incrementing bursts are used by masters, such as DMA controllers, that are filling a buffer in memory which may not be aligned to a particular address boundary.

What default state should be used for the HREADY and HRESP outputs from a slave?

Answer

It is recommended that the default value for HREADY is high and the default value for HRESP is OKAY. This combination ensures that the slave will respond correctly to IDLE transfers to the slave, even if the slave is in some form of power saving mode.

What is the difference between a dummy bus master and a default bus master?

Answer

The term default bus master is used to describe the master that is granted when none of the masters in the system are requesting access to the bus. Usually the bus master which is most likely to request the bus is made the default master.

The dummy bus master is a master which only performs IDLE transfers. It is required in a system so the arbiter can grant a master which is guaranteed not to perform any real transfers. The two cases when the arbiter would need to do this are when a SPLIT response is given to a locked transfer and when a SPLIT response is given and all other masters have already been SPLIT.

What is the recommended default value for HPROT?

Answer

Many bus masters will not be able to generate accurate protection information and for these bus masters it is recommended that the HPROT encoding shows, Non-cacheable, Non-bufferable, Privileged, Data Accesses which corresponds to HPROT[3:0] = 4'b0011.

What is the state of the AHB signals during reset?

Answer

The specification states that during reset the bus signals should be at valid levels. This simply means that the signals should be logic '0' or '1', but not Hi-Z. The actual logic levels driven are left up to the designer. HTRANS is the only signal specified during reset, with a mandatory value of IDLE.

It is important that HREADY is high during reset. If all slaves in the system drive HREADY high during reset then this will ensure that this is the case. However, if slaves are used which do not drive HREADY high during reset it should be ensured that a slave which does drive HREADY high is selected at reset.

What sequences of transfers types (HTRANS) can occur on the bus?

Answer

The following examples show some of the sequences of HTRANS that can occur on the bus:

A large burst of four transfers followed by an IDLE.

N - S - S - S - I

A large burst of four transfers which includes BUSY transfers.

N - S - B - S - B - S - I

A burst of four transfers followed by another burst.

N - S - S - S - N - S - S - S - I

A single transfer followed by a burst of four transfers.

N - N - S - S - S - I

A single transfer followed by an IDLE

N - I

An undefined length burst which concludes with a BUSY transfer.

N - B - S - B - S - B - I

An undefined length burst which concludes with a BUSY transfer and is followed immediately by another burst.

N - B - S - B - S - B - N – S

When a master rebuilds a burst which has been terminated early are there any limitations on how it rebuilds the burst?

Answer

The only limitation is that the master uses legal burst combinations to rebuild the burst. For example, if a master was performing an 8 beat burst, but had only completed 3 transfers before losing control of the bus, then the remaining 5 transfers could be performed either by using a 1 beat SINGLE burst followed by a 4 beat INCR4 burst, or it could be performed using a 5 beat undefined length INCR burst.

For simplicity it is recommended that masters use INCR bursts to rebuild the remaining transfers.

How does the AHB handle LOCKed SPLITs?

Answer

When a transfer is SPLIT the arbiter degrants and removes the SPLIT master out of the arbitration until the slave indicates that the transfer can complete. When an access is LOCKed the access cannot be interrupted by an access from another master.

The only possible way that an AHB system can handle these two requirements simultaneously is to grant a "dummy master" when the LOCKed access is SPLIT. The dummy master will only perform IDLE transactions, which are allowable during a locked transfer. To grant any other master would violate the LOCK protocol, for the arbiter to ignore the SPLIT would violate the SPLIT protocol - the dummy master is the only option.

The dummy master is also used when all masters are have received a SPLIT response (the dummy master cannot receive a SPLIT response).

It is recommended that the designer of the split-capable slave(s) makes sure that the slave monitors its HMASTLOCK input so that it doesn't return a SPLIT on a LOCKed transfer, as this serves no purpose.

How many clock cycles should the reset signal in an AMBA system be asserted for?

Answer

It is recommended that master and slave components should clearly state if they have a reset requirement greater than 1 or 2 cycles. It is also recommended that the system design should hold reset asserted for at least 16 cycles, unless it is known that a master or slave component has a longer reset requirement.

Is it legal for an AHB wrapping burst to be aligned with respect to the total number bytes in the burst, such that it does not wrap?

Answer

Yes, this behavior is compliant with the AHB protocol.

Consider a four-beat wrapping burst of word (4-byte) transfers (which will wrap at 16-byte boundaries).

