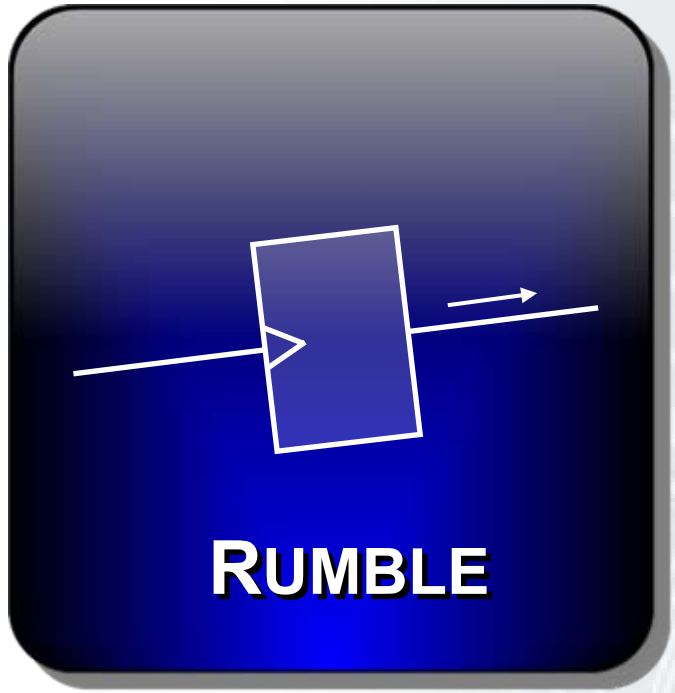
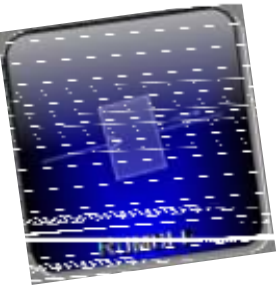
RUMBLE : Rip Up and Move Boxes with Linear Evaluation



D. A. Papa*, T. Luo, M. D. Moffitt, C. N. Sze, Z. Li,
G.-J. Nam, C. J. Alpert and I. L. Markov*

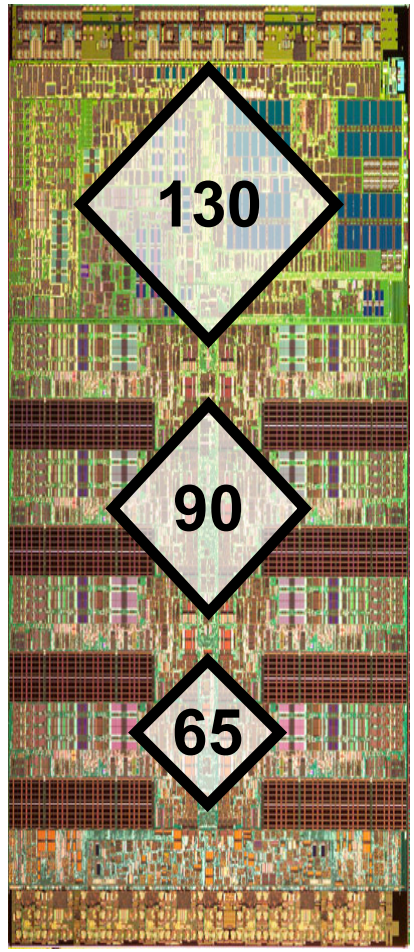
* University of Michigan

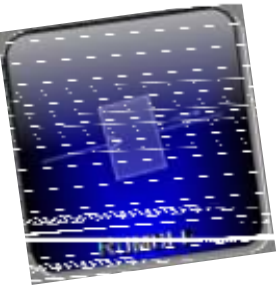
ISPD 2008 April 14, 2008



Decreasing Speed Limits for On-chip Traffic

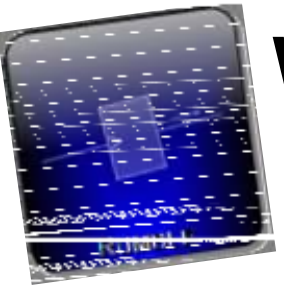
- Physical limits for on-chip traffic
- Technology scaling limits the propagation distance over one clock cycle
  - Increased interconnect resistivity
- IBM's Design Productivity Group supports
  - High performance ASIC and SOC designs
  - Processors (power6, power7, bluegene, etc.)



