Superscalar* Club Meeting #2

*we really mean: superscalar speculative out-of-order

James C. Hoe
Department of ECE
Carnegie Mellon University

Working DRAFT
Do NOT Redistribute

Today's Goal:
One step closer to understanding R10K

Working DRAFT
Do NOT Redistribute

References Used

★ Popescu, et al., The Metaflow Architecture, 1991.
- Yeager, MIPS R10K Superscalar Microprocessor, 1996.

- Gonzalez, et al., Processor Microarchitecture: An Implementation Perspective, Synthesis Lectures, 2010.
- Hennessy&Patterson, Computer Architecture: A Quantitative Approach, 5th Edition, 2017.
- Johnson, Superscalar Microprocessor Design, 1990.
- Shen&Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, 2013.

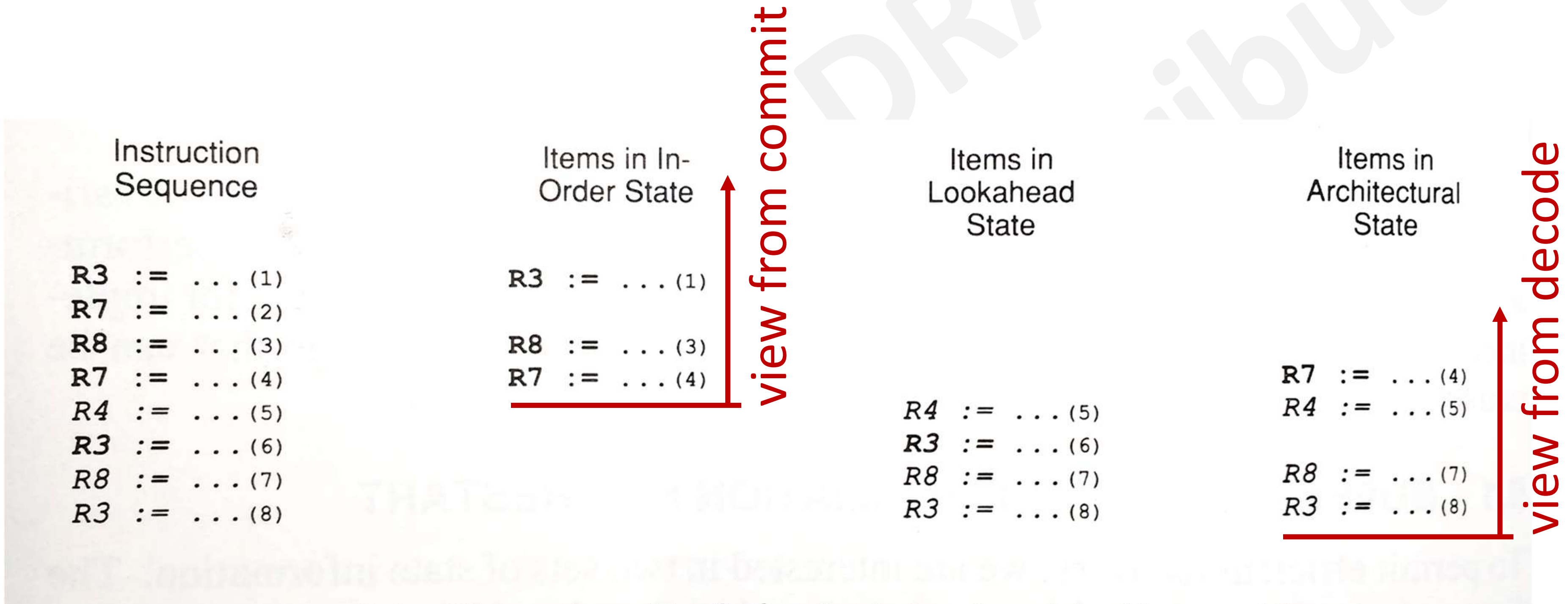
Working DRAFT
Do Not Redistribute

Agree on Terminology btw Us (not a universal agreement)

Working DRAFT
Do NOT Redistribute

Program State Views

DRAFT Do not redistribute



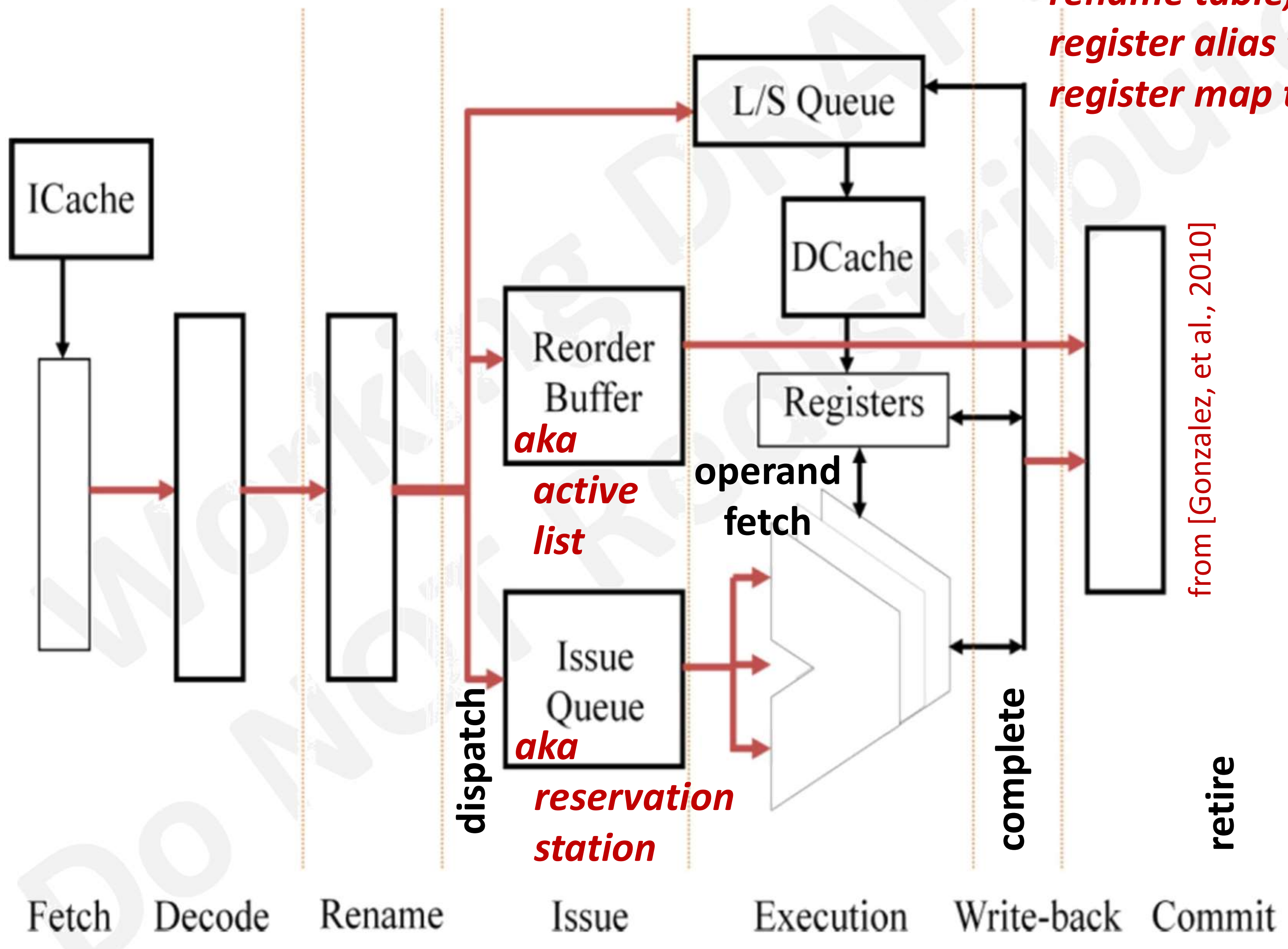
Instruction Sequence	Items in In-Order State	Items in Lookahead State	Items in Architectural State
R3 := ...(1)	R3 := ...(1)		
R7 := ...(2)			
R8 := ...(3)	R8 := ...(3)		
R7 := ...(4)	R7 := ...(4)		R7 := ...(4)
R4 := ...(5)		*R4* := ...(5)	*R4* := ...(5)
R3 := ...(6)		*R3* := ...(6)	
R8 := ...(7)		*R8* := ...(7)	*R8* := ...(7)
R3 := ...(8)		*R3* := ...(8)	*R3* := ...(8)

