A Real-Time and Bandwidth Guaranteed Arbitration Algorithm for SoC Communication

Chien-Hua Chen, Juinn-Dar Huang, Geeng-Wei Lee, Jing-Yang Jou

Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan

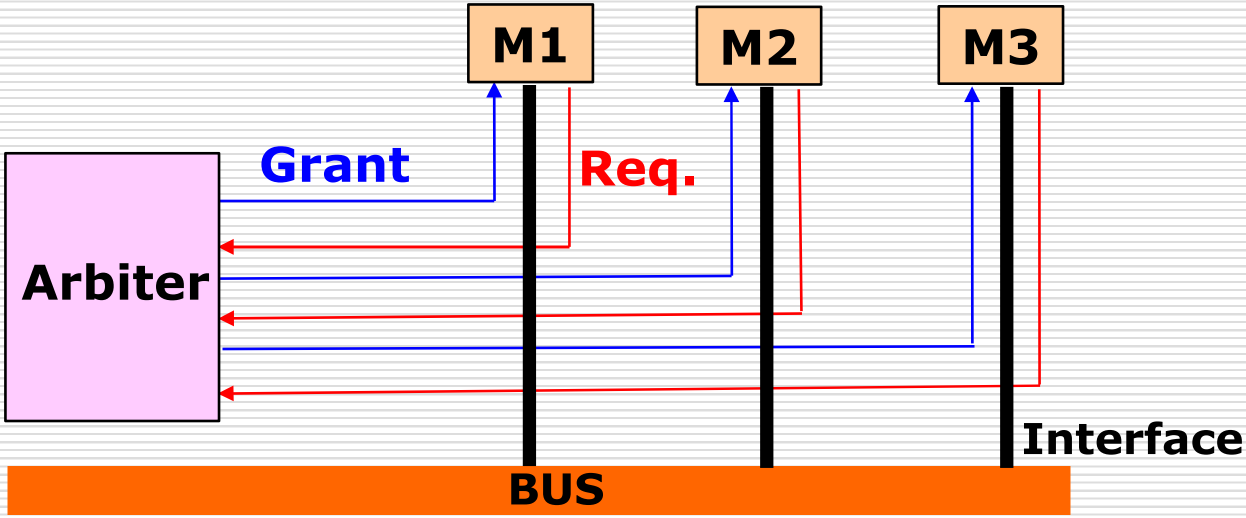
2006/01/26

Outline

- Previous Work
- Our Approach
- Experimental Results and Conclusions

Objective of an Arbiter

- An arbiter resolves contention problems
  - decide which master can access the bus



Performance Evaluation of an Arbiter

- Low latency
- Real-time handling
  - some masters require tasks accomplished within fixed cycles
- Guaranteed fraction of communication bandwidth
  - QoS concept
- Efficient channel utilization
- Low hardware complexity

Various Types of Arbitration Schemes

- Static Fixed Priority
- TDM (Timed Division Multiplexed)
- LOTTERY
- etc.

Static Fixed Priority

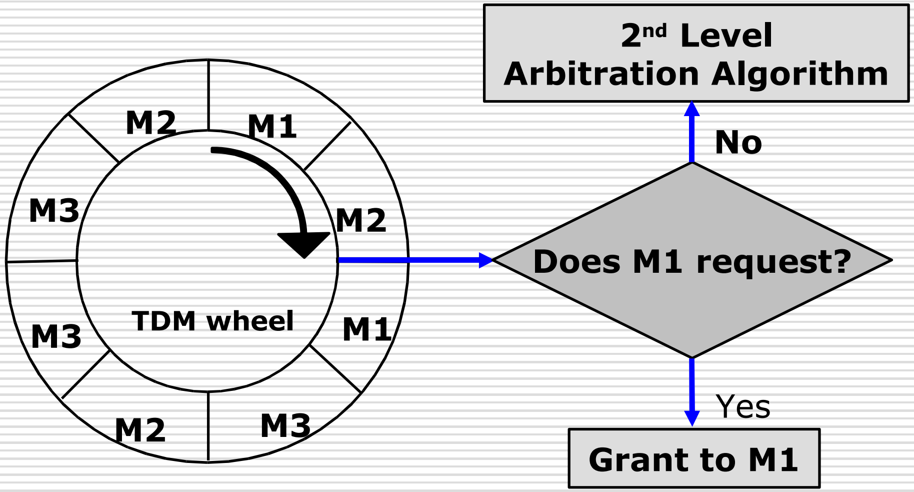
- Each master is assigned a unique priority value
- The master with the highest priority always gets granted

- Pros
  - simple implementation
  - low hardware complexity
- Cons
  - starvation of low priority masters
  - unfair bandwidth allocation

TDM

- Time Division Multiplexed (TDM)
  - divide access time on the channel into time slots
  - allocate time slots to masters

