**INSTRUCTIONS FOR FILLING OUT AND SUBMITTING THE INVENTION DISCLOSURE FORM (IDF)**

The instructions below are intended for inventor-initiated IDF submissions. If this IDF is from a patent harvest, you should receive instructions for completing and submitting the IDF via email.

**CHECKLIST FOR IDF SUBMISSIONS:**

***Fully*** complete all sections. Inventor names must be FULL LEGAL NAMES as they appear on a valid Driver’s License or Passport. This is the name attorneys would use to submit a patent application. Do not use nicknames, shortened names, or swap middle/first names.

***Each*** Inventor to sign the IDF.

IDF to be reviewed, understood and signed by two non-inventor, AMD employee witnesses (their signatures indicate they understood the contents of the document). The witnesses are to also **initial each page**.

Submit an electronic copy of the completed IDF in an email to “AMD IDFs” ( idf@amd.com ) **AND**

Submit the original signed and witnessed paper copies to AMD Legal at:

(if using internal mail):

**Mailstop 10CV 4N 4N4054**

(if using external mail or courier):

**33 Commerce Valley Drive East**

**M/S 4N4055**

**Markham, ON L3T7X6 Canada**

**NOTE:**

Each inventor may sign and submit a separate copy. If there are multiple signed copies of the IDF, each signed original paper copy should be mailed to AMD Legal. Only one electronic copy should be submitted to AMD Legal via email.

ATTENTION: This document is confidential, subject to SOLICITOR/ATTORNEY-CLIENT PRIVILEGE, and is being submitted to the AMD Legal Department. Distribution of this document should be *strictly limited* to AMD Legal and others at AMD supporting AMD Legal including relevant outside counsel.

A patent application for this invention must be filed at an appropriate Patent Office prior to disclosing the invention publicly or AMD will NOT be able to file a patent application in most countries. In the US there are potential exceptions in that a patent application can be filed at the US Patent Office *no later than one year* from (a) the first public disclosure of the invention, (b) the first offer for sale of a product implementing the invention, or (c) the first secret use of the invention in production.

|  |
| --- |
| **A. BRIEF DESCRIPTION OF THE INVENTION** |

# WORKING TITLE: Hardware mechanism for global synchronization for gpu

(Select a title for the invention that is short, non-limiting and describes the main feature of your idea.)

|  |
| --- |
| 1. Provide a “nutshell” summary of your invention in five sentences or less. The description should avoid implementation-specific details, and instead should focus on the broader concept of the invention. |
| Many-core processors like GPU usually execute many threads in parallel to achieve high throughput. Synchronization is often required to coordinate the threads. Existing GPUs provide synchronization for thread group on the same core but synchronizations among cores are leftto programmers through software mechanisms. Software synchronizations, which in nature are serial operations, have the drawback of low parallelism and long latency. Inefficient synchronization can result in performance degradation by keeping the cores waiting. This invention proposes a hardware-based global synchronization mechanism which enables efficient parallel synchronization between different cores. |

2. Select what you believe to be the most relevant PAC, select from pull-down menu. (Definitions located at the [Inventor's Corner - After you submit IDF](http://amdcentral.amd.com/AMDTeams/Corporate/Legal/InventorsCorner/Pages/AfteryousubmitIDF.aspx)) Central Engineering Technologies

|  |  |
| --- | --- |
| 3. Internal/public funded R&D project name/#. If this submission is from a patent harvest, note the attending group and date of session. | |
| Research, 7887530 |

|  |  |
| --- | --- |
| 4. **AMD Use**: To your knowledge, has a product including the invention been (1) offered for sale (i.e. have binding offers for the product been placed) or (2) disclosed or scheduled for disclosure to anyone outside AMD?  YES  NO | |
| If yes, on what **date** was the product first offered for sale or disclosed/planed for disclosure? |  |

|  |
| --- |
|  |

IF, AT ANY TIME PRIOR TO FILING A PATENT APPLICATION, YOU BECOME AWARE THAT THE ANSWER TO QUESTION 4 WILL CHANGE, YOU MUST ADVISE AMD LEGAL ASAP.