Figure 5-1. Illustration of In-Order, Lookahead, and Architectural States

from [Johnson, 1990]

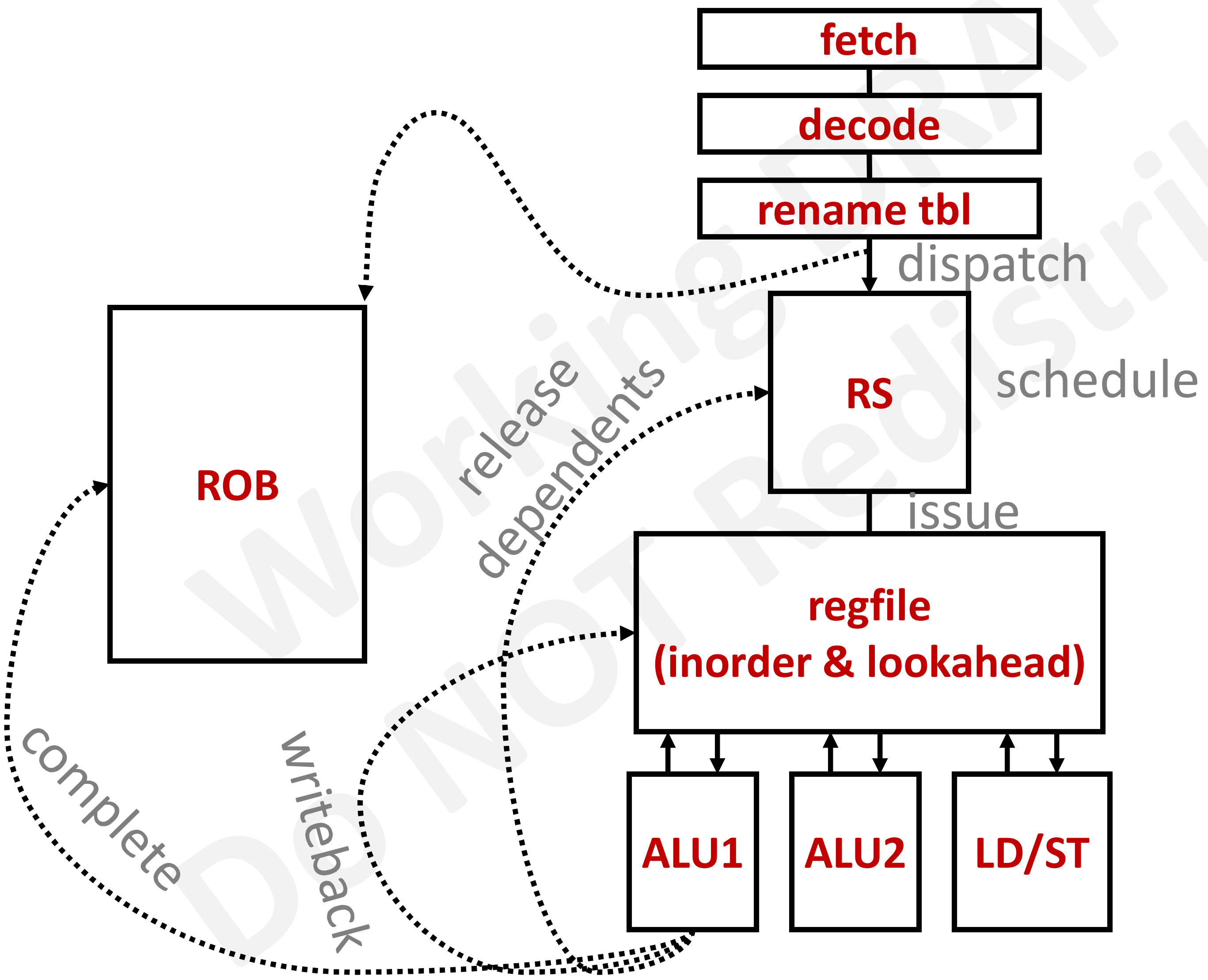
Execution Stages

Not shown
rename table, aka
register alias table, aka
register map table



Working DRAFT Do Not Redistribute

Generic Mental Model



Working DRAFT
Do NOT Redistribute

Metaflow DRIS

Working DRAFT
Do NOT Redistribute

Metaflow Lightning SPARC Processor

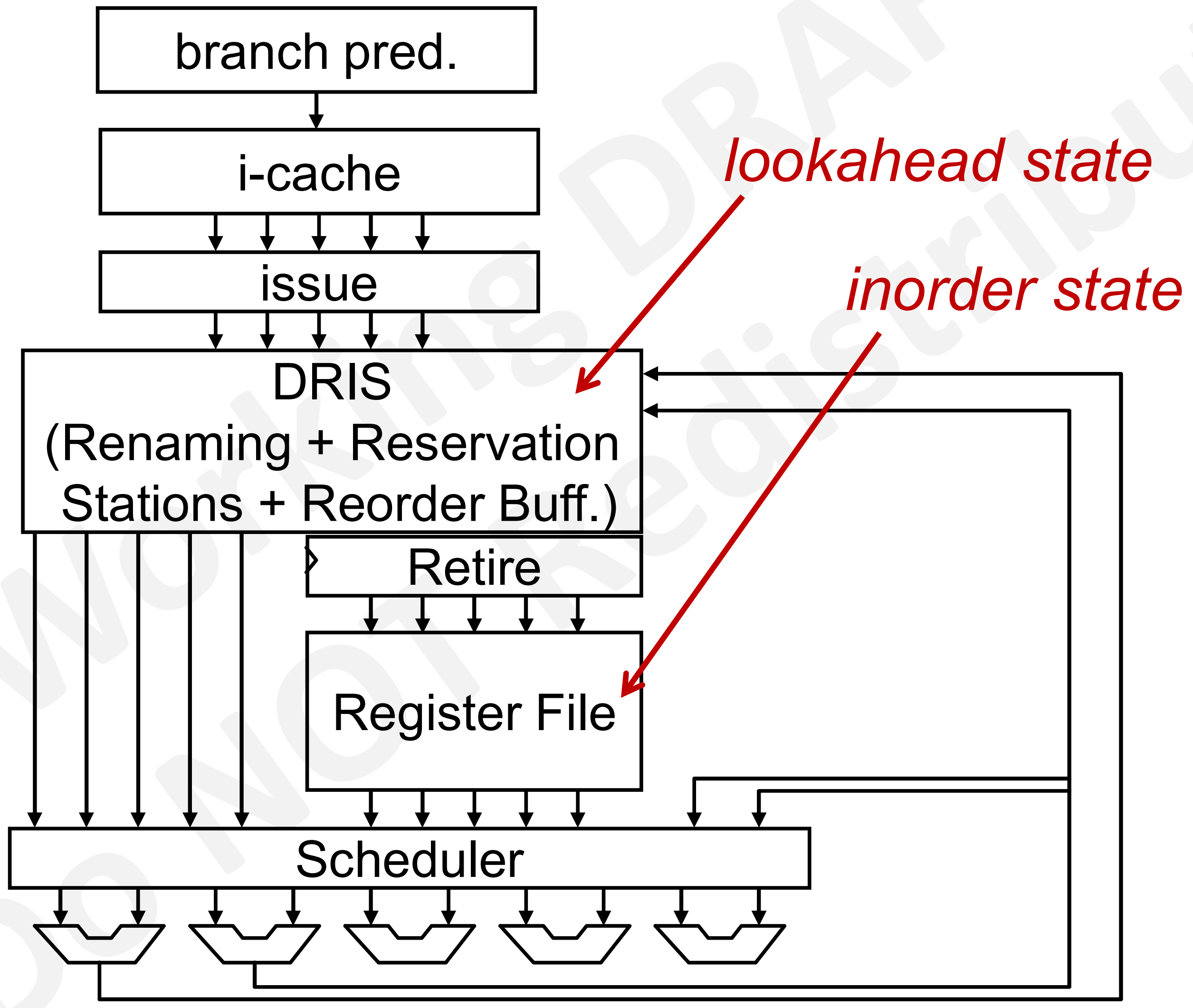
- Superscalar fetch, issue, and execution
- Micro-dataflow instruction scheduling
- Register renaming + memory renaming
- Speculative execution with rapid rewinding
- Precise Exceptions

circa 1991

Claimed "Factor of 2-3 performance advantage from architecture"

