

# Dynamic Graph Multi Processor

## 1

1. Etched GPU TFLOPS
2. + +
3. Etched H100 800 3.3%  
Transformer Sohu FLOPS  
90% GPU 30%
4. Scaling Law Scale Up  
Scaling Law Scale Down

## 2

- AI
- Int8 pattern
- 2D =>
- GPU DRAM
- NoC Cache Fork/Join  
NoC flow control
- NoC/Cache
- consistency corhenrenncy fence+sync
- 1. GMP fork/join
- 2. fence
- 3. fence
- IP cpu  
IP cache  
cache\_hint
- DSA
- SOC
- NOC
- x

- DSA[ ] GPU[ ]

## [ ]

1. [ ] Dot / GEMM => Join+Reduce
2. [ ] [ ] ElementWise Broadcast
3. [ ] [ ] Reduce [ ] G/S
4. [ ]
5. [ ]

1. [ ] + [ ]
2. [ ] imm [ ]
3. [ ]

```
4. /* emulate GPU's LOP3.LUT (three-input logic op with 8-bit truth table) */
uint32_t lop3_fast (uint32_t a, uint32_t b, uint32_t c, uint8_t ttbl)
{
    uint32_t r = 0;
    if (ttbl & 0x01) r |= ~a & ~b & ~c;
    if (ttbl & 0x02) r |= ~a & ~b & c;
    if (ttbl & 0x04) r |= ~a & b & ~c;
    if (ttbl & 0x08) r |= ~a & b & c;
    if (ttbl & 0x10) r |= a & ~b & ~c;
    if (ttbl & 0x20) r |= a & ~b & c;
    if (ttbl & 0x40) r |= a & b & ~c;
    if (ttbl & 0x80) r |= a & b & c;
    return r;
}
```

## [ ] / [ ]

1. [ ]
  1. [ ] graph [ ] DRAM [ ]
  2. [ ] LLC [ ] DRAM [ ] LLC [ ] L3->LLC->DMA->L1->L0->MALU
2. [ ]
  1. [ ] fork/join [ ] L1/L0 [ ] DMA [ ]
  2. [ ]
3. [ ]
  1. Root [ ] & [ ] & [ ]
  2. [ ] NoC [ ] Cache
  3. Fork Join
  4. [ ] CPU [ ] RISCv [ ] IP [ ]
4. [ ]
  1. NPU [ ] NPU Kernel [ ] 1D 2D [ ]
  2. DSA Confige
  3. Atomic Reduce [ ]
  4. LD/ST+Relu [ ]

## [ ]

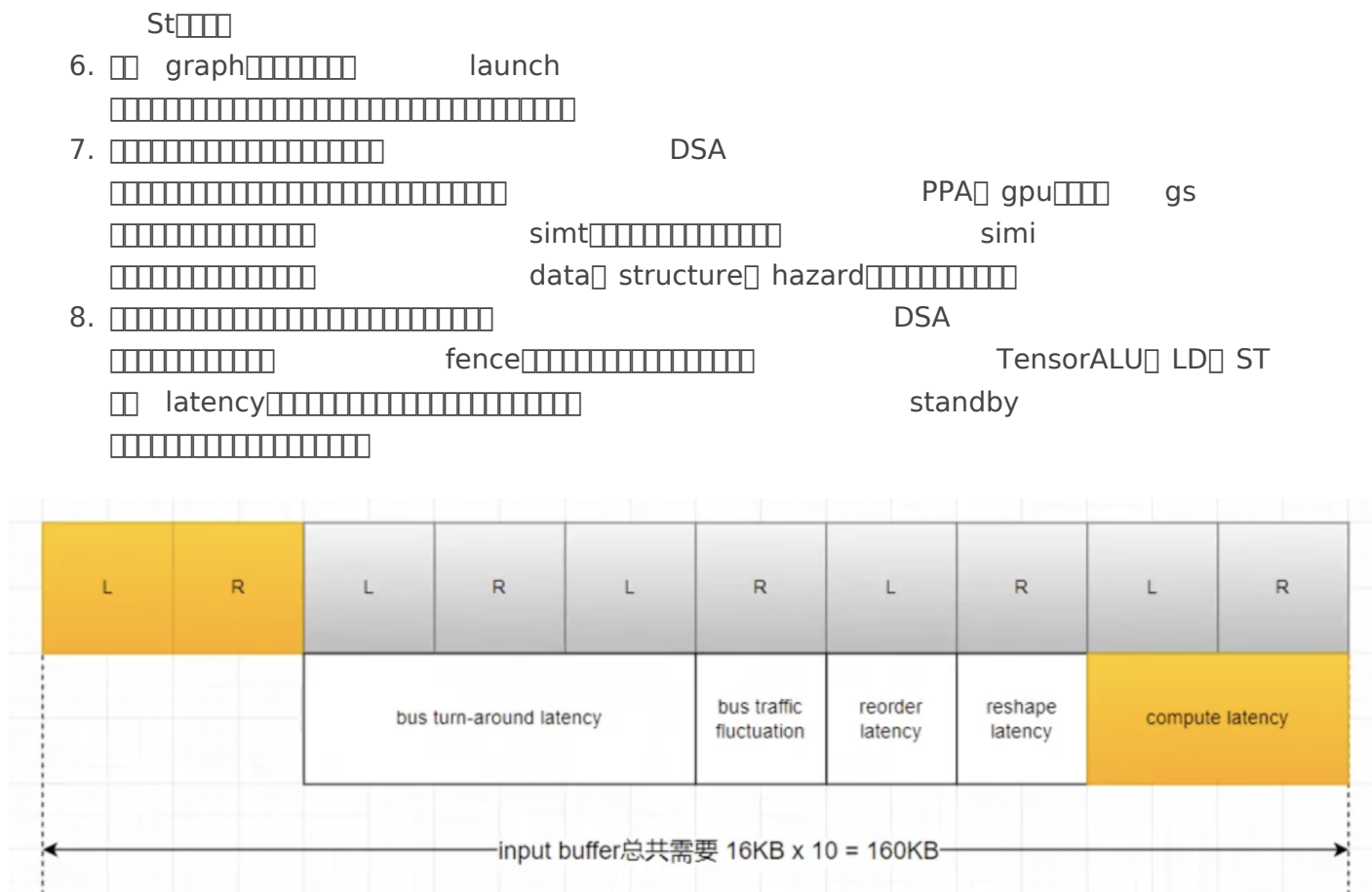
1. NPU LD/ST L2/L1 <-> L2/L1
2. MLI HLI
- 3.
4. MLI
5. Leading/Tainling scheduler pipeline NPU
6. debug / /