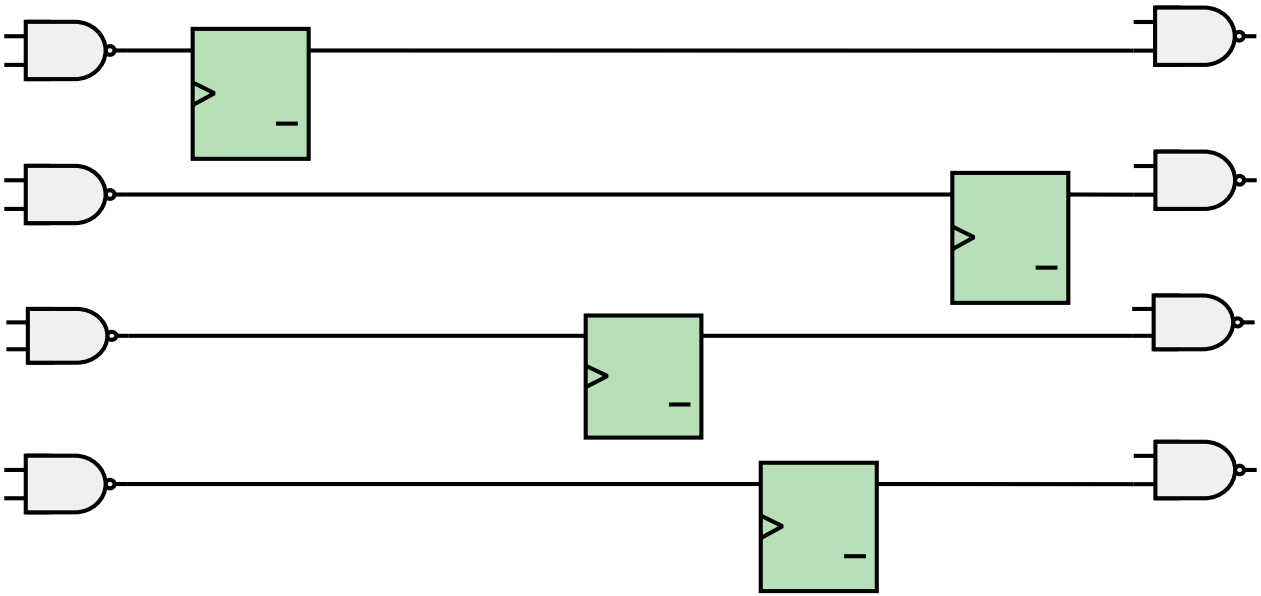


Multi-cycle Datapaths Need Pipeline Latches

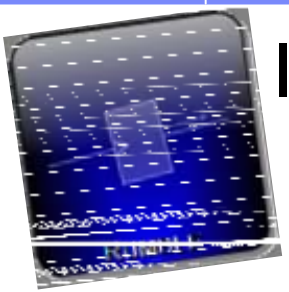
- Lower speed limits → more stops along the way
- Designers insert latches to pipeline global signals
- Pipeline latches — a bottleneck to timing closure
  - Traditional placement doesn't handle them well
  - May come as a surprise (found late in the flow)



Why Wirelength-driven Placement Falls Short

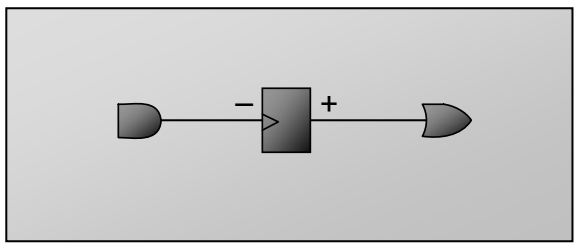


- Optimal wirelength may not have good timing
- Timing influenced net weights always pull to one side
- The center is not right either…
- There may be only a single optimal point
- How do we find it?

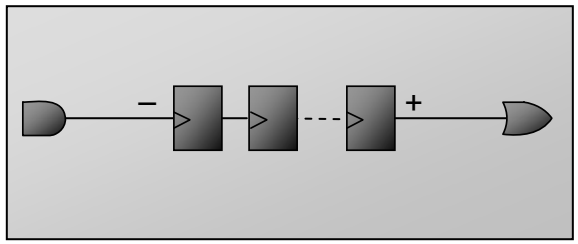


Incremental Timing-Driven Placement

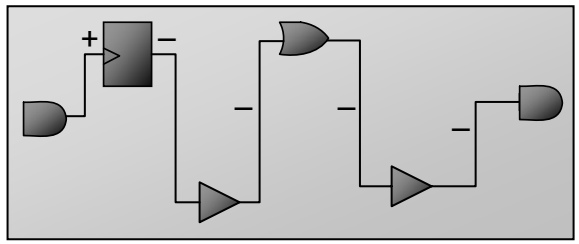
- Poor latch timing is not evident until after optimization
- Returning to global placement costs a lot of runtime
- Low placement stability may prohibit convergence
- Calls for an incremental solution