|  |
| --- |
| 5. State the **problem solved** by the invention. |
| As GPU extends to cover general purpose computing, more synchronization and coordination patterns will show up. However the existing GPU architectures don’t provide efficient solution. So far, only efficient local synchronization within one workgroup is supported and global synchronization relies on software methods. As a result, the breadth of computation that can be efficiently supported on a GPU has largely been limited to highly data parallel or task parallel applications. In more general situations like reducing, k-mean, Gibbs sampling etc., applications often need all or several threads to share their output information in order to continue the computing despite of their data parallelism. So several software methods using global variables such as flags are introduced to do the synchronization to ensure global threads can read the correct data from global memory.  However, those software synchronization methods have to pay considerablely high price for following reasons. Firstly, those software methods often have to use chip-off memory as the flags or counters, which lead to high latency to read and write. Secondly, in some methods, atomic operations have to be applied to change flags or accumulate counters, resulting in data race and long time waiting. What’s more, some methods make use of kernel terminating mechanism to do synchronization. So there are extra overhead of invoking kernels repeatedly and transferring via PCI-E bus. In one word, software synchronizations are comparable low efficient and block GPU to be applied to more general situations.  This invention proposes an on-chip hardware synchronization mechanism, which can be used for efficient synchronization between cores. The synchronization unit includes a light register-vector for each core to track the critical information including flag, counter etc., and a destination mask which indicates the list of cores that have dependence on the current core. Different synchronization units can communicate with each to update the critical information and synchronization decision can be made locally without waiting for a global signal. |

|  |
| --- |
| 6. What are the **known relevant existing solution(s)** to the problem (e.g. current solutions known to exist in literature or products)? Please explain why you believe they are insufficient or inferior to your invention.  IMPORTANT: Failure to disclose material prior art that you are aware of (or that you become aware of in the future) to the patent office may invalidate the patent. |
| Nowadays there are 3 main known software synchronization mechanisms: API based, atomic counter based and memory flag based. The following paragraphs introduce how they work and their drawbacks.   1. Naïve synchronization by CPU calling API   Obviously, when kernel finishes executing, all the threads will come to the same point. Taking advantages of it, all the threads can be synchronized when kernel finishes. And kernel has to be invoked again to continue. This method introduces in significant overhead of invoking repeatedly and transferring via PCI-E bus.   1. GPU atomic counter synchronization   The threads in one workgroup can be synchronized by existing barrier synchronization. And a volatile global variable works as a counter. When all threads in one workgroup have come to synchronization point, one thread in the workgroup will atomically add one into the counter. Then the workgroup wait for counter being accumulated into the number of workgroups to continue execution.  In this method, the atomic operation means data race and global memory means high latency to access. Although the tree based synchronization (namely several volatile global counter works in the form of tree instead of only one global counter) can alleviate it, the performance suffers from using atomic operation and access to global memory frequently.   1. GPU lock-free synchronization   Lock-free synchronization has avoided all the low efficient atomic operations, but it employs a flag array in global memory whose length is the number of workgroups. Within one workgroup threads still use barrier to synchronize and workgroups that have come to the point will set its own element in the array into a specific value. Then a special workgroup will be assigned to check the total array and all the workgroups having come to the point have to wait for the special workgroup to set another array to a special value to continue execution. This method can be described as following:  Global arrayIn[workgroupNum]=0, arrayout[workgroupNum]=0;  If(local\_Id==0)  arrayIn[workgroupID]=1;  If(WorkgroupID==0){  if(local\_id<workgroupNum)  while(arrayIn[local\_id]!=1);  barrier();  if(local\_id<workgroupNum)  arrayOut[local\_id]=1;  }  If(local\_Id==0)  while(arrayOut[workgroupID]!=1);  Barrier();  Although the slow atomic operations have been avoided in lock-free synchronization, frequently checking on global flag array still bring high latency. As the known fastest GPU global synchronization, it still comparable slower than a mechanism implemented well on hardware level. |