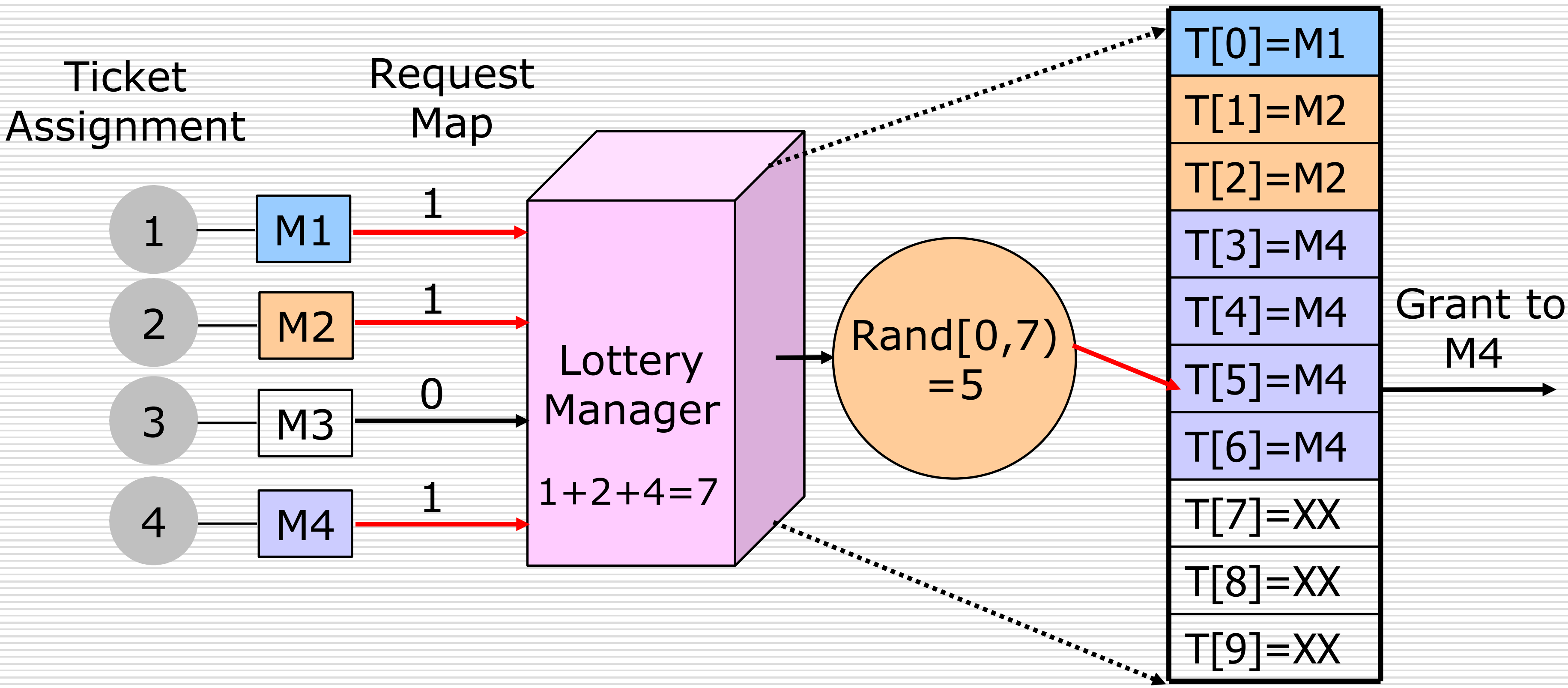
- A 2nd level arbitration algorithm is usually adopted for efficiency
  - 1st level
    - TDM wheel
  - 2nd level
    - any algorithm (application dependent)



Lottery* (1/3)

- Each master is allocated its own ‘lottery tickets’
- The master is chosen probabilistically
  - according to 'lottery tickets'
- The arbiter generates a pseudo random number
  - matching one ticket number
- The master having more tickets is more likely to be granted

Lottery (2/3)


Ticket Assignment
Request Map
1
2
3
4
M1
M2
M3
M4
1
1
0
1
Lottery Manager
1+2+4=7
Rand[0,7) =5
T[0]=M1
T[1]=M2
T[2]=M2
T[3]=M4
T[4]=M4
T[5]=M4
T[6]=M4
T[7]=XX
T[8]=XX
T[9]=XX
Grant to M4

Lottery (3/3)

- Ticket assignment
  - the number of tickets assigned is similar to the weight of each master in other arbitration algorithms
  - masters with larger number of tickets will have:
    - lower response latency
    - higher allocated bandwidth

Summary of Previous Works

- Static Fixed Priority and TDM can not handle real-time and bandwidth requirements at the same time
- Lottery
  - the resultant bandwidth ratio does not conform to the weight ratio
    - finer weight tuning is required (ticket re-assignment)
  - failed in hard real-time applications
    - extra care for real-time requirements is required (real time handler)