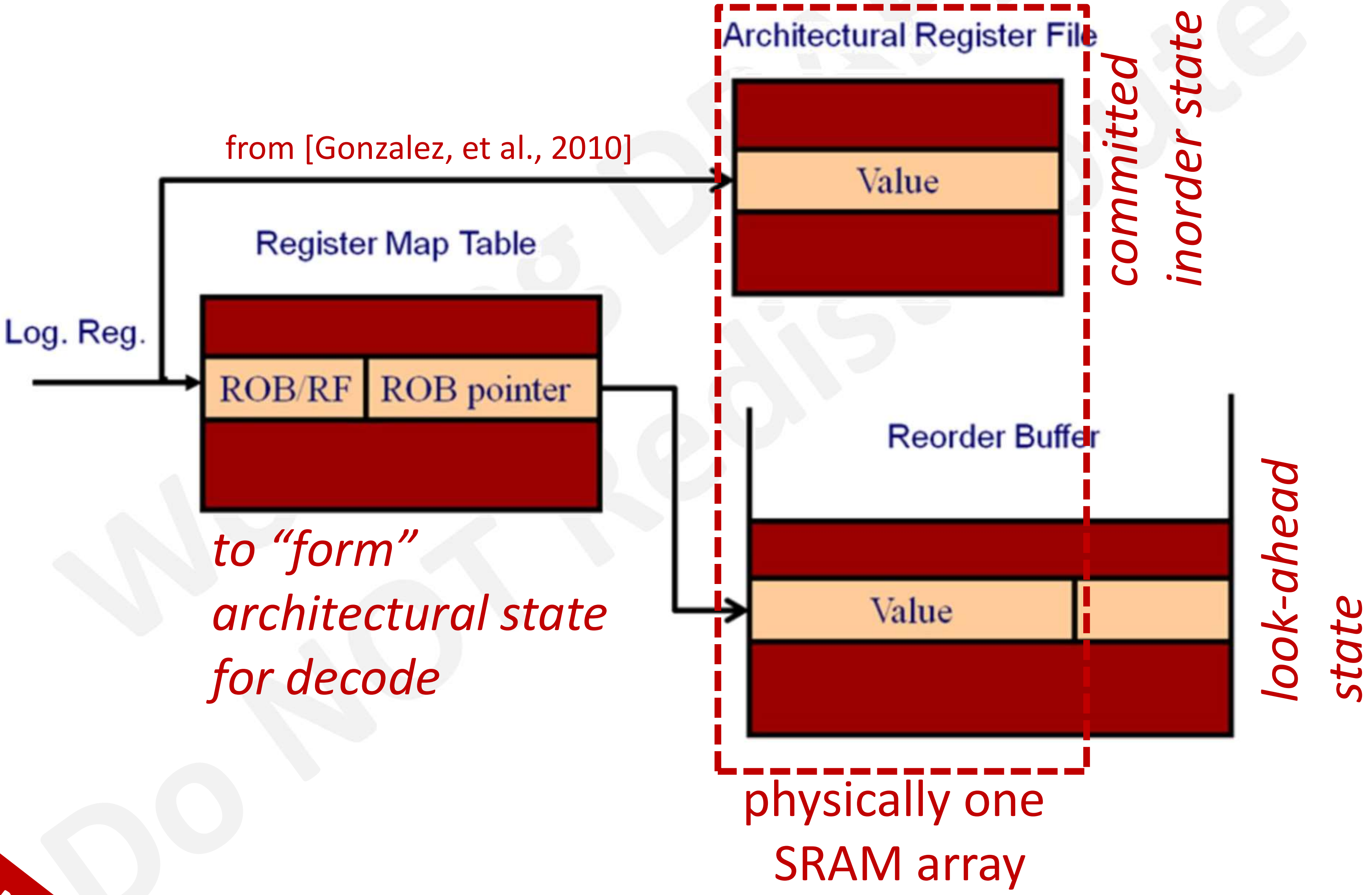
Working DRAFT Do NOT Redistribute

Metaflow Datapath



Working DRAFT
Do NOT Redistribute

ROB Rename Registers



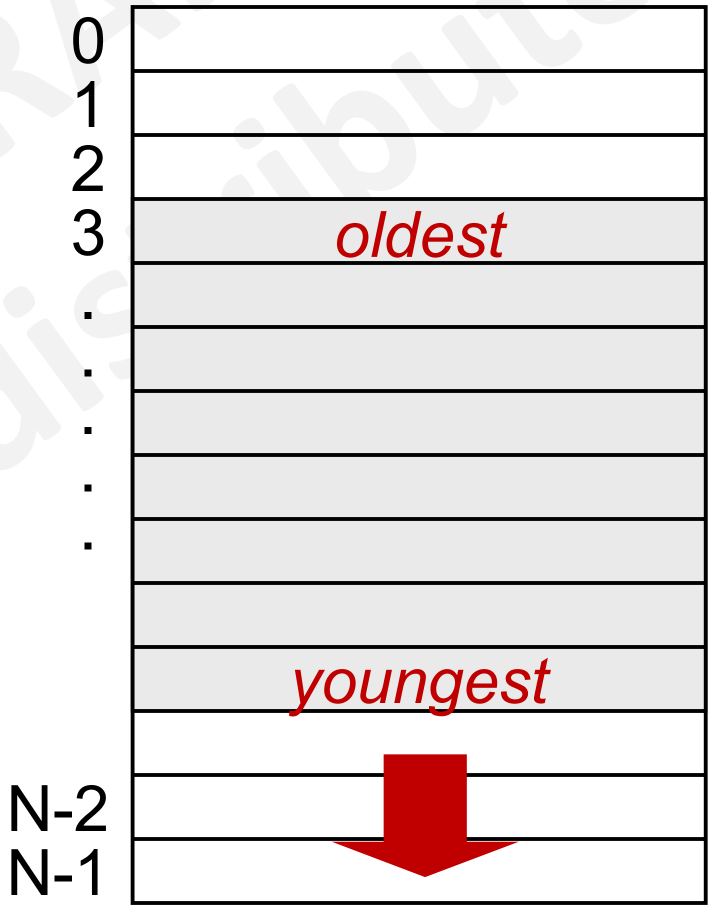
to "form" architectural state for decode

Recall

Working Draft Do NOT Redistribute

DRIS is ROB-like

- Circular Queue Structure
- Instructions held in original program order
  - new entry allocated at tail of queue when instruction issues
  - completed entry committed inorder from head of queue to update regfile or memory
- Instruction's position in ROB is its unique *tag*



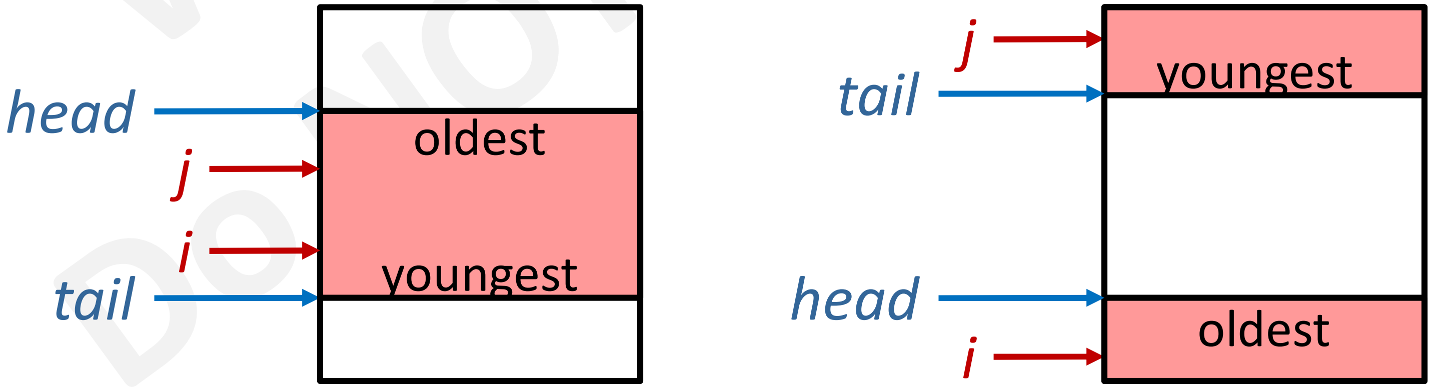
Working DRAFT Do NOT Redistribute

Color Bit

- Head and tail pointers count around N-entry DRIS using $1+\log_2(N)$ bit
  - bottom $\log_2(N)$ ***index*** bits work the way you think
  - top bit ***color*** bit alternate each round through
- Given indices ***i*** and ***j***, ***i*** older than ***j*** when

if (***color***(***i***)==***color***(***j***)) then ***index***(***i***) < ***index***(***j***)