|  |
| --- |
| **Describe your invention** in a manner that is sufficient to enable someone with basic skill in this field to fully understand the invention and its operation. Do not limit your description to your preferred implementation; instead, include as many possible alternatives or modifications that you can imagine a competitor might use. |
| NOTE: Please use appropriate drawings (e.g., system, circuit, timing, or flow diagrams). Hand-drawn sketches are acceptable. You may also attach additional documents to this IDF so long as they are referenced below. |
| This invention proposes a hardware-based global synchronization mechanism which enables efficient parallel synchronization between different cores. The basic idea of the proposal is to track the synchronization flag and dependency between cores in hardware, and update the flag to the dependent cores whenever it is ready. Synchronization decisions can be made locally once the core receives all the expecting flags. Future many-core processors can integrate the synchronization unit on chip to replace the inefficient software synchronizations through API calls or off-chip memory operations.   1. **H/W synchronization unit**   The synchronization unit includes a light register-vector for each core to track the critical information including flag, counter etc., and a destination mask which indicates the list of cores that have dependence with the current core. Different synchronization units can communicate with each other to update the critical information and synchronization decision can be made locally without waiting for a global signal.  Synchronization unit can sit next to the on-chip interconnectin order to communicate with each other, options can be core’s private L1 cache or shared/private L2 cache. Synchronization flags can be sent through the cache ports to each other. Once current core reaches the synchronization point, it first sends out the “finished” status information to all the cores that are depended on its output data. Then all the receivers will update the flag of corresponding core in their synchronization unit. The core checks if the received\_flags have met the requirement of dependency determined by the pattern unit. For example, core A requires the output of core B, C, D. If the pattern mode indicates that at least two inputs have to be available and the received\_flags of core B and C are set, core A can continue the computation. If the pattern mode indicates an “and” operation which requires all the inputs to be ready, then core A still needs to wait for core D’s synchronization message.  The hardware components of synchronization units are as shown in Fig1: an entry table to track the synchronization information and a packet generator to generate packets (containing synchronization message) and send them out through the out port, a packet receiver to receive synchronization messages from the network. A detailed description of the synchronization unit is as following.    Fig 1 H/W synchronization unit for each GPU processor  GroupID: the ID of the thread group (workgroup for AMD’s GPU)  OutFlag: the synchronization flag of the current GroupID used for synchronization, can be a counter, 0/1 flag, a specific number. Hardware synchronization signal will be sent out once the OutFlag is set.  OutMask: indicate a list of other GroupID s which are waiting for the current OutFlag in order to synchronize, eg, GroupID 1,2,3,4 are dependent on this local flag. Hence, a synchronization message is sent out to GroupID 1,2,3,4 when OutFlag is set.  InFlags: Synchronization flags received from the network. InFlag includes the value of the flag and the senderID who sends the flag. In practice it is common that there are more than one flags. So the InFlags will be a vector.  InMask: indicate a list of other works groups whose flags the current GroupID needs in order to synchronize. eg, current Group needs the flags from thread group with ID 5,6,7,8 to synchronize. Only the sync flags coming from the GroupID in the list will be received by the current group.  Pattern: decision logic, equal (received flag=local flag), all clear, all set, larger  Sync: Synchronization decision made based on the received InFlags and pattern. A simple way to do it in hardware is whenever a new received InFlag, hardware will check all the InFlags and the pattern to see if it satisfies the sync conditions.  Packet generator logic: generate packets to other groups based on the out\_mask, “set” indicates the message has been sent the network.  Packet receive logic: snoop the synchronization messages on the network and only receive packets indicated the current GroupID and then records the flag in the InFlags.  The synchronization messages are placed in the specific queue in the network interface or a dedicated Virtual Channel so that they won’t be blocked other data/control message to ensure that the synchronization units are updated in the same order.  Each GPU core should have one synchronization unit and deals with the synchronization for all the workgroups mapped on the core. Limited by GPU core resources (local memory, register files etc.), the number of workgroups on one GPU core is usually from a few to a dozen. Thus each synchronization units should be designed have enough entries to hold information for all the workgroups.   1. **H/W synchronization mechanism**   Based on the structure of the synchronization units, the hardware mechanism can be simply implemented by two major steps: Programmer initializes the synchronization entry; Set up the synchronization flag during program run, and synchronization units will send out synchronization messages and draw synchronization decisions.  **Step one: initialization of synchronization unit**  Initially, the content of synchronization unit is clear. The programmers are in charge to setup the data dependency table if global-synchronization is necessary. The hardware synchronization is optimal to programmer. Original software can still be used when synchronization does not affect performance badly.  The synchronization table can be accessed as memory mapped device. It may require the runtime to first allocate a table entry for each workgroup. The allocated address will be read or write as volatile type. Initialization can be done by writing to the entry variables(GroupID, OutMask, InMask, Pattern with the rest of fields cleared). Hardware will be directed the writes to the synchronization entry of the workgroup. If the data dependency among the cores changes in the program, the masks can be also updated during the execution.  **Step two: communication of synchronization units**  During the program run, the OutFlag is set by programmer. Once the flag is set, the packet generator will generate the message to sent the new flag to the GroupIDs that specified by OutMask. Meanwhile all the active entries will snoop the network to receive flags from other Groups and only receive the flags that match their GroupID. Whenever a new InFlag is received, the InFlags and the Pattern will be checked and to check whether make synchronization decision can be made. The programmer can read (pull) the sync signal to get synchronization decision. As potential optimization, the sync signal could be used by the task queue scheduling to speedup the scheduling process as soon as the synchronization decision is made.     1. **Deadlock avoidance**   When too many workgroups are mapped to one GPU core, the synchronization entry table will become a resource bottleneck and can result in deadlock. In order to avoid the deadlock, the synchronization can fall back to the existing software mechanisms. Runtime could help detect whether the synchronization entries are running out and force the synchronization go back to software.   1. **Hardware complexity**   The hardware complexity lies in majorly the storage overheads of the entries (a dozen also should be enough for each GPU core). Second, communication between synchronization units needs the trigger logic based on the flags, the logic to receive the incoming signals from the network, generate packets to other units based on the destination mask and network support to send the packets. The control logic is pretty straightforward and simple. Third, some runtime modification is needed to support access to synchronization unit as memory mapped device. But the major part of the runtime related how the program runs stays the same. |