The
"Imbalanced Latch"
Problem

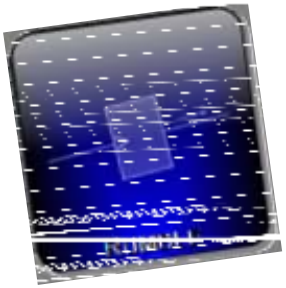


The
"Pipeline Latch"
Problem



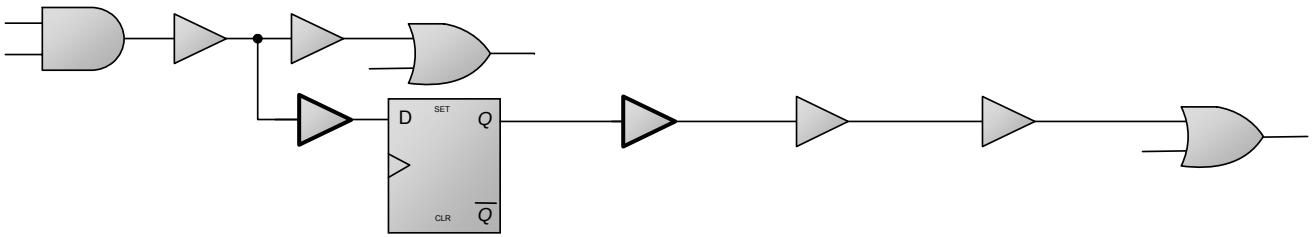
The
"Crooked Path"
Problem

Three Variations of ONE Problem: *Poorly Placed Latches on Critical Paths*

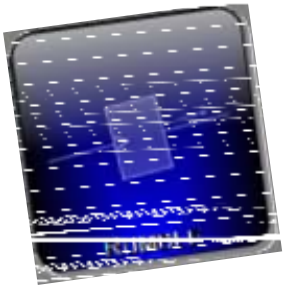


RUMBLE: Rip-up and Move Boxes with Linear Evaluation

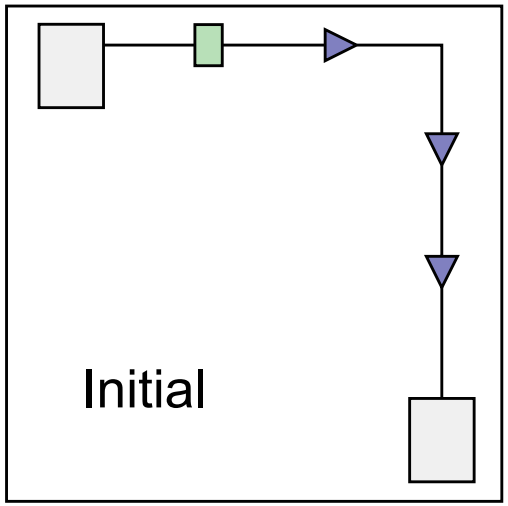
- **Incremental placement for timing improvement is not new…**
  - Various heuristics to move individual latches / gates exist [Wang05, Chowdhary05]
- **Problem:** Neighboring logic (esp. buffers) prohibit local improvements

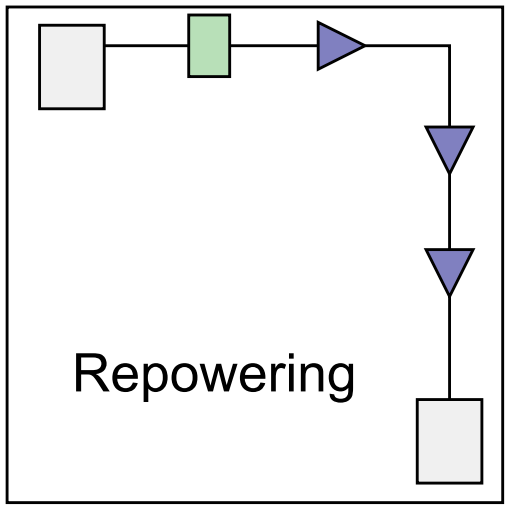


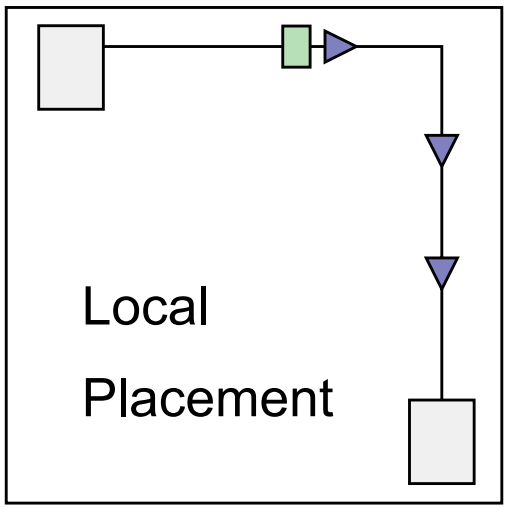
- **Our Solution:**
  - Remove repeaters and use linear delay model
    - Delay of a buffered net is linearly proportional to its length
    - Accurately represents *virtually buffered* nets
  - Use linear programming to efficiently calculate optimal solution