Motivations

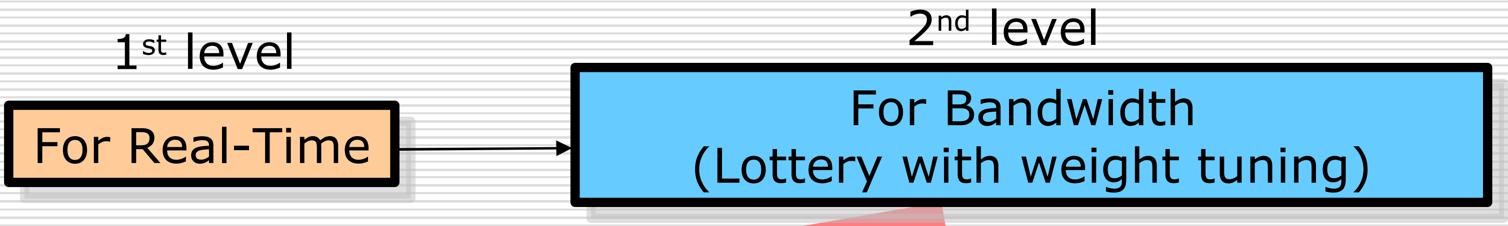
- Develop an arbitration algorithm to meet hard real-time and bandwidth requirements at the same time

Outline

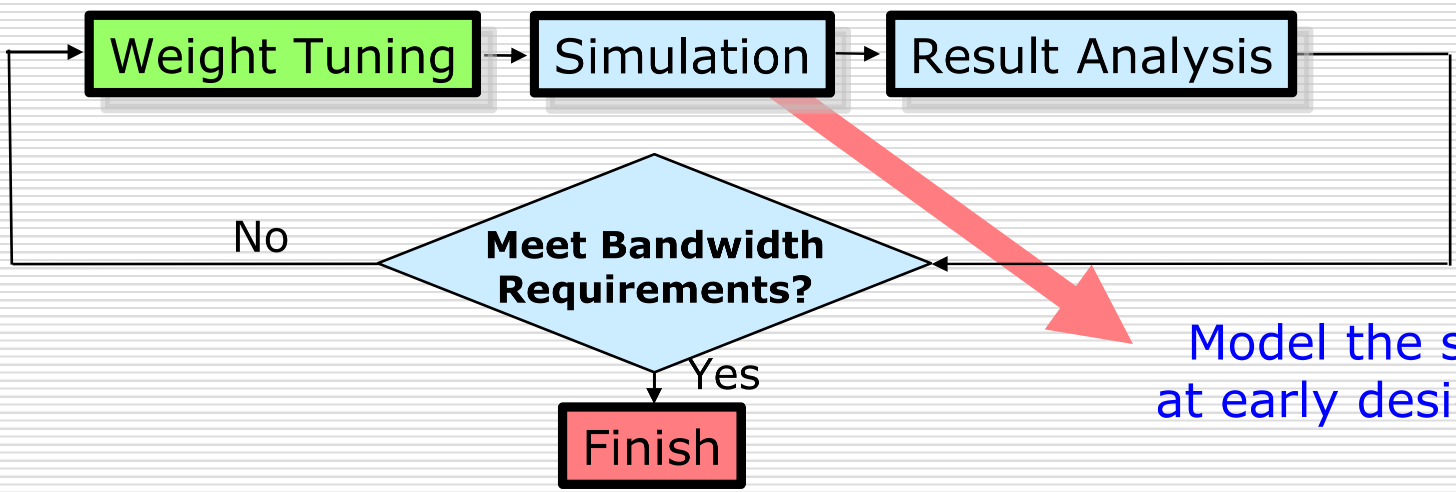
- Previous Work
- Our Approach
- Experimental Results and Conclusions

Proposed Arbiter Architecture

- To meet both bandwidth and real-time requirements



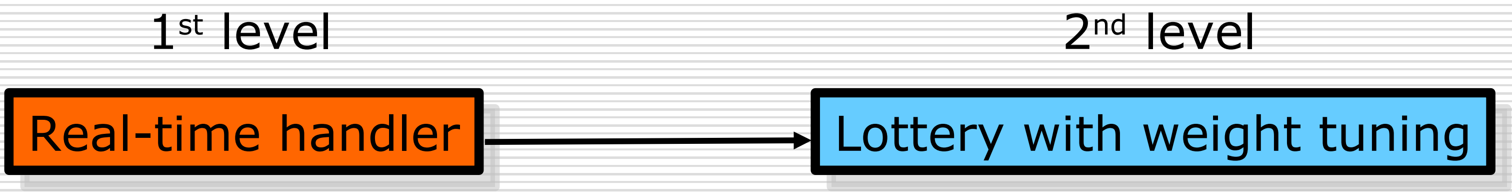
- Weight Tuning (simplified version)



Model the system
at early design stage

Proposed Arbitration Algorithm

- 2-level Arbiter
  - 1st level intends to handle real-time requirements (Real-time handler)
  - 2nd level intends to reserve bandwidth for masters (Lottery with weight tuning)
- The proposed algorithm is named RT_lottery (R for Real-time, T for weight Tuning)