If the start address of the transfer is 0x30, then the burst consists of four transfers to addresses 0x30, 0x34, 0x38, and 0x3C.

Again, although HBURST is set to WRAP4, the burst will not actually wrap, which is allowed.

When should a master assert and deassert the HLOCK signal for a locked transfer?

Answer

The HLOCK signal must be asserted at least one cycle before the start of the address phase of a locked transfer. This is required so that the arbiter can sample the HLOCK signal as high at the start of the address phase.

The master should deassert the HLOCK signal when the address phase of the last transfer in the locked sequence has started.

When should a master deassert its HBUSREQ signal?

Answer

For an undefined length burst (INCR) a master must keep its HBUSREQ signal asserted until it has started the address phase of the last transfer in the burst. This will mean that if the penultimate transfer in the burst is zero wait state then the master may be granted the bus for an additional transfer at the end of an undefined length burst.

For a defined length burst the master can deassert the HBUSREQ signal once the master has been granted the bus for the first transfer. This can be done because the arbiter is able to count the transfers in the burst and keep the master granted until the burst completes.

However it is not a mandatory requirement for an Arbiter to allow a burst to complete, so the master will have to re-assert HBUSREQ if the Arbiter removes HGRANT before the burst has been completed.

When will the arbiter grant another master after a locked transfer?

Answer

The arbiter will always grant the master an extra transfer at the end of a locked sequence, so the master is guaranteed to perform one transfer with the HMASTLOCK signal low at the end of the locked sequence. This coincides with the data phase of the last transfer in the locked sequence.

During this time the arbiter can change the HGRANT signals to a new bus master, but if the data phase of the last locked transfer receives either a SPLIT or RETRY response then the arbiter will drive the HGRANT signals to ensure that either the master performing the locked sequence remains granted on the bus for a RETRY response, or the Dummy master is granted the bus for the SPLIT response.

Can a master deassert HLOCK during a burst?

Answer

The AHB specification requires that all address phase timed control signals (other than HADDR and HTRANS) remain constant for the duration of a burst.

Although HLOCK is not an address phase timed signal, it does directly control the HMASTLOCK signal which is address phase timed.

Therefore HLOCK must remain high for the duration of a burst, and can only be deasserted such that the following HMASTLOCK signal changes after the final address phase of the burst.

Can a master perform transfers other than IDLE when the bus was granted to it, but not requested by the master?

Answer

Yes. A master can perform transfers other than IDLE when it had not requested the bus. Please note that in this case it is still recommended that the master asserts its request signal so that the arbiter does not change ownership of the bus to a lower priority master while the transfers are in progress.

If a master is currently granted the bus by default, how many cycles before starting an non-IDLE transfer does it have to assert HBUSREQ?

Answer

None. It can start a non IDLE transfer immediately.

What is the relationship between the HLOCK signal and the HMASTLOCK signal?

Answer

At the start of the address phase of every transfer the arbiter will sample the HLOCK signal of the master that is about to start driving the address bus and if HLOCK is asserted at this point then HMASTLOCK will be asserted by the arbiter for the duration of the address phase of the transfer.

When can the HGRANT signal change?

Answer

The HGRANT signal can change in any cycle and the following cases are possible:

- It is possible that the HGRANT signal may be asserted and then removed before the current transfer completes. This is acceptable because the HGRANT signal is only sampled by masters when HREADY is high.
- A master can be granted the bus without requesting it.
- The above point also means that it is possible to be granted the bus in the same cycle that it is requested. This can occur if the master is coincidentally granted the bus in the same cycle that it requests it.

Why is HADDR sometimes shown as an input to the arbiter?

Answer

The address bus, HADDR, is not required as an input to the arbiter but in some system designs it may be useful to use the address bus to determine a good point to change over between bus masters. For example, the arbiter could be designed to change bus ownership when a burst of transfers reaches a quad word boundary.

Can an arbiter be designed to always allow bursts to complete?

Answer

A SPLIT, RETRY or ERROR response from a slave can always cause a burst to be early terminated. This is outwith the control of the Arbiter and so must be supported.

Undefined length INCR bursts cannot have their end point predicted, so there is no efficient way that an Arbiter design can allow the burst to complete before granting another master. INCR bursts must be arbitrated on a cycle by cycle basis.

Defined length INCRx and WRAPx bursts can have their beats counted, and so allowed to complete by the Arbiter. However because of the AHB arbitration synchronous timing, there is no way to avoid possibly terminating a burst immediately after the first transfer of the burst has been indicated.