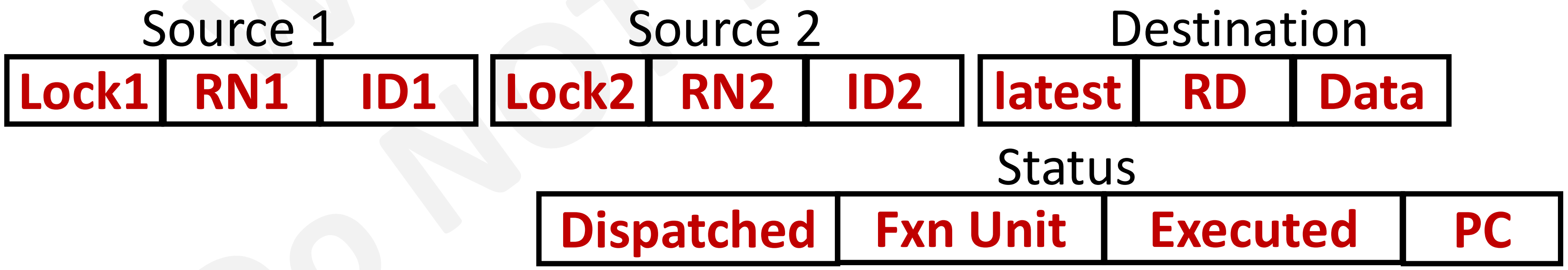
else ***index***(***i***) ≥ ***index***(***j***)



Working DRAFT
Do NOT Redistribute

DRIS is also everything else

- **ROB**: inorder record of in-flight instructions
- **Rename Regfile**: **Data** holds lookahead state of **RD**
- **Rename Table (CAM-based)**: given operand **rs**, search **RD** youngest to oldest for re-definition
- **Reservation Station**: if !(**Lock1** || **Lock2**)
- **Address Queue**: . . .



Know everything, look everywhere philosophy . . .

Working DRAFT
Do NOT Redistribute

“Issue”: (Decode+Dispatch+Rename)

- A new ID (aka tag) is allocated to each instruction when issued into DRIS
  - ID is index to next free DRIS circular queue entry
- Search DRIS (young to old) to see if **rs1/2** matches **RD** of older entry ***i***.
  - if found, set **ID1/2** to *i*; set **Lock1/2** if !**Executed**[*i*]
  - if not found, set **ID1/2** to **(?)** ; set **RN1/2**=**rs1/2**

Source 1			Source 2			Destination		
Lock1	**RN1**	**ID1**	**Lock2**	**RN2**	**ID2**	**latest**	**RD**	**Data**

Working DRAFT Do NOT Redistribute

Rename: add rd, rs, rt

Let's be a little more precise. You give it a try first.

Source 1			Source 2			Destination		
Lock1	RN1	ID1	Lock2	RN2	ID2	latest	RD	Data

Working DRAFT
Do NOT Redistribute

Rename: add rd, rs, rt

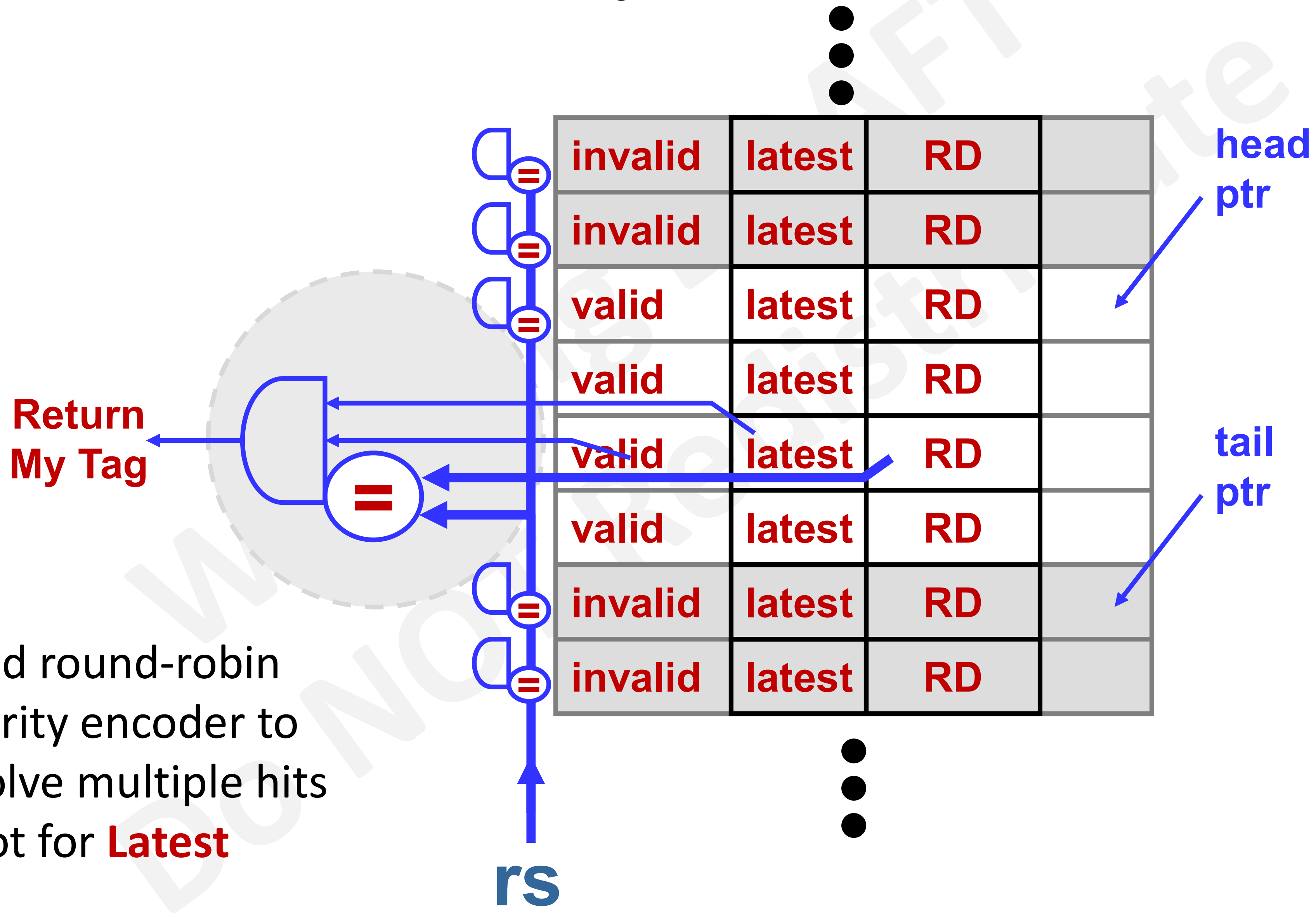
Assume ***new*** is ID of current instruction

```
                RN1[new]= rs ; RN2[new]= rt ;
default   ⎧     Locked1[new]= false; Locked2[new]=false;
  value   ⎩     ID1[new]= "not_valid"; ID2[new]="not_valid";    ??
                forall valid id             // over all active DRIS entries
                        if ((RD[id] == rs)  && Latest[id] )
                                ID1[new] = id ;
                                Locked1[new]=!Executed[id] ;
                forall valid id
                        if ((RD[id] == rt)  && Latest[id] )
                                ID2[new] = id ;
                                Locked2[new]=!Executed[id] ;
```