Model - User Input

- Master parameters
  - type
  - R_{cycles}
    - the real-time requirement (in cycles) of a master
  - required bandwidth
  - beat numbers and their probabilities
  - interval cycles and their probabilities

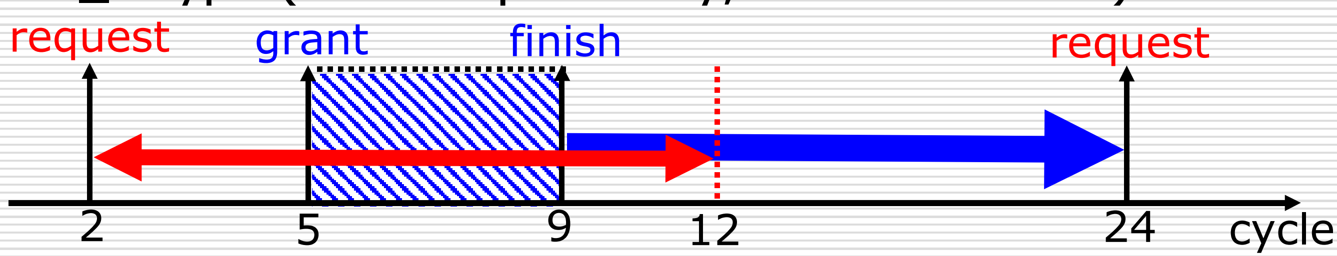
	type	R_{cycles}	req. BW	beat/prob.			interval/prob.	
M1	D		10	4/50	5/20	6/30	60/20	70/80
M2	D_R	100	30	3/20	4/50	5/30	80/10	90/90
M3	ND_R	120	20	5/30	6/50	7/20	14/50	16/50

3 Types of Masters

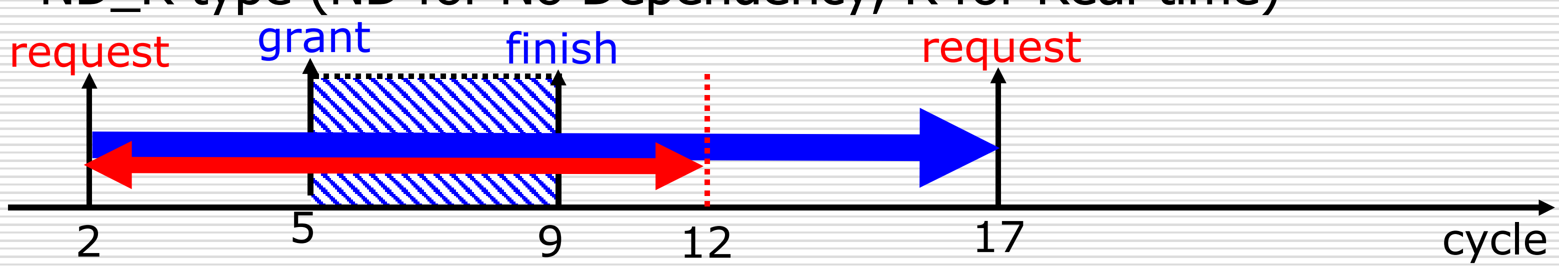
- D type (D for Dependency)

Example : beat = 4, interval = 15, R_{cycles} = 10

- D_R type (D for Dependency, R for Real-time)



- ND_R type (ND for No Dependency, R for Real-time)



Algorithm of Real-Time Handler (1/5)

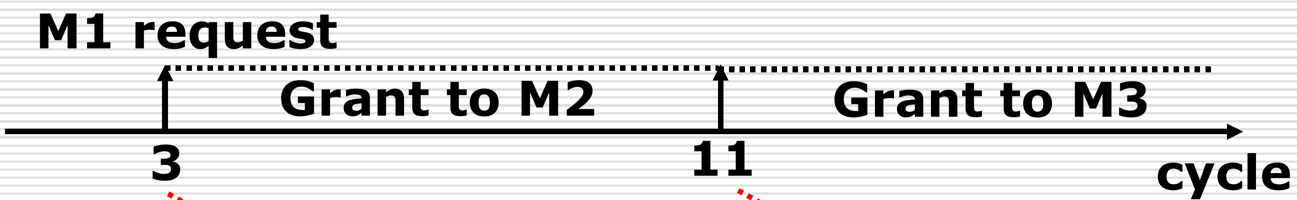
- Real-Time Handler sets real-time counters for those masters with real-time requirements
- When a master asserts request high, the real-time counter for this master is set to its R_{cycles}
- Each real-time counter for the requesting masters is decremented by 1 every cycle until the request is granted
- Warning line mechanism is used to grant emergent masters

Algorithm of Real-Time Handler (2/5)

- A master would have higher priority if its corresponding real-time counter is below the warning line
- When two or more real-time counters are below the warning line, the master with the smallest counter value gets granted

Algorithm of Real-Time Handler (3/5)