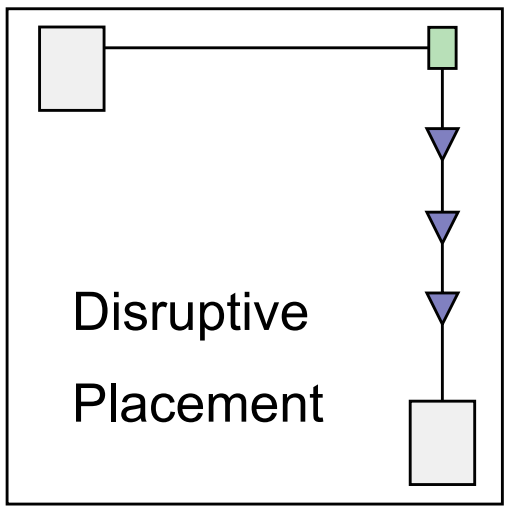


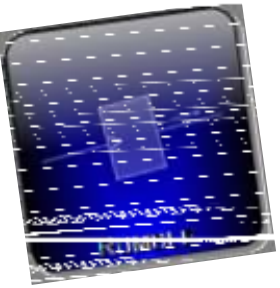
Problems with Existing Solutions





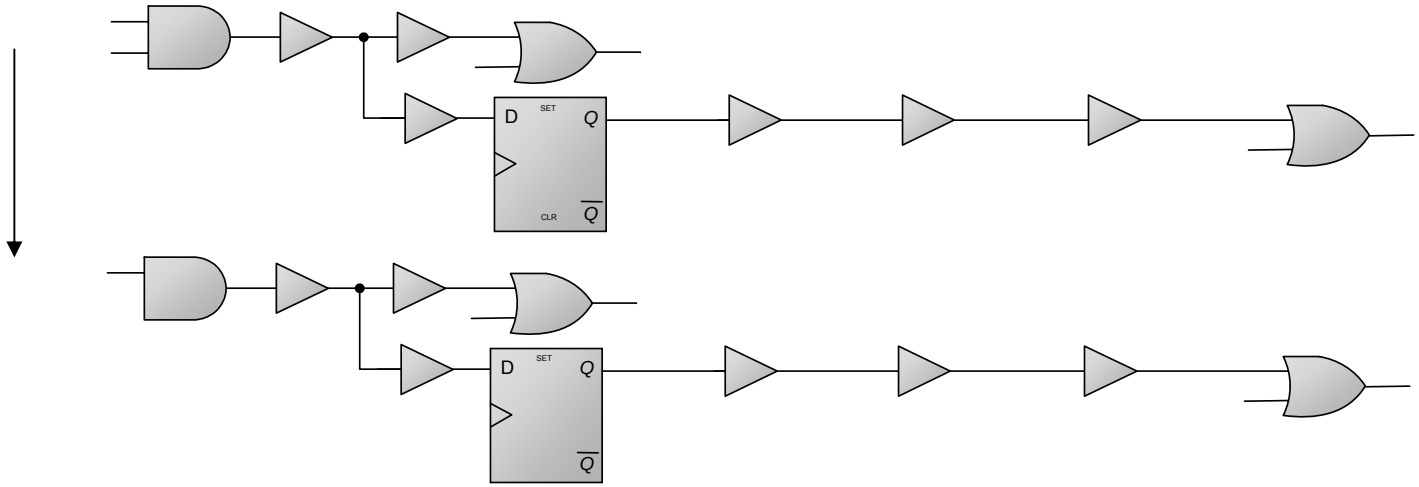