The Arbiter only knows that a defined length burst is in progress by sampling the HBURST bus. However the first point at which HBURST can be sampled is after the first clock cycle of the first burst beat, by which time the Arbiter may already have decided to grant another master and will have changed the HGRANT outputs accordingly. Only a combinatorial path from HBURST to HGRANT would allow the burst to be detected in time to avoid early termination in this scenario, but combinatorial paths in the AHB bus are not allowed.

Why is there a 1KB restriction in AHB?

Answer

The 1KB restriction you refer to is not a restriction on maximum slave size but a constraint within AHB that says that a burst must not cross a 1KB boundary. The limit is designed to prevent bursts crossing from one device to another and to give a reasonable trade-off between burst size and efficiency. In practise, this means

that a master must ALWAYS break a burst that would otherwise cross the 1KB boundary and restart it with a non-sequential transfer, thus:

Address: 0x3F0 0x3F4 0x3F8 0x3FC 0x400 0x404 0x408

Transfer: NSEQ SEQ SEQ SEQ NSEQ SEQ SEQ

What's the difference between retry and split in AHB?

Answer

The SPLIT and RETRY responses provide a mechanism for slaves to release the bus when they are unable to supply data for a transfer immediately. Both mechanisms allow the transfer to finish on the bus and therefore allow a higher-priority master to get access to the bus.

When a master initiates a transaction on the AMBA bus, if the target detects that the transfer will take a large number of cycles to perform, it can issue a SPLIT signal. What happens now is that the arbiter can grant the bus to other masters even before the SPLIT transaction is complete. The master to which the SPLIT has been issued has to then wait and complete the entire transaction.

During the address phase of a transfer the arbiter generates a tag, or bus master number, on HMASTER[3:0] which identifies the master that is performing the transfer. Any slave issuing a SPLIT response must be capable of indicating that it can complete the transfer, and it does this by making a note of the master number on the HMASTER[3:0] signals.

Later, when the slave can complete the transfer, it asserts the appropriate bit, according to the master number, on the HSPLITx[15:0] signals from the slave to the arbiter. The arbiter then uses this information to unmask the request signal from the master and in due course the master will be granted access to the bus to retry the transfer. The arbiter samples the HSPLITx bus every cycle and therefore the slave only needs to assert the appropriate bit for a single cycle in order for the arbiter to recognize it.

The basic stages of a SPLIT transaction are:
1. The master starts the transfer in an identical way to any other transfer and issues address and control information
2. If the slave is able to provide data immediately it may do so. If the slave decides that it may take a number of cycles to obtain the data it gives a SPLIT transfer response. During every transfer the arbiter broadcasts a number, or tag, showing which master is using the bus. The slave must record this number, to use it to restart the transfer at a later time.
3. The arbiter grants other masters use of the bus and the action of the SPLIT response allows bus master handover to occur. If all other masters have also received a SPLIT response then the default master is granted.
4. When the slave is ready to complete the transfer it asserts the appropriate bit of the HSPLITx bus to the arbiter to indicate which master should be regranted access to the bus.
5. The arbiter observes the HSPLITx signals on every cycle, and when any bit of HSPLITx is asserted the arbiter restores the priority of the appropriate master.
6. Eventually the arbiter will grant the master so it can re-attempt the transfer. This may not occur immediately if a higher priority master is using the bus.
7. When the transfer eventually takes place the slave finishes with an OKAY transfer response.

For a SPLIT transfer the arbiter will adjust the priority scheme so that any other master requesting the bus will get access, even if it is a lower priority. In order for a SPLIT transfer to complete the arbiter must be informed when the slave has the data available.

For RETRY the arbiter will continue to use the normal priority scheme and therefore only masters having a higher priority will gain access to the bus.

What value should be used for HTRANS when an AHB master gets a RETRY response from a slave in the middle of burst?

Whenever a transfer is restarted it must use HTRANS set to NONSEQ and it may also be necessary to adjust the HBURST information (usually just to indicate INCR).

What address should be on the bus during the IDLE cycle after a SPLIT or RETRY?

It does not matter what address is driven onto the bus during this cycle. The slave selected by the driven address should not take any action and must respond with a zero wait state OKAY response.

In many cases it will be simpler for the master to leave the address unaltered during this cycle, so that it remains at the address of the next transfer that the master wishes to perform and only in the following cycle does the master return the address to that of the transfer that must be repeated because of the SPLIT or RETRY response.

In some designs it may be possible for the master to return the address to that required to repeat the previous transfer during the IDLE cycle and this behaviour is also perfectly acceptable.