- Example
  - warning_line = 25
  - R_{cycles} of M1 = 30



	Real-time counter	Request	Grant
M1	30	Yes	No
M2	25	Yes	Yes
M3	27	Yes	No

	Real-time counter	Request	Grant
M1	22	Yes	No
M2	100	No	No
M3	19	Yes	Yes

Algorithm of Real-Time Handler (4/5)

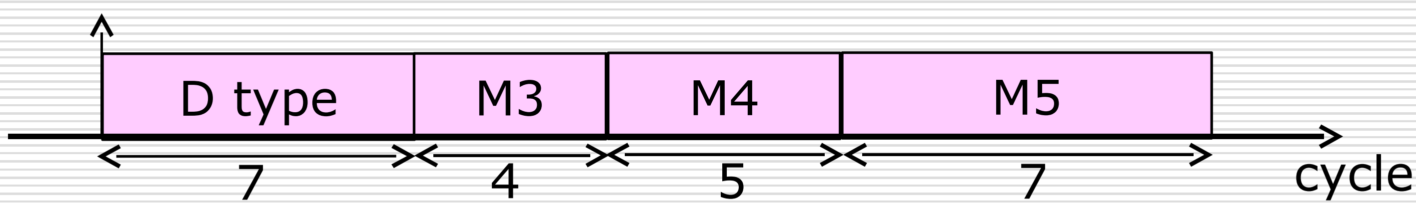
- To satisfy all real-time requirements, we set the value of warning_line
  - considering the worst contending case
  - warning_line = *Σ(maximum possible beat of D_R- and ND_R-type masters) + (maximum possible beat of D-type masters)*

Algorithm of Real-Time Handler (5/5)

	type	R_{cycles}	beat/prob.			interval/ prob.	
M1	D		5/20	6/40	7/40	40/50	50/50
M2	D		4/50	5/20	6/30	60/20	70/80
M3	D_R	200	2/30	3/30	4/40	40/50	60/50
M4	D_R	100	3/20	4/50	5/30	80/10	90/90
M5	ND_R	120	5/30	6/50	7/20	14/50	16/50

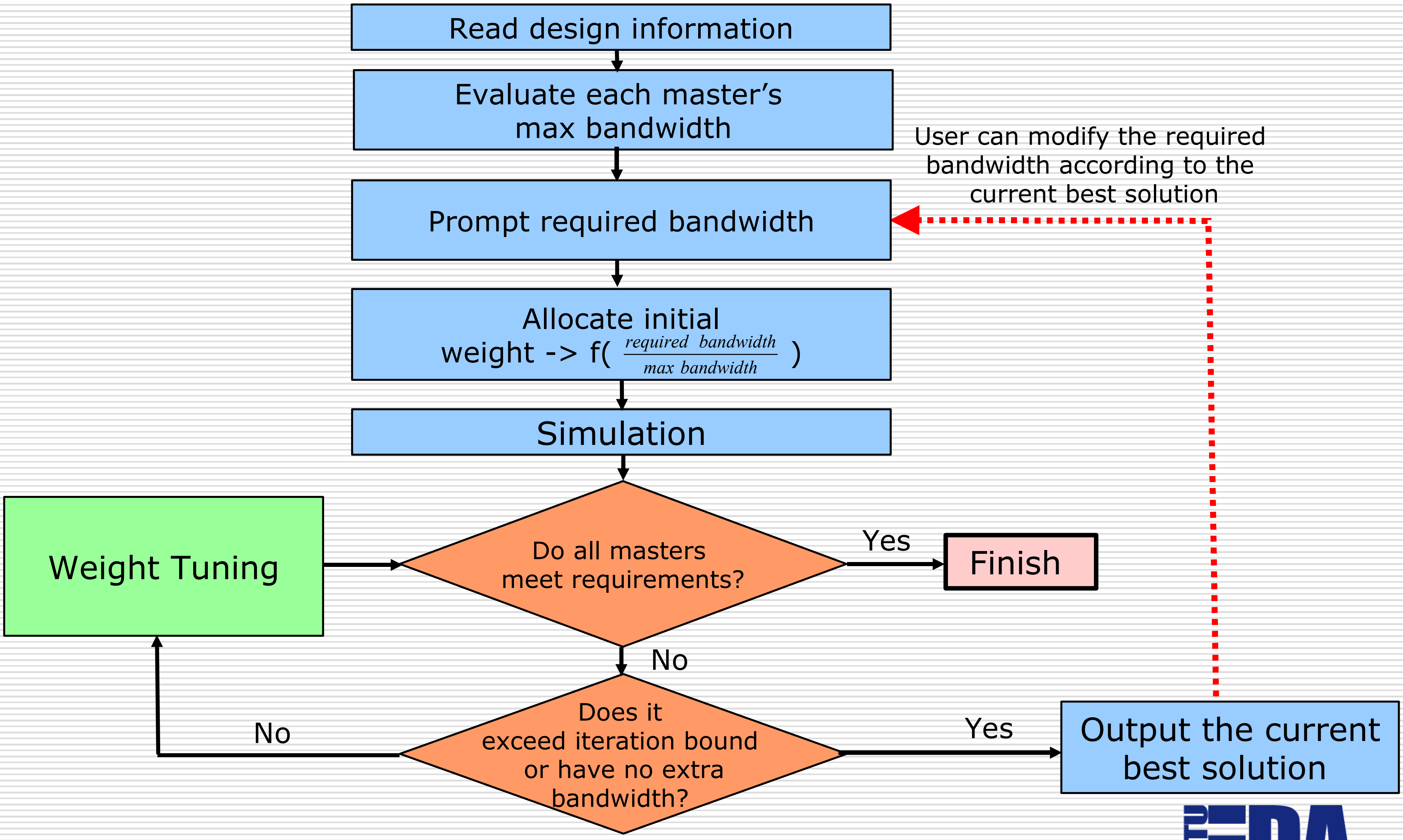
warning_line = (max (2,3,4)+ max(3,4,5)+max (5,6,7))
+ max (5,6,7,4,5,6) = **23**