|  |
| --- |
| 8. Describe the **potential advantages** of the invention (compared to known art). |
| Existing GPUs provide synchronization for thread group on the same core but synchronizations among cores are left to programmers through software mechanisms. Software synchronizations, which in nature are serial operations in main memory, have the drawback of low parallelism and long latency. Inefficient synchronization can result in performance degradation by keeping the cores waiting. This invention proposes a hardware-based global synchronization mechanism which enables efficient parallel synchronization between different cores. Each GPU core is extended with a synchronization unit which can deal with the synchronization of all the workgroups that are mapped on the GPU core. This synchronization works by tracking the synchronization flags and exchanging flags between different GPU cores. Synchronization decisions can be made locally on chip within each GPU core once received all the expecting flags. Future many-core processors can integrate the synchronization unit on chip to replace the inefficient software synchronizations. |

|  |
| --- |
| **B.** **GENERAL INFORMATION ABOUT THE INVENTION** |

9. Use the pull-down menu associated with the lead inventor’s to select the single most relevant technology Domain, select from pull-down menu.. More Domain information such as CTO, Architect, and Business Owner can be found at [Products Group Domains](http://amdcentral.amd.com/AMDTeams/ProductsGroup/Documents/Forms/AllItems.aspx?RootFolder=%2fAMDTeams%2fProductsGroup%2fDocuments%2fDomains&FolderCTID=&View=%7b796C7677-2926-464F-BD54-0A5F206901FE%7d): APU & GPU Power - Optimization management, Methodology/Flows

10. If applicable, what is the marketing or sales area for your technology (so that we can investigate marketing and/or sales plans? Select from pull-down menu. Choose an item..

#### 11. On what date did you first conceive the invention? \_05/22/2013\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### (The date you formed an idea of the invention that was expected to be operative?)

#### 12. On what date was a document describing this invention first drafted? 05/28/2013\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(e.g. technical document, drawing, or email. Please attach a copy to this IDF. If this IDF is the first description, then use today’s date.)

#### 13. If applicable, on what date was this invention first built/implemented? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(This is the date an embodiment of the invention was built or simulated and sufficiently tested to demonstrate that it works as intended. **Note: There is no requirement that this has occurred for you to submit this form or file a patent application.**)

14. Name(s) of Attorneys(s)/Firm(s) preferred by inventors to prepare a patent application, if known:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**C. INVENTOR INFORMATION [ALL inventors AND *ONLY true* inventors, i.e., those responsible for conceiving the invention, are to be named. Failure to do so may affect the validity of any patent for this invention and result in a loss of rights.]**

**IN ORDER TO PROCESS AN IDF IN AN EFFICIENT MANNER, ALL OF THE FOLLOWING FIELDS MUST BE FILLED OUT. FAILURE TO FILL OUT THESE BASIC FIELDS WILL RESULT IN THE IDF BEING RETURNED FOR COMPLETION!**

|  |
| --- |
| **If there are more than 3 inventors, copy and paste the inventor section as needed.** |

Inventor #1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LAST NAME  Gu | | | | | FIRST NAME (Full Legal, as on Driver’s License or Passport)  Junli | | COMMON/NICK NAME (if any) | | | MIDDLE INITIAL (if any) |
| AMD BADGE #  472729 | MAILSTOP | | | EMAIL ADDRESS  JunLi.Gu@amd.com | | | | | PHONE  ( 86-10)62820864 | |
| VICE PRESIDENT  Alan Lee | | | DIRECTOR | | | | CITIZENSHIP  China | | | |
| DEPARTMENT #  Research 7887530 | | DIVISION  TE | | | | MANAGER  Yuan Xie | | SITE  Beijing | | |
| HOME ADDRESS  Zhong guan cun, Haidian District, Science Institute South Rd, No. 2, Raycom Infotech Park Tower C, North Building, 19th Floor, Beijing, China, 100190 | | | | | | | | | | |
| If not an AMD employee, then name employer and identify contract with AMD: | | | | | | | | | | |
| INVENTOR SIGNATURE | | | | | | | DATE | | | |