Source 1			Source 2			Destination		
Lock1	RN1	ID1	Lock2	RN2	ID2	latest	RD	Data

Working DRAFT
Do NOT Redistribute

Associative Lookup for Just 1 Rename



Need round-robin priority encoder to resolve multiple hits if not for **Latest**

Superscalar Rename

- Replicate the previous circuit, 2 per instruction
- All new entries read and update DRIS together
  - replicated circuits all rename relative to same "architectural" view (i.e., from same tail ptr)
  - what happens for 3-wide rename:

add r1, r2, r3		add r1, r1, r3
add r4, r5, r6	vs	add r1, r1, r4
add r7, r8, r9		add r1, r1, r5

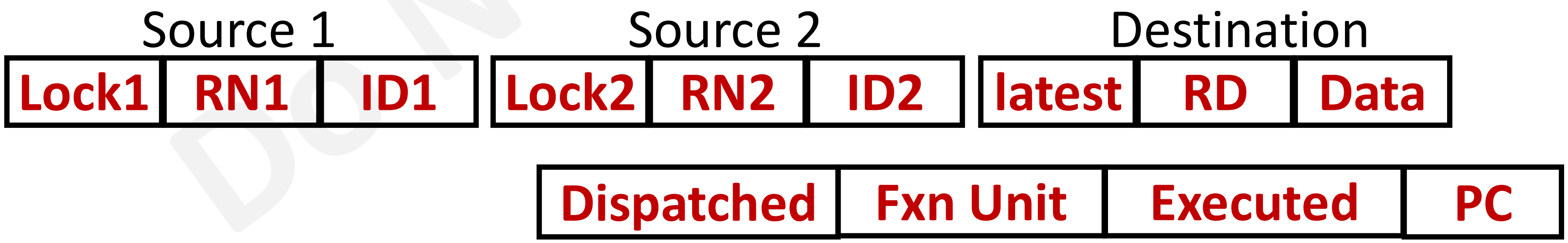
- Must do on-the-fly RAW dependence check and re-direct, O(N^2) complexity

Working DRAFT
Do NOT Redistribute

Rest is Easy: add rd, rs, rt

```
RD[new] = rd ;
forall valid id
        if (RD[id] == rd)
                Latest[id]=false ;
Latest[new]=true ;
Dispatched[new]= false ;
Executed[new]= false ;
FxnU[new]=Integer ALU ;
```

one more CAM



Working DRAFT Do NOT Redistribute

Micro-Dataflow Scheduling

- The scheduler dispatches according to
  - availability of pending instructions' operands
  - availability of the functional units
  - chronological order of the instructions

Is oldest-first the "best" strategy? What is?

- Find instructions such that

valid[*id*] && !**Locked1**[*id*] && !**Locked2**[*id*] &&
!**Dispatched**[*id*] && !**Executed**[*id*] &&
notBusy(**fxnUnit**[*id*])

Think about the circuits & multiply for superscalar

yet one more CAM

Working DRAFT Do NOT Redistribute

Issue: add rd, rs, rt

- A issued instruction is sent to the functional unit with its operands and its *id*
- Operands come from:
  - DRIS: **Data**[**ID1/2**[*id*]] when **ID1/2**[*id*] is younger than head ptr

    in lookahead state
  - Regfile: Regfile[**RN1/2**[*id*]] when **ID1/2**[*id*] is older than head ptr (producer was already retired at decode or has since retired)

    in inorder state

Source 1			Source 2			Destination		
Lock1	**RN1**	**ID1**	**Lock2**	**RN2**	**ID2**	**latest**	**RD**	**Data**

Working DRAFT
Do Not Redistribute

Writeback/Complete

- A Fxn unit returns both the ***result*** and the associated ***id***

    Data[*id*]=***result*** ;

    Executed[*id*]=*true* ;

- Unlock RAW dependent instructions
    - forall *valid* ***i***

        if (**ID1**[*i*]== *id*) **Locked1**[*i*]=*false*;
    - forall *valid* ***i***

        if (**ID2**[*i*]== *id*) **Locked2**[*i*]= *false*;

yes, more CAMs

Source 1			Source 2			Destination		
Lock1	**RN1**	**ID1**	**Lock2**	**RN2**	**ID2**	**latest**	**RD**	**Data**

Working DRAFT Do NOT Redistribute

Retire

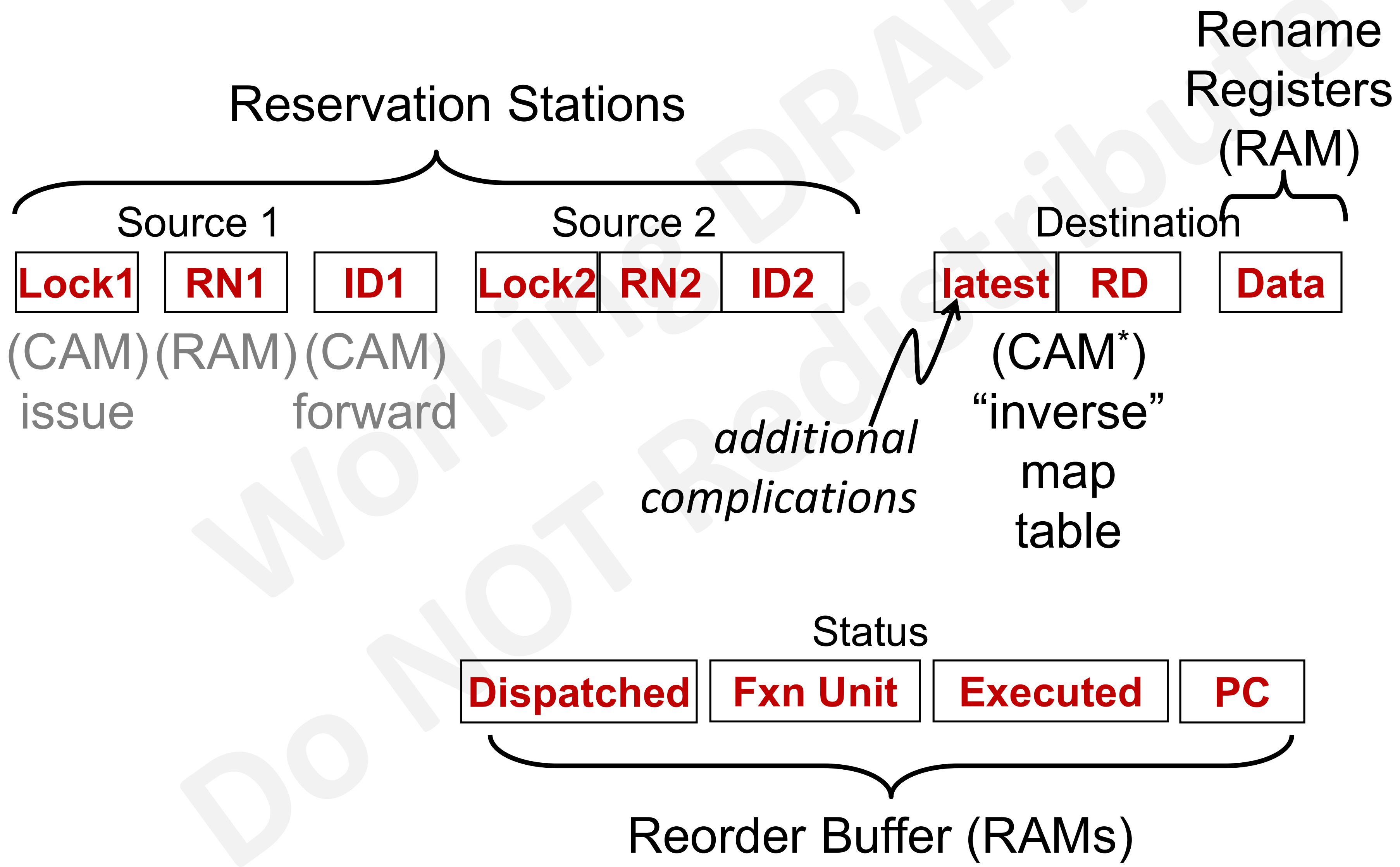
- Instructions retire from DRIS inorder from oldest
  - must be **Dispatched**, wait if not **Executed**
  - not on wrongpath
  - no older exceptions, itself could be
- Commit lookahead state to inorder state