Worst case

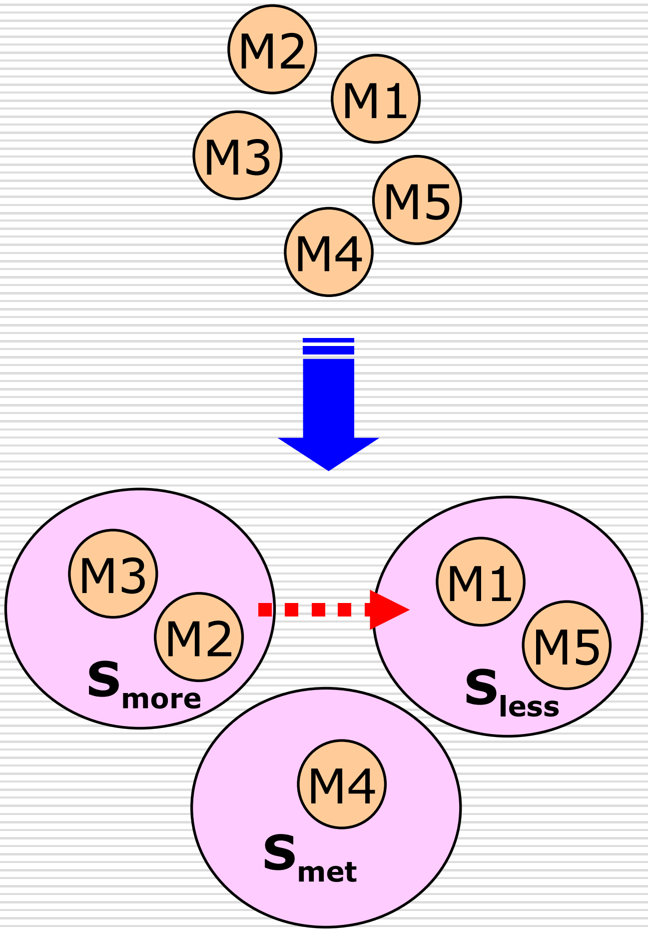


- All hard real-time requirements can be met if R_{cycle} of each matster < (warning_line)

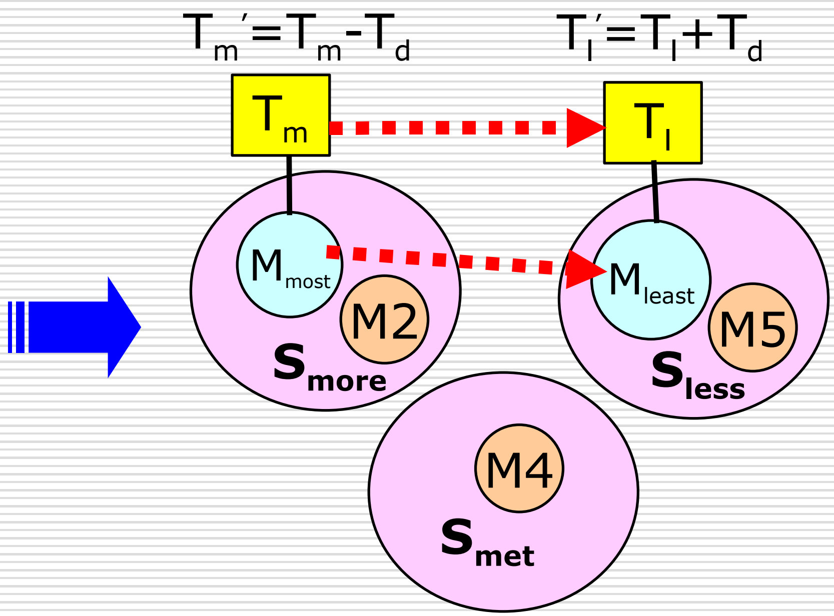
Weight Tuning Flow (2nd Level)



How to Tune Weight ?