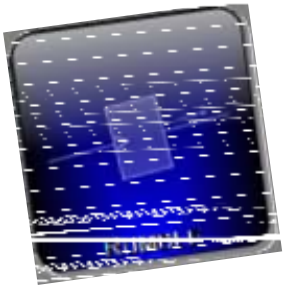




RUMBLE: Rip-up and Move Boxes with Linear Evaluation

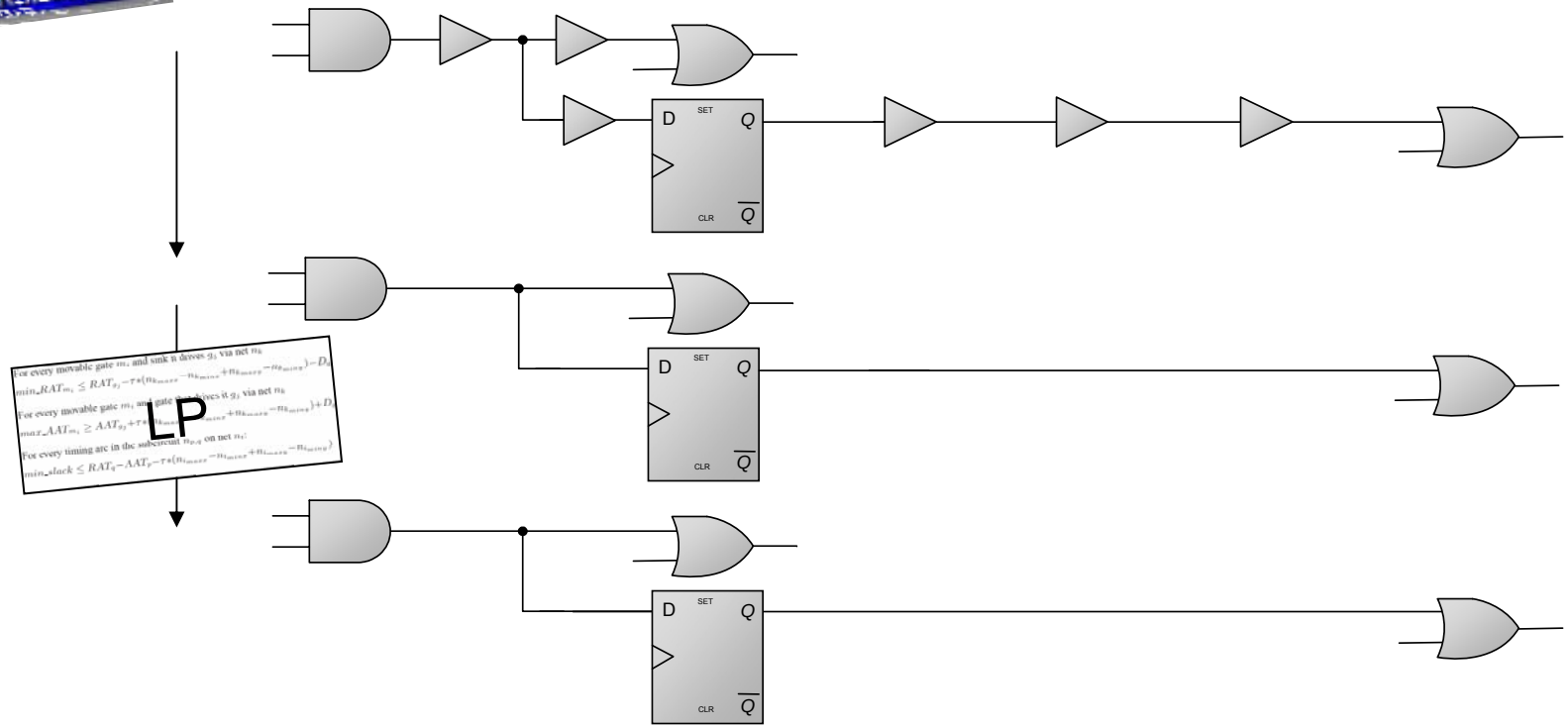
- **Conceptual Step (1):** *Rip out Buffer Trees*