  Regfile[**RD**[*retiree*]]=**Data**[*retiree*]
- Similarly, SW can only write memory when retiring

Source 1			Source 2			Destination		
Lock1	RN1	ID1	Lock2	RN2	ID2	latest	RD	Data

Working DRAFT Do NOT Redistribute

Logical vs. Physical



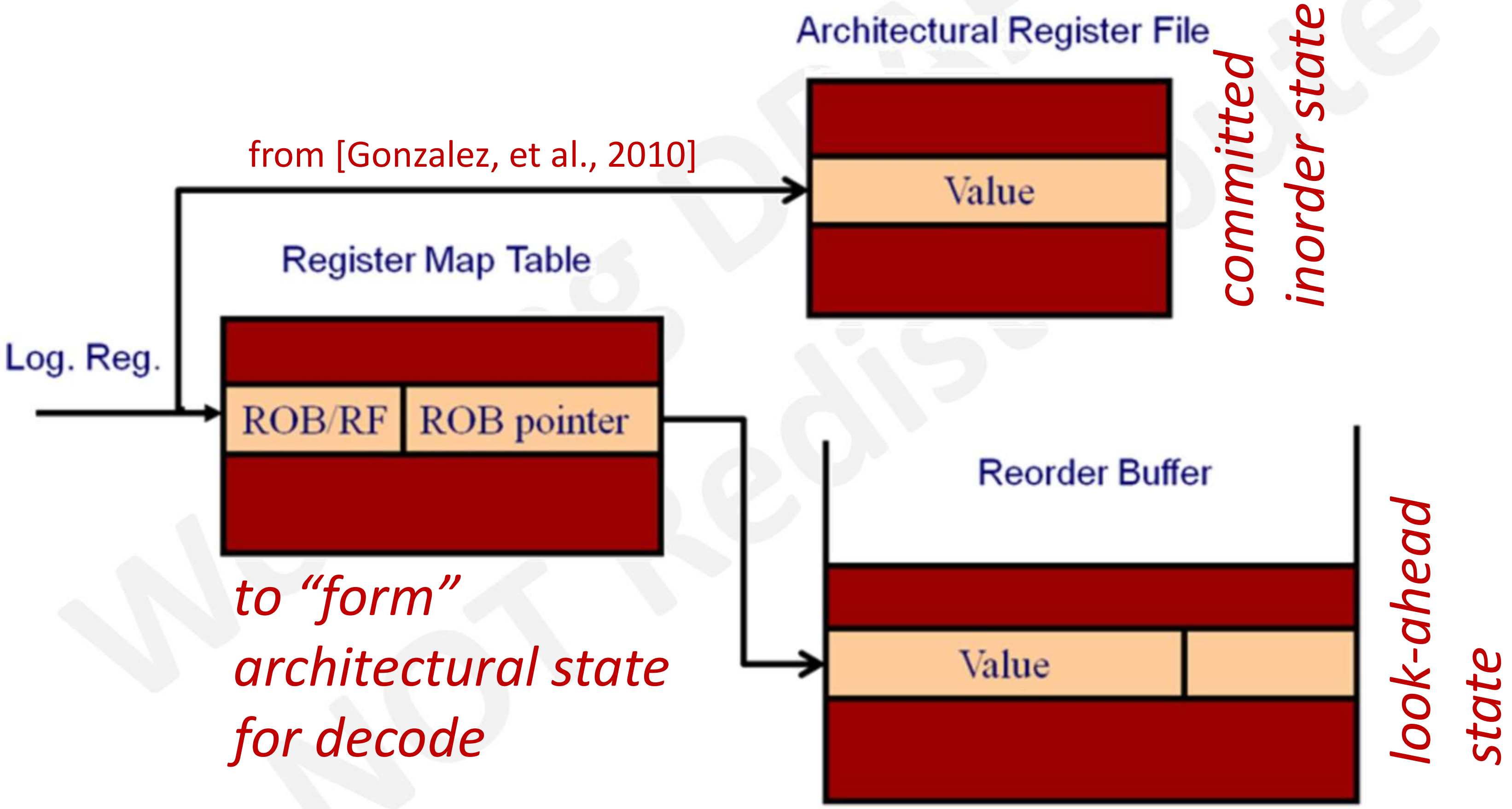
Working DRAFT
Do NOT Redistribute

The Cost of Implementing DRIS

- To support *N-way* rename per cycle
  - *N*x3 associative lookup and read
  - *N*x2 indexed read, *N*x2 indexed write
- To support *N-way* issue per cycle
  - 1 prioritized associative lookup of *N* entries
  - *N* indexed write
  - *N*x2 indexed read in DRIS
  - *N*x2 indexed read in Regfile
- To support *N-way* complete *per cycle*
  - *N* indexed write to DRIS
  - *N*x2 associative lookup and write in DRIS
- To support *N-way* commit per cycle
  - *N* indexed lookup in DRIS
  - *N* indexed write to DRIS
  - *N* indexed write to Regfile

Working DRAFT Do NOT Redistribute

Why CAM and not MAP Table?



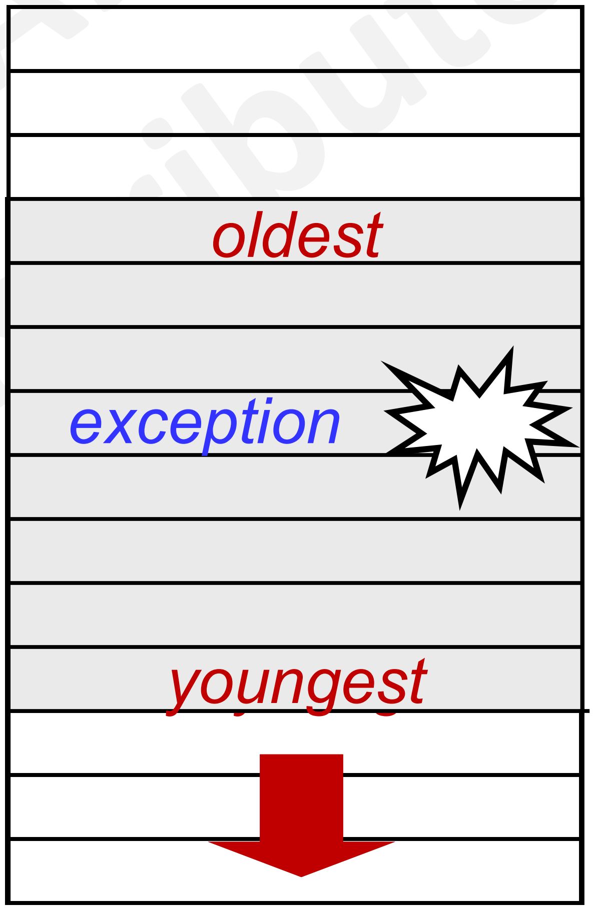
What happens on exception and branch rewind?

Work DRAFT Do NOT Redistribute

Precise Exceptions

- On exception, stop fetching
- Wait until exception oldest in DRIS, set tail ptr to head ptr
- Done!

Does this work for branch rewind?