Inventor #2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LAST NAME  Xu | | | | | FIRST NAME (Full Legal, as on Driver’s License or Passport) Yi | | COMMON/NICK NAME (if any) | | | MIDDLE INITIAL (if any) |
| AMD BADGE # | MAILSTOP | | | EMAIL ADDRESS | | | | | PHONE | |
| VICE PRESIDENT | | | DIRECTOR | | | | CITIZENSHIP | | | |
| DEPARTMENT # | | DIVISION | | | | MANAGER | | SITE | | |
| HOME ADDRESS | | | | | | | | | | |
| If not an AMD employee, then name employer and identify contract with AMD: | | | | | | | | | | |
| INVENTOR SIGNATURE | | | | | | | DATE | | | |

Inventor #3

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LAST NAME  Wang | | | | | FIRST NAME (Full Legal, as on Driver’s License or Passport) Weiyan | | COMMON/NICK NAME (if any) | | | MIDDLE INITIAL (if any) |
| AMD BADGE # | MAILSTOP | | | EMAIL ADDRESS | | | | | PHONE | |
| VICE PRESIDENT | | | DIRECTOR | | | | CITIZENSHIP | | | |
| DEPARTMENT # | | DIVISION | | | | MANAGER | | SITE | | |
| HOME ADDRESS | | | | | | | | | | |
| If not an AMD employee, then name employer and identify contract with AMD: | | | | | | | | | | |
| INVENTOR SIGNATURE | | | | | | | DATE | | | |

Inventor #4

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LAST NAME  Huang | | | | | FIRST NAME (Full Legal, as on Driver’s License or Passport) | | COMMON/NICK NAME (if any) | | | MIDDLE INITIAL (if any) |
| AMD BADGE # | MAILSTOP | | | EMAIL ADDRESS | | | | | PHONE | |
| VICE PRESIDENT | | | DIRECTOR | | | | CITIZENSHIP | | | |
| DEPARTMENT # | | DIVISION  TE | | | | MANAGER | | SITE  BEIJING | | |
| HOME ADDRESS | | | | | | | | | | |
| If not an AMD employee, then name employer and identify contract with AMD: | | | | | | | | | | |
| INVENTOR SIGNATURE | | | | | | | DATE | | | |

**D. VALUE DATA**

A patent is valuable only if someone else uses or wants to use it - and we **know** about it! To identify invention value, we need your help. Please answer the following for your invention. To keep value information current, we may check back with you periodically.

|  |  |
| --- | --- |
| 15. **AMD use**: List **all** AMD products in which the invention is planned to be implemented. | |
| Give details: | n/a |
| 16. **Use by others**: What companies/products/processes may find the invention useful/essential? | |
| Give details: | Other semiconductor design/IP companies (not limited to processors/GPUs/APUs). |

|  |  |
| --- | --- |
| 17. **Breadth of use**: Is the invention useful across many product types or processes? | |
| Give details: | Yes: GPUs, APUs, other many-core processors, etc. |

|  |  |
| --- | --- |
| 18. **Difficult to avoid**: Describe how you can achieve the same advantages of the invention while avoiding using the invention you described? | |
| Give details: | Not easy to achieve without invention. |

|  |  |
| --- | --- |
| 19. **Detectability**: How can AMD determine that another company is using the invention? | |
| Give details: | A simple way is to check the program since program needs to be modified to leverage the hardware synchronization unit. If not, memory accesses could be used to check whether the hardware synchronization is used. Design a micro benchmark like this: GPU does little computation (very little computation data) but most synchronization. The microbenchmarks can be run in a loop for a more obvious difference. Check whether off-chip memory accesses are incurred. If more memory accesses are created during the run, then software synchronization is used. Otherwise the proposed hardware synchronization is used. |

|  |  |
| --- | --- |
| 20. **Industry standards**: Is this relevant to technology being promoted or used as an industry standard? If so, please give standard(s) name. | |
| Give details: |  |

|  |  |
| --- | --- |
| 21. **Other value information**: |  |

**I have read, understood and initialed each page of this disclosure and the attachments**:

**Witness 1 signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Printed name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Employee #: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Witness 2 signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Printed name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Employee #: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_