- Initially, $T_d=T_m/2$
- If any master in S_{more} or S_{met} -> S_{less}
  - $T_d = T_d / 2$



Outline

- Previous Work
- Our Approach
- Experimental Results and Conclusions

Experimental Setup

- Compare 4 types of arbitration algorithms
  - Lottery
    - assign the number of tickets according to each master's required bandwidth
  - Static Priority
    - the master with higher required bandwidth has higher priority
  - TDM + Lottery
    - 1st level - TDM
    - 2nd level - Lottery without weight tuning
  - RT_lottery

Experiment 1.1 - Input Pattern

Heavy traffic | Light traffic

	type	R_{cycles}	req. BW	beat/prob.		interval/prob.				
Master1	D		20	8/50	16/50	6/10	7/20	8/40	9/20	10/10
Master2	D		5	1/50	4/50	10/10	11/20	12/40	13/20	14/10
Master3	D_R	65	40	8/50	16/50	6/10	7/20	8/40	9/20	10/10
Master4	D_R	85	10	1/50	4/50	10/10	11/20	12/40	13/20	14/10
Master5	ND_R	65	17	8/50	16/50	65/10	66/20	67/40	68/20	69/10
Master6	ND_R	85	2	1/50	4/50	85/10	86/20	87/40	88/20	89/10

94 % in total

Experiment 1.1 – Results

- *bw_miss_num*
  - the number of masters violating bandwidth requirements
- *max_latency*
  - maximum response latency is recorded during simulation
- *rt_vio_time*
  - SUM(the number of real-time violations of all masters' requests)

	bw_miss_num	*max_latency* (cycle)	*rt_vio_time*
Static Priority	3 (50%)	7060	244
Lottery	3 (50%)	954	160
TDM+Lottery	1 (17%)	314	0
RT_lottery	0 (0%)	170	0

6 masters, 10000 cycles

Experiment 1.2 – Random Required Bandwidth Generation

- Randomly generate the required bandwidth
  - R_{sum}
    - SUM (required bandwidth of all masters)
  - higher R_{sum} is usually harder to meet
- The four algorithms are simulated for comparison
- Generate 100 random cases for each R_{sum}
  - R_{sum_i} represents the i^{th} case of simulation for R_{sum}
  - 10k cycles are simulated for each case

Experiment 1.2 – Metrics

- *rt_vio_time_sum*
  - SUM(*rt_vio_time* in each $\boldsymbol{R}_{sum\_i}$)
- *rt_fail_sum*
  - number of total real-time failed cases in the simulation
  - if *rt_vio_time* > 0 in $\boldsymbol{R}_{sum\_i} \Rightarrow \boldsymbol{R}_{sum\_i}$ is a real-time failed case
- *bw_fail_sum*
  - number of total bandwidth failed cases in the simulation
  - if *bw_miss_num* > 0 in $\boldsymbol{R}_{sum\_i} \Rightarrow \boldsymbol{R}_{sum\_i}$ is a bandwidth failed case
- *fail_sum*
  - number of total failed cases in the simulation
  - if (*rt_vio_time* > 0 or *bw_miss_num* > 0) in $\boldsymbol{R}_{sum\_i} \Rightarrow \boldsymbol{R}_{sum\_i}$ is a failed case

Experiment 1.2 – Results

RT_lottery

R_{sum}	*rt_v*	*bw_f*	*rt_f*	*fail*
95	0	87	0	87
90	0	80	0	80
85	0	79	0	79
80	0	68	0	68
75	0	66	0	66
70	0	57	0	57
65	0	38	0	38

Lottery

R_{sum}	*rt_v*	*bw_f*	*rt_f*	*fail*
95	12915	99	100	100
90	12150	97	100	100
85	11159	98	100	100
80	10535	86	100	100
75	9007	73	100	100
70	9022	58	100	100
65	8274	45	100	100

TDM+ Lottery

R_{sum}	*rt_v*	*bw_f*	*rt_f*	*fail*
95	1	99	1	99
90	8	96	8	96
85	8	95	8	96
80	6	91	6	91
75	6	83	6	84
70	3	75	3	75
65	2	58	2	58

Static Priority

R_{sum}	*rt_v*	*bw_f*	*rt_f*	*fail*
95	18577	100	100	100
90	17396	100	100	100
85	13739	100	99	100
80	14235	98	100	100
75	11200	88	99	100
70	11076	83	97	97
65	10345	82	96	98

rt_v : *rt_vio_time_sum*　　*rt_f* : *rt_fail_sum*

bw_f : *bw_fail_sum*　　*fail* : *fail_sum*

Conclusions of Experiment 1

Arbitration algorithm	Real-time capability	Bandwidth allocation capability
RT_lottery	Always holds	Best
TDM + Lottery	Fails for few critical cases	Good but requiring weight tuning
Lottery	No consideration	Good but requiring weight tuning
Static Priority	No consideration	Poor