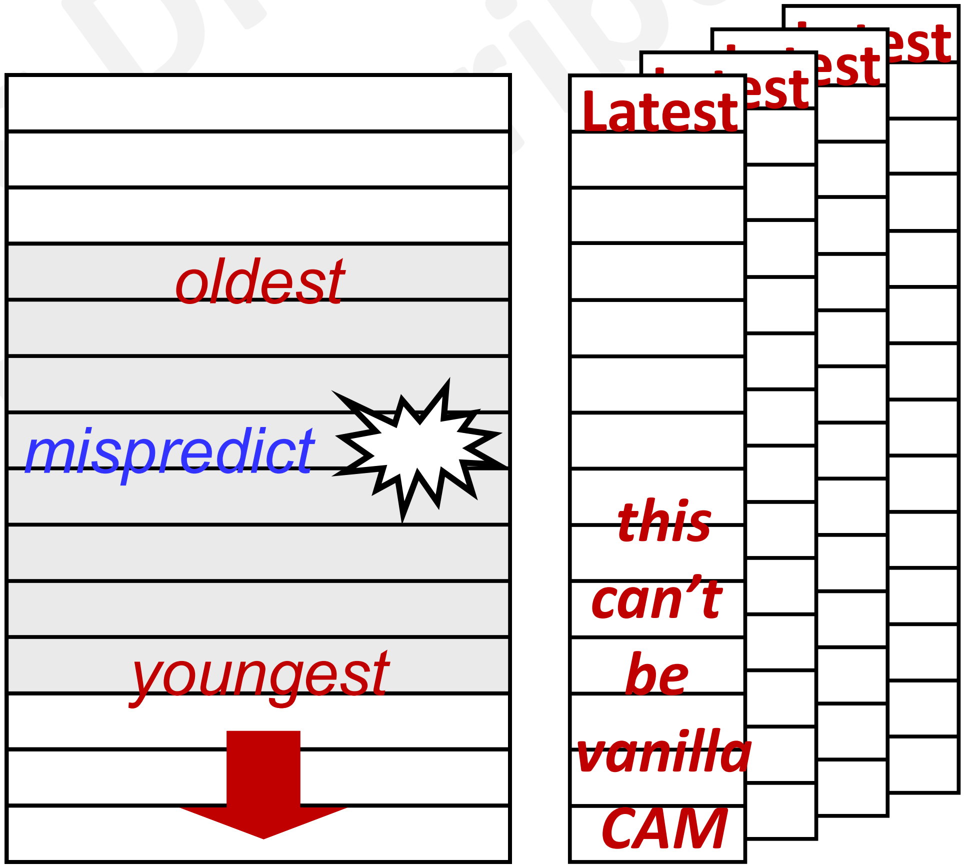


Working DRAFT
Do NOT Redistribute

Rewind cannot wait for the head/oldest

- Set tail ptr to after mispredicted branch to restart . . .
- What about **Latest**?

Branch Stack
(more next wk)



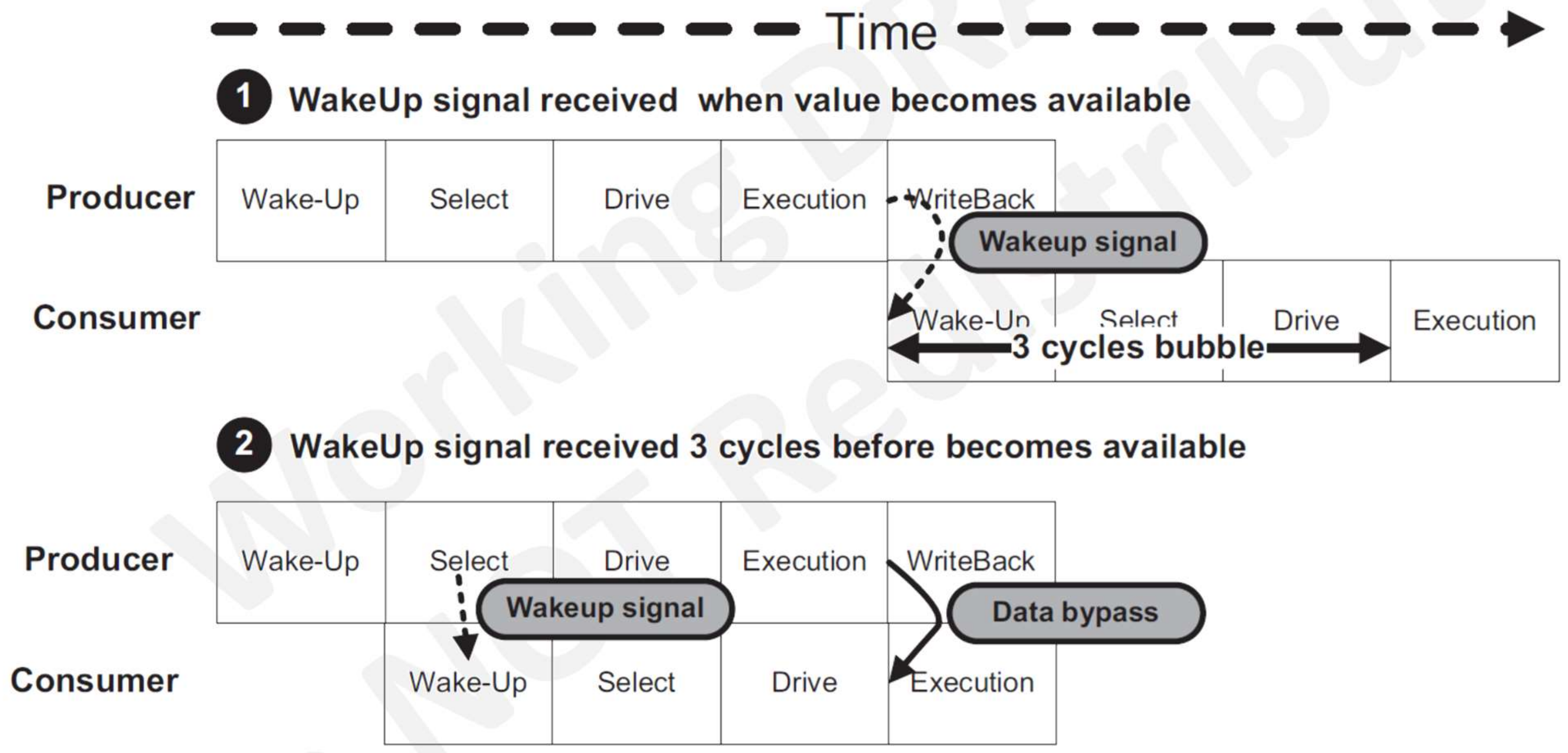
Working DRAFT
Do NOT distribute

To Sum Up

- Superscalar, speculative, out-of-order made "easy" by centralizing bookkeeping (if you could know everything at the same time)
- Circuits too elaborate (gratuitous use of large CAM especially) and inefficient (size and critical path) for what need to be accomplished
- On branch rewind
  - identifying younger instructions occupying out-of-order structures by index-comparison too costly
  - how to recovery the "latest" column?
- What is IPC when ILP=1? (paper doesn't say but it is handled the right way)