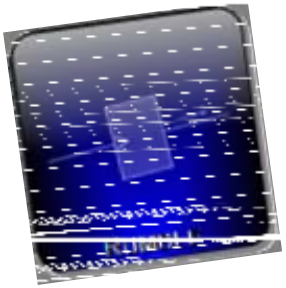




RUMBLE: Rip-up and Move Boxes with Linear Evaluation

- **Conceptual Step (2):** *Cast Optimization as Linear Program*

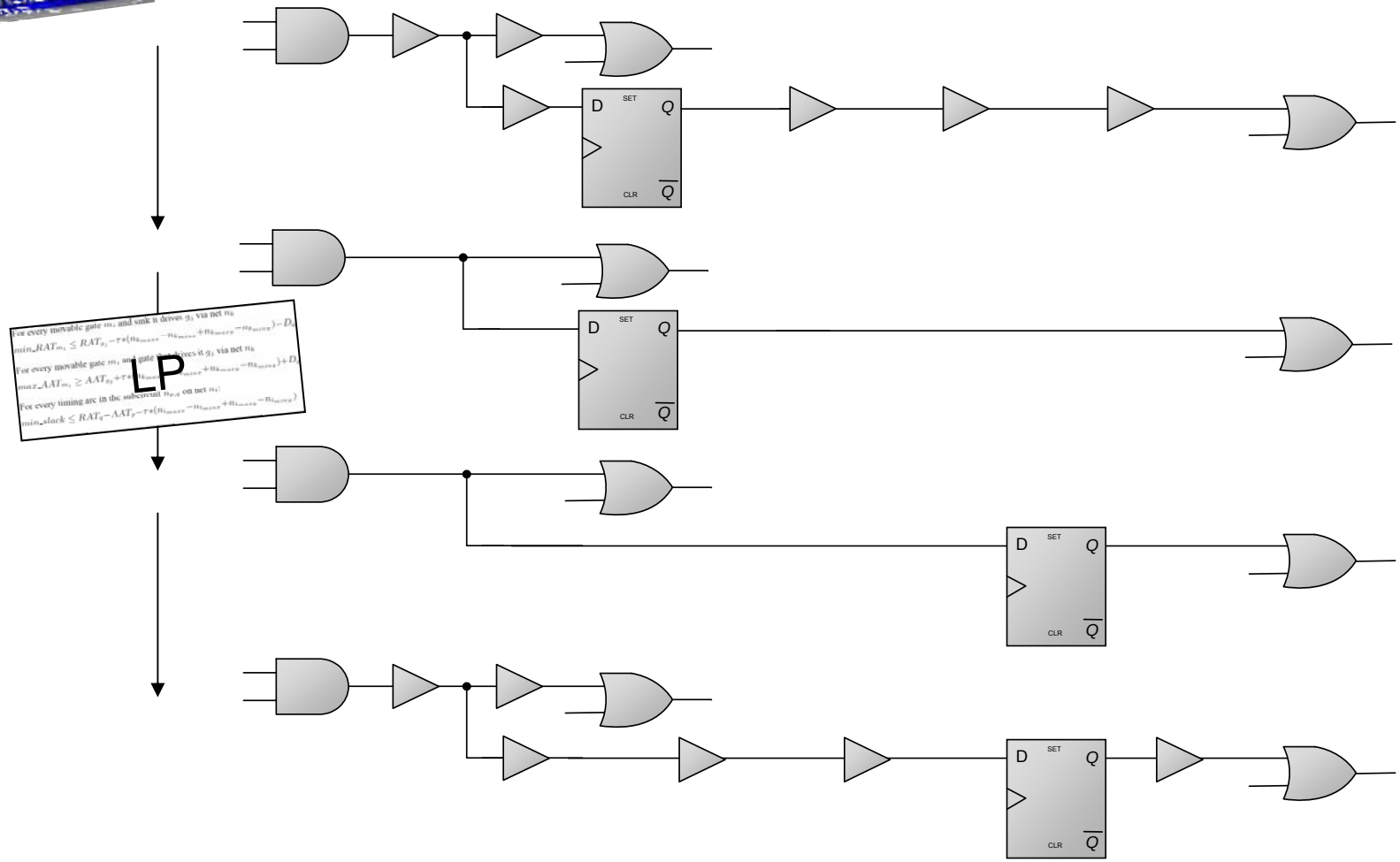


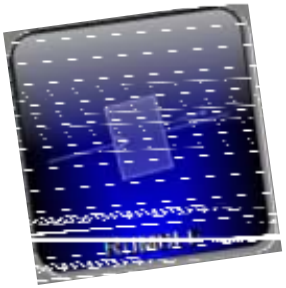


RUMBLE: Rip-up and Move Boxes with Linear Evaluation

- **Conceptual Step (3):**

Legalize, Rebuffer, and Measure Improvement

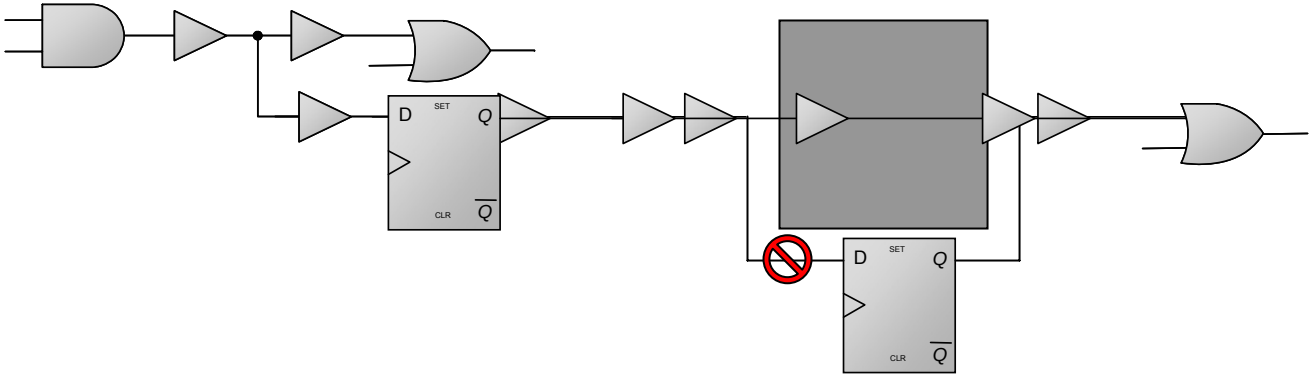




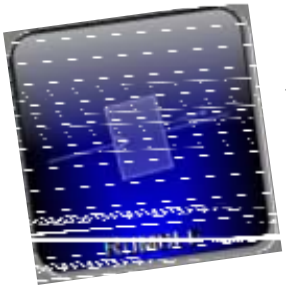
RUMBLE: Rip-up and Move Boxes with Linear Evaluation

"Do No Harm"