Do all masters have to support SPLIT and RETRY?

Yes. All masters must support SPLIT and RETRY responses to ensure they are

compatible with any bus slave. A master will handle both SPLIT and RETRY responses in an identical manner.

Can a SPLIT or RETRY response be given at any point during a burst?

Yes. A SPLIT, RETRY or ERROR response can be given by a slave to any transfer during a burst. The slave is not restricted to only giving these responses to the first transfer.

Will a master always lose the bus after a SPLIT response?

Yes. A slave must not assert the relevant bit of the HSPLIT bus in the same cycle that it gives the SPLIT response and therefore the master will always lose the bus.

Can a slave assert HSPLITx in the same cycle that it gives a SPLIT response?

No. The specification requires that HSPLITx can only be asserted after the slave has given a SPLIT response.

Do all slaves have to support the SPLIT and RETRY responses?

No. A slave is only required to support the response types that it needs to use. For example, a simple on-chip memory block which can respond to all transfers in just a few wait states does not need to use either the SPLIT or RETRY responses.

Can a slave use both SPLIT and RETRY responses?

Normally a slave will not use both the SPLIT and RETRY responses. The SPLIT response should be used by any slave that may be accessed by many different masters at the same time. The RETRY response is intended to be used by peripherals that are only accessed by one bus master.

Explain Lock Transfer in AHB ?

Answer

LOCK tells the arbiter to keep the current master granted, SPLIT tells the arbiter to grant another master, so the only possible action the arbiter can take for these contradictory requests is to grant the dummy master that must exist in any system with SPLIT capable slaves.

The dummy master will only perform IDLE transfers (i.e. no data transfers), so cannot corrupt the LOCKed sequence that is ongoing.

When the original slave is able to complete the SPLIT transfer, it will signal this to the arbiter on HSPLIT and the arbiter can then re-grant the original master, and the LOCKed sequence can then continue.

However as the slave has an HMASTLOCK input telling it that the current transfer is part of a LOCKed sequence, it should know that there is no system advantage in returning a SPLIT.

So yes, a slave can return a SPLIT response to a LOCKed transfer, and the arbiter must then grant the dummy master, but the slave should use the HMASTLOCK input to see that a SPLIT response is not useful at this time.

Explain Wrap Boundary Calculation in AHB ? WRAP4,WRAP8,WRAP16 ?

Answer

Wrap boundary depends on both Hsize and the No of beats(4 ,8,16)

"**For wrapping bursts, if the start address of the transfer is not aligned to the total number of bytes in the burst (size x beats) then the address of the transfers in the burst will wrap when the boundary is reached**"

Case1: Start Address is 0x4,Wrap4,Hsize is 2.

0000 0100 - beat1 - 0x4

0000 1000 - beat2 - 0x8

0000 1100 - beat3 - 0xc

0000 0000 - beat4 - 0x0

Here as Hsize is 2,it means 4 bytes are to be transfered. As it is Wrap4 no of beats are 4. Total no of bytes is 16(beats*Hsize in bytes). Therefore 4bit alignment is to be done.

In beat3 the address is 0000 1100.Now as Hsize is 2,address shld be incremented by 4.

0000 1100

+0000 0100

0001 0000 (it is crossing the address boundary(4 bits))

So we are aligning it to 0000 0000.-beat4

Case2: Start Address is 0x4,Wrap4,Hsize is 1.

0000 0100 - beat1 - 0x4

0000 0110 - beat2 - 0x6

0000 0000 - beat3 - 0x0

0000 0010 - beat4 - 0x2

Here as Hsize is 1,it means 2bytes are to be transfered. As it is Wrap4 no of beats are 4. Total no of bytes is 8(beats*Hsize in bytes).

Therefore 3 bit alignment is to be done. In beat2 the address is 0000 0110. Now as Hsize is 1,address shlould be incremented by 2.

0000 0**110**

+0000 0010

0000 1**000** (it is crossing the address boundary(3 bits).)

So we are aligning it to 0000 0000.-beat3

Case3: Start Address is 0x4,Wrap8,Hsize is 1.

0000 0100 - beat1 - 0x4

0000 0110 - beat2 - 0x6

0000 1000 - beat3 - 0x8

0000 1010 - beat4 - 0xa

0000 1100 - beat5 - 0xc

0000 1110 - beat6 - 0xe

0000 0000 - beat7 - 0x0

0000 0010 - beat8 - 0x2

Here as Hsize is 1,it means 2 bytes are to be transfered.

As it is Wrap8 no of beats are 8.