## ASIC

1. MLI
2. scheduler NPU
3. latency
4. HLI

## 

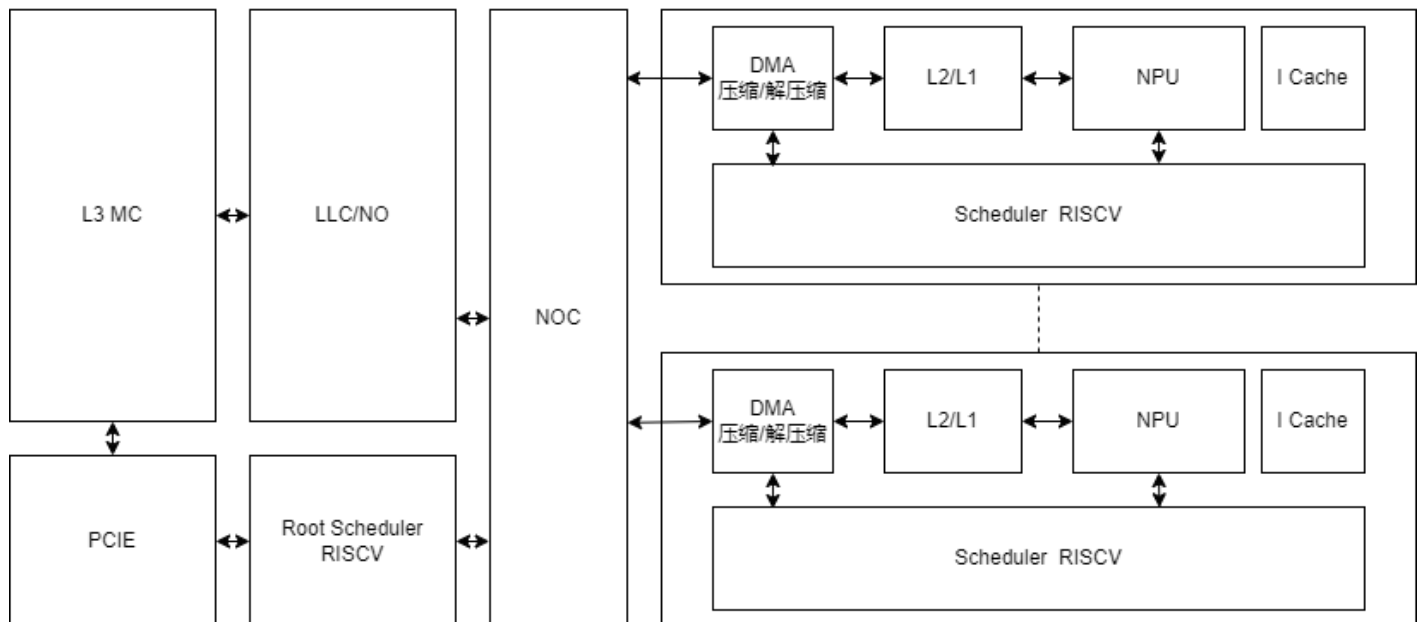
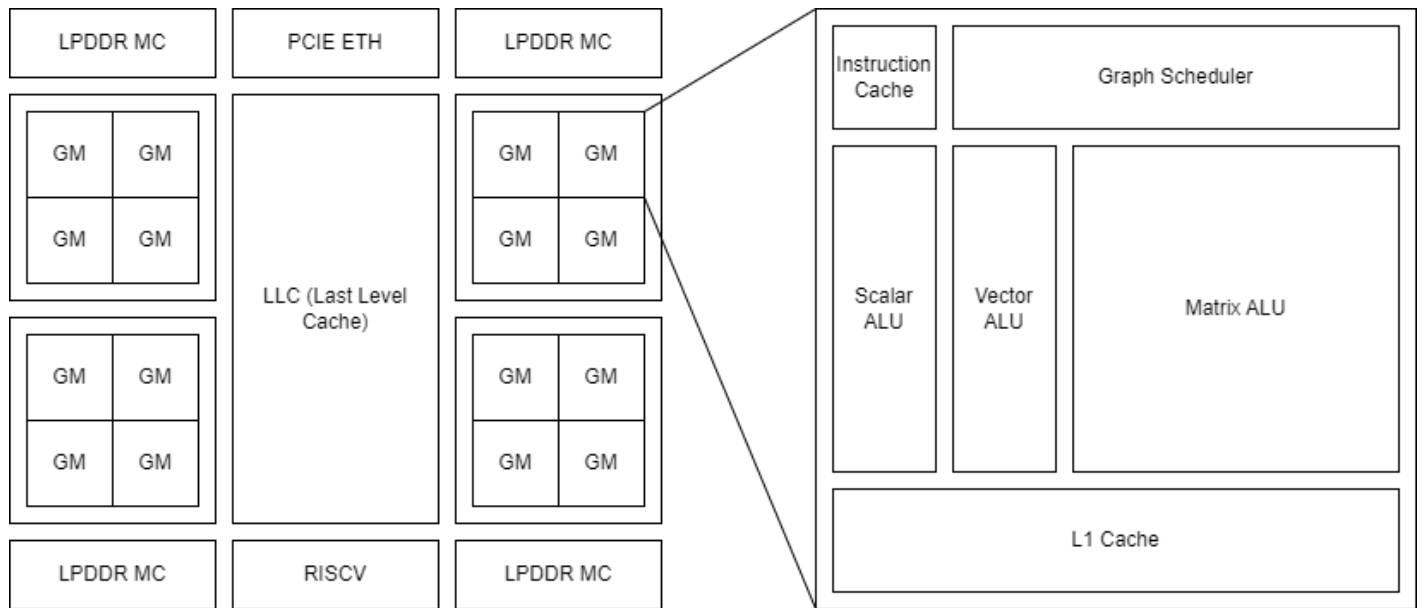
1. GM
  1. scheduler
2.
  1. <8bit DMA 8bit bit
3. Launch/Sync/DataFlow
  1. Launch Sync Fork/Join
    1. **Fork Join** **A fork B C join B**
    2. fork launch join wait
    3. fork join
    4. fork
  2. Launch Sync
  3. NoC Launch
  4. Graph Kernel
  5. **latency**
4. launch
  1. -> -> in flight pipeline
  - 2.
  - 3.
  - 4.
  5. SRAM
    1. SRAM L1 L1 L2 LLC
    2. L1 cache GEMM
    - 3.
    4. L1 bank thread Engine ALU RegRead MALU Id Cal