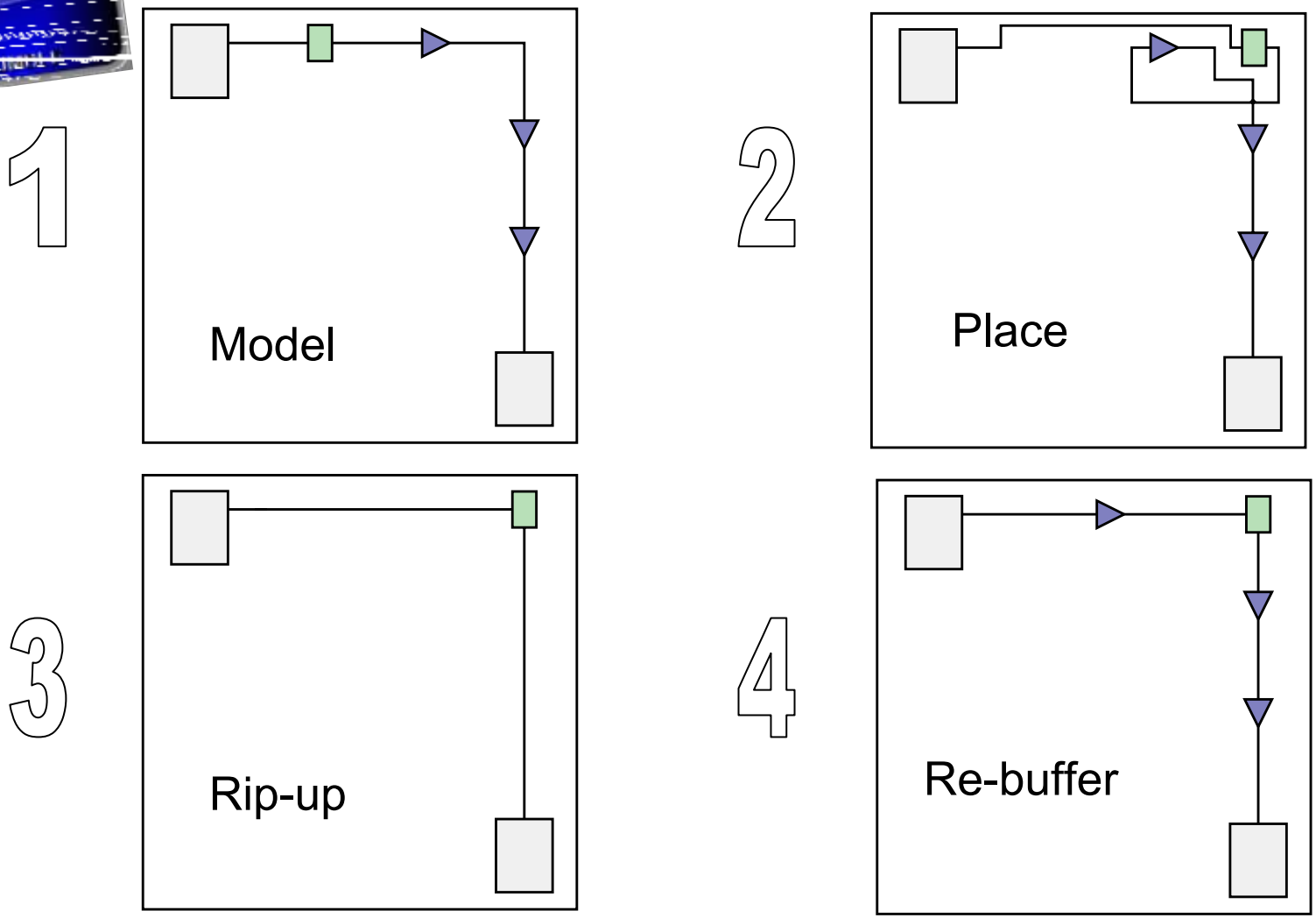
- **What if something goes wrong?**

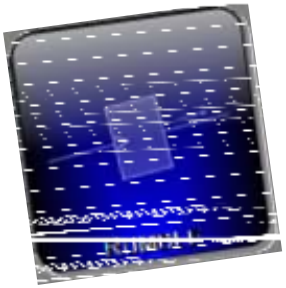


- **Restore the original state**

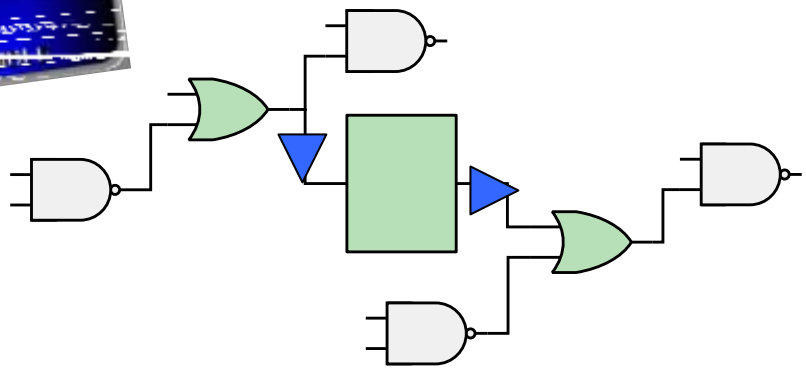


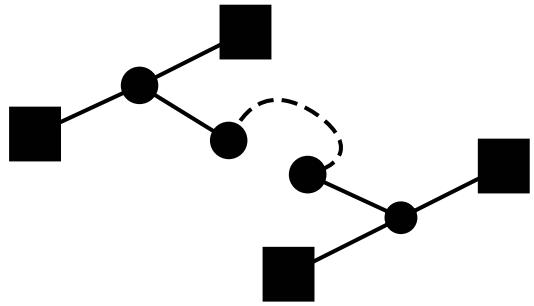
A Physical View of the RUMBLE Flow





Subcircuit Timing Model

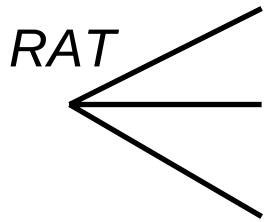


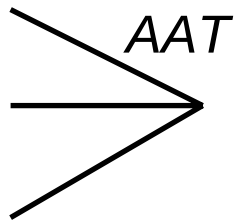


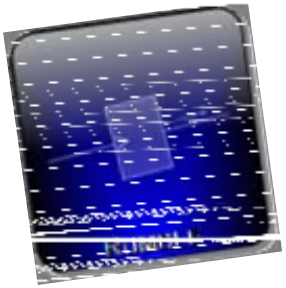
- Gather the neighbors (gray) of movable gates (green)
- Decompose nets into two-pin timing arcs
- Delay of arc proportional to Manhattan distance
- Timing points on neighbors are fixed
- Timing does not propagate through latches

The RUMBLE Linear Program