Working DRAFT Do NOT Redistribute

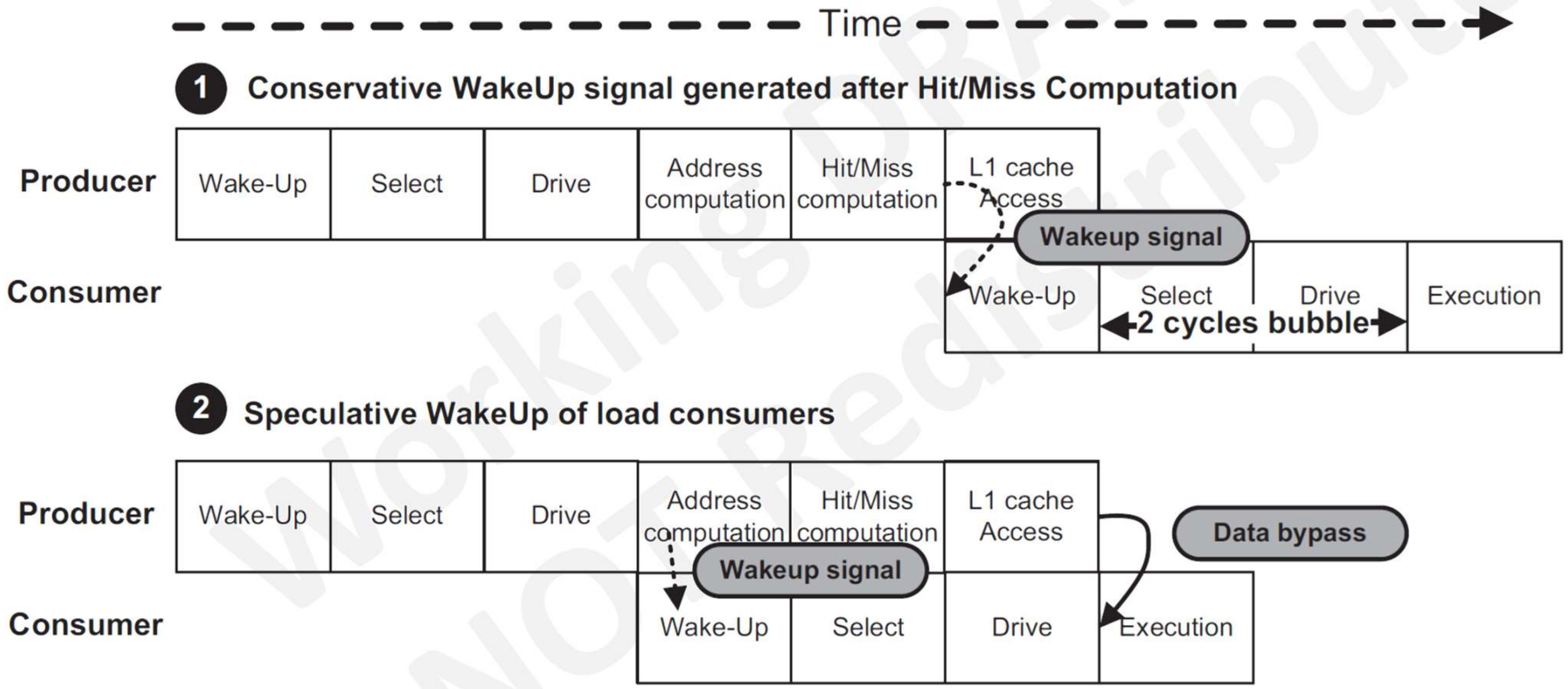
Issue and Forwarding Timing



from [Gonzalez, et al., 2010]

Working DRAFT Do NOT Redistribute

Scheduling Consumer of Loads

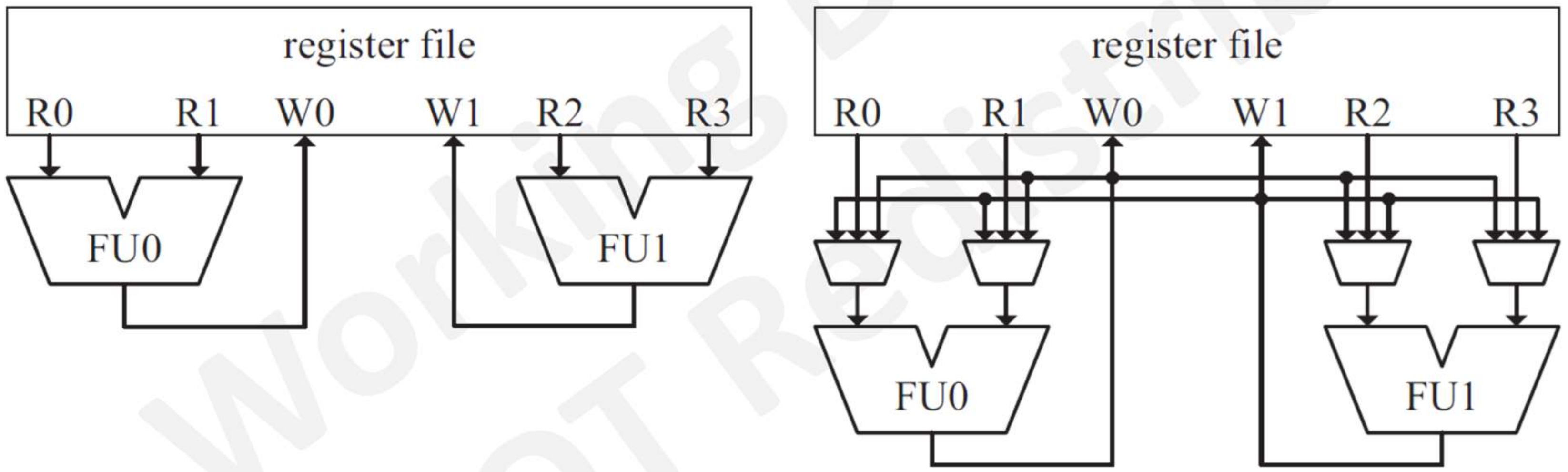


What if LW misses?

from [Gonzalez, et al., 2010]

Working DRAFT
Do NOT Redistribute

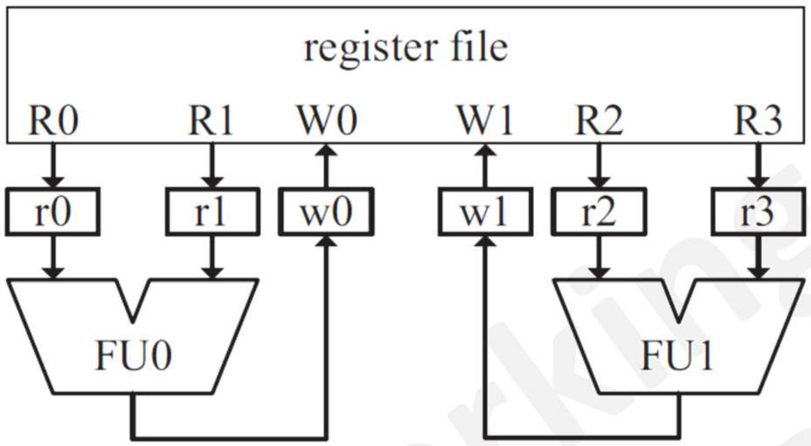
Forwarding Paths Required

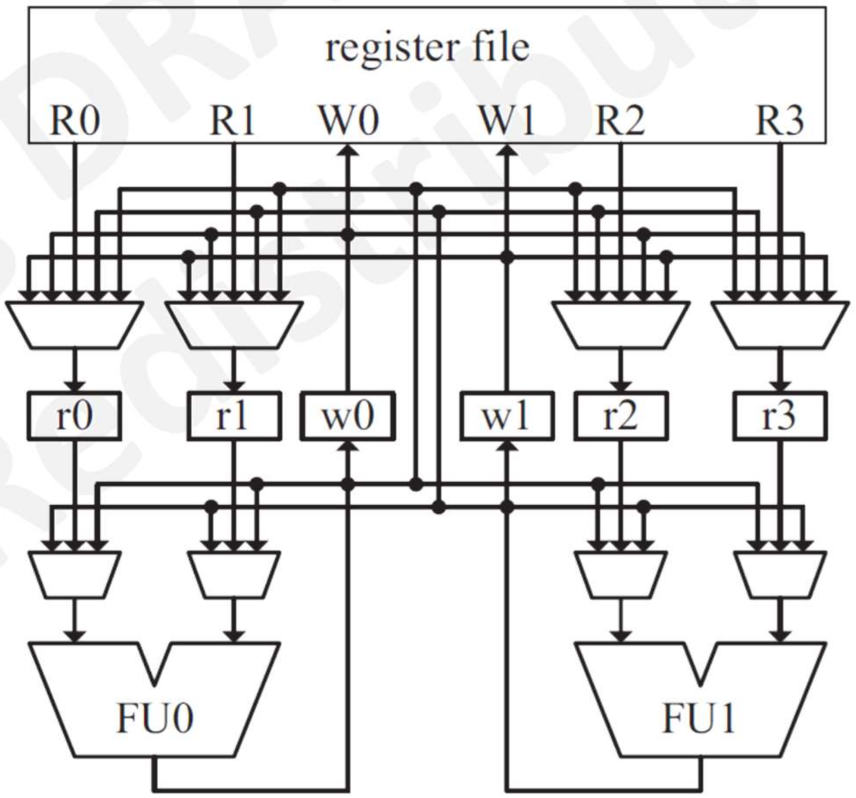


from [Gonzalez, et al., 2010]

Working DRAFT
Do NOT Redistribute

Forwarding Path Complexity





from [Gonzalez, et al., 2010]

Working DRAFT
Do NOT Redistribute

Read R10K very, very carefully for next week

Working DRAFT
Do NOT Redistribute