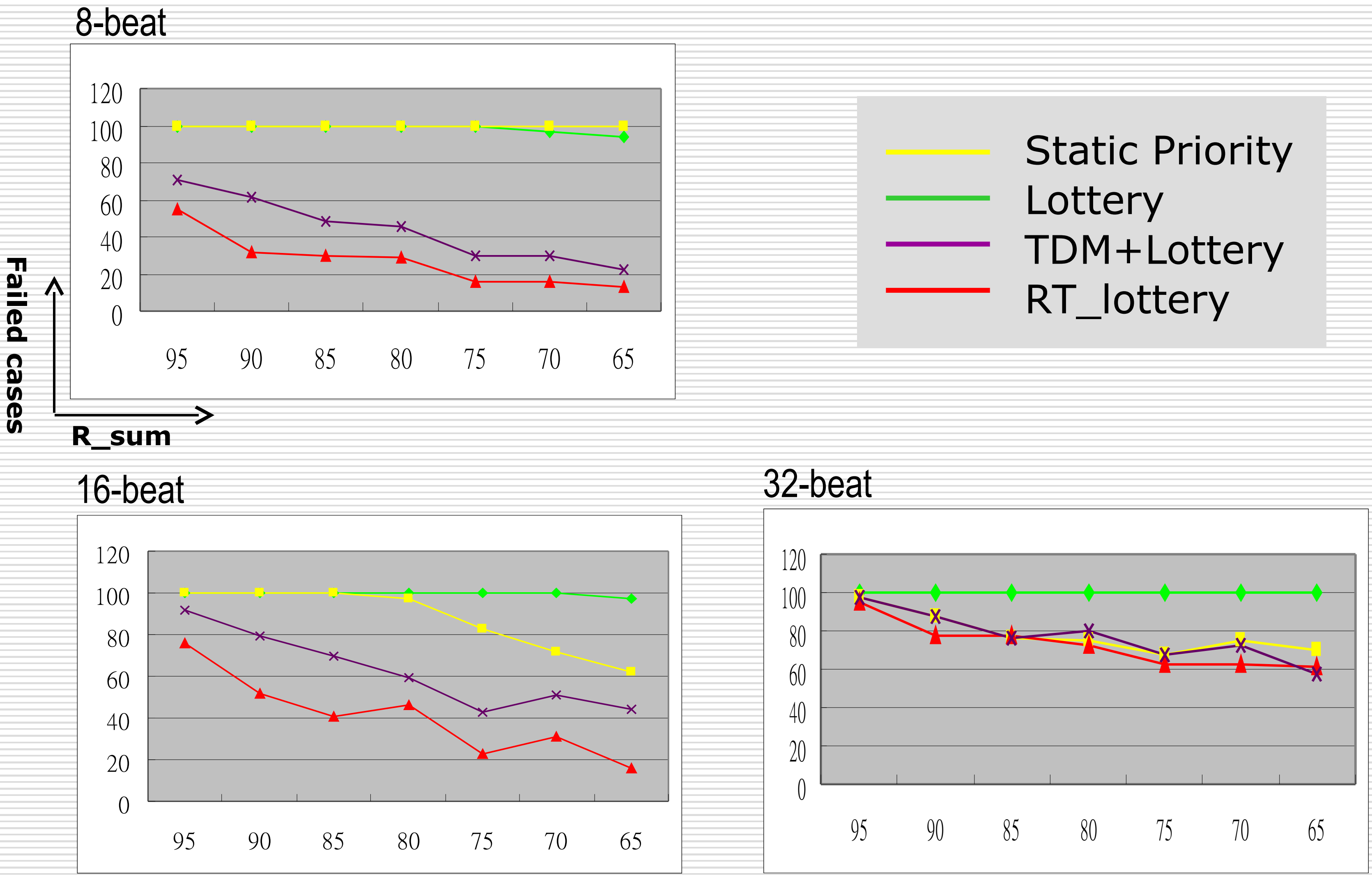
- Number of failed cases in different R_{sum}
  - RT_lottery < (TDM + Lottery) < Lottery ≈ Static Priority

Experiment 2 : Beat Number

- Objective
  - observe the effect of beat numbers on arbitration algorithms
- Three scenarios are simulated:

	type	R_{cycles}	beat/prob.			interval/prob.				
			(a)	(b)	(c)					
M1	D		8/100	16/100	32/100	6/10	7/20	8/40	9/20	10/10
M2	D_R	100	8/100	16/100	32/100	6/10	7/20	8/40	9/20	10/10
M3	ND_R	100	8/100	16/100	32/100	100/10	101/20	102/40	103/20	104/10

Trend of Failed Cases for 100 Random Cases



Conclusions of Experiment 2

- RT_lottery and TDM + Lottery are much better than the other arbitration algorithms
  - have capability of handling both bandwidth and real-time requirements
  - RT_lottery is the best in our experiments
- For RT_lottery and TDM + Lottery
  - number of failed cases arises with larger fixed beat numbers
    - the granularity of weight (number of tickets) gets coarser with larger fixed beat number

Summary

- The two-level arbitration algorithm, RT_lottery with weight tuning, is proposed
  - handle both bandwidth and real-time requirements
- Experimental results show that RT_lottery is the best among the four arbitration algorithms

Thank you!