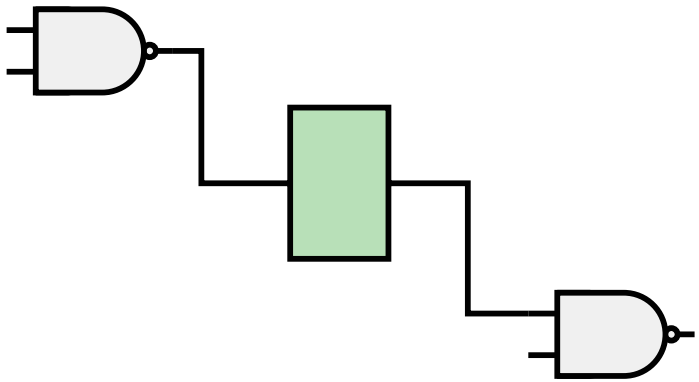
- Delay (timing arc) = length(timing arc) * constant
  - Constant is an empirically determined technology dependent parameter
- Objective: **Maximize** minimum_slack
- Variables: (x, y) coordinates of each movable object, minimum_slack
- Constraints implement static timing analysis
  - RAT := Required Arrival Time
  - AAT := Actual Arrival Time
  - For each movable gate G
    - $RAT = \min(RAT_{output} - delay(arc) - delay(G))$
    - $AAT = \max(AAT_{input} + delay(arc) + delay(G))$
  - $minimum\_slack = \min(RAT_{output} - AAT_{input} - delay(L))$

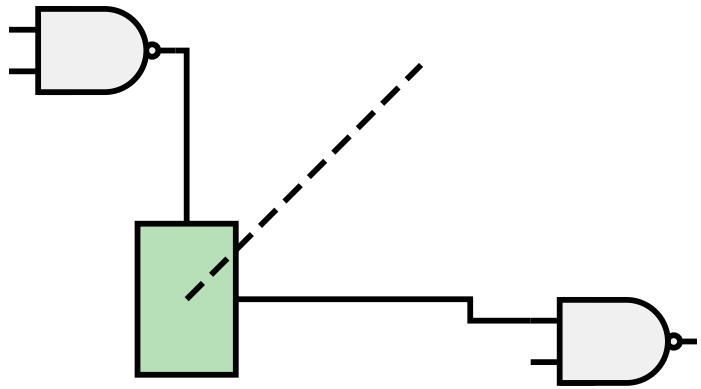




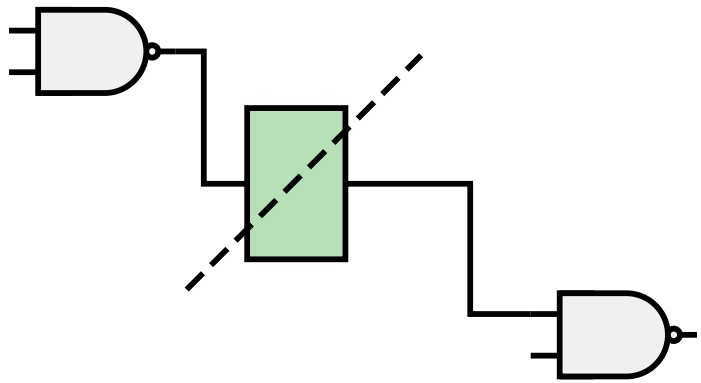