## 

- 1.
  - pipeline
  - L2L1
- graph
  - DMA /
  - DMA NPU Scheduler Inst SRAM
  - scheduler fork join
  - fork join
- VMM VLD VST pipeline
- kernel
- IP
  - HWSync RISC-V CPU
  - IP mailbox trigger mailbox pipeline
  - forkjoin
- fence leading tailing
  1. NPU leading tailing scheduler




# Graph Multiprocessor = GM












1. `fork(kernel)` `fork(kernel)` `for(kernel)` `launch(kernel)`
2. `kernel` `fork(kernel)` `gemm(kernel)`
2. `kernel`
  1. 32bit `kernel`
  2. `kernel` + `kernel` + 2D `kernel`



1. pull wait wait signal  
2. launch  
3. launch launch  
4. launch sub graph graph group  
"signal" "  
9. delay  
1. data hzd  
10. stall stall














1. Fork Join
- 1. fork
    - 1. 
    - 2.  join 
    - 3.  index
    - 4.  launch  ID  ID  IP  memory mapping
  - 2. fork
    -  load 
  - 3. join
    - 1. launch  ID

Width (bits)	4	6	3	3	1	
Meaning	Reuse flags	Wait barrier mask	Read barrier index	Write barrier index	Yield flag	St cycl

8. `async_group`   
     1. `async copy bulk`   
 9. `mbarrier` fence  
       
 10. credit  valid/ready  
       
     1.   
     2. 

3. 



1.  die
  1. 4T int8
  2.  256MB DDR3 DRAM 1866 16bit  3.7GB/s
  3. 
2. 
  1.  die
  2. 
  3. die to die  serdes

# ISA

## ctrl

1. Fork(code\_index)
2. launch
3. Join()

## Scalar

1. scalar 

## Vector

## Tensor