Total no of bytes is 16 (beats*Hsize in bytes).

Therefore 4bit alignment is to be done.

In beat6 the address is 0000 1110.Now as Hsize is 1,address shld be incremented by 2.

0000 1110

+0000 0010

0001 0000 (it is crossing the address boundary(4 bits)).

So we are aligning it to 0000 0000.-beat7

Another Wrap Boundry Calculation :

1) convert hex addr to decimal - 0x38 = 56 (dec)
2) as its a word transfer (4 bytes) and WRAP 4 so 4 x 4 = 16
3) 56 / 16 = 3.5
4) take the whole number from previous calculation as 3
5) now 3 x 16 = 48 (dec)
6) your roll over addr is 48 (dec) = 0x30 (hex) !!!

so the address goes as 38 - 3c - 30 and 34

Exaplain Wrap Beat Calulation in AHB ?

Answer

Following Tasks will Give Information About Wrap Boundary Beat Location Calculation

```
//-------------------------------------------------------
// wrap4_beat_info()
//-------------------------------------------------------
task wrap4_beat_info (logic [2:0] hburst,logic [31:0] haddr,logic [2:0] hsize);
  if(hsize==3'b010 && hburst==`AHB_WRAP4 )
   begin
         if(haddr[3:2]==2'b00) wrap4_boundry_location=0 ; // No Wrap
    else if(haddr[3:2]==2'b01) wrap4_boundry_location=3 ; // Wrap at 3rd Beat
    else if(haddr[3:2]==2'b10) wrap4_boundry_location=2 ; // Wrap at 2nd Beat
    else if(haddr[3:2]==2'b11) wrap4_boundry_location=1 ; // Wrap at 1st Beat
   end
endtask : wrap4_at_info
//-------------------------------------------------------
// wrap8_beat_info()
//-------------------------------------------------------
task wrap8_beat_info (logic[2:0] hburst,logic[31:0] haddr,logic[2:0] hsize);
  if(hsize==3'b010 && hburst==`AHB_WRAP8 )
   begin
         if(haddr[5:3]==3'b000) wrap8_boundry_location=0 ;
    else if(haddr[5:3]==3'b001) wrap8_boundry_location=7 ;
    else if(haddr[5:3]==3'b010) wrap8_boundry_location=6 ;
    else if(haddr[5:3]==3'b011) wrap8_boundry_location=5 ;
    else if(haddr[5:3]==3'b100) wrap8_boundry_location=4 ;
```

```
    else if(haddr[5:3]==3'b101) wrap8_boundry_location=3 ;
    else if(haddr[5:3]==3'b110) wrap8_boundry_location=2 ;
    else if(haddr[5:3]==3'b110) wrap8_boundry_location=1 ;
  end
endtask : wrap8_beat_info
//-------------------------------------------------------
// wrap16_at_info()
//-------------------------------------------------------
task wrap16_beat_info (logic[2:0] hburst, logic[31:0] haddr,logic[2:0] hsize);
  if(hsize==3'b010 && hburst==`AHB_WRAP16 )
   begin
         if(haddr[7:4]==4'b0000) wrap8_boundry_location=0  ;
    else if(haddr[7:4]==4'b0001) wrap8_boundry_location=15 ;
    else if(haddr[7:4]==4'b0010) wrap8_boundry_location=14 ;
    else if(haddr[7:4]==4'b0011) wrap8_boundry_location=13 ;
    else if(haddr[7:4]==4'b0100) wrap8_boundry_location=12 ;
    else if(haddr[7:4]==4'b0101) wrap8_boundry_location=11 ;
    else if(haddr[7:4]==4'b0110) wrap8_boundry_location=10 ;
    else if(haddr[7:4]==4'b0111) wrap8_boundry_location=9  ;
    else if(haddr[7:4]==4'b1000) wrap8_boundry_location=8  ;
    else if(haddr[7:4]==4'b1001) wrap8_boundry_location=7  ;
    else if(haddr[7:4]==4'b1010) wrap8_boundry_location=6  ;
    else if(haddr[7:4]==4'b1011) wrap8_boundry_location=5  ;
    else if(haddr[7:4]==4'b1100) wrap8_boundry_location=4  ;
    else if(haddr[7:4]==4'b1101) wrap8_boundry_location=3  ;
    else if(haddr[7:4]==4'b1110) wrap8_boundry_location=2  ;
    else if(haddr[7:4]==4'b1111) wrap8_boundry_location=1  ;
  end
endtask : wrap16_beat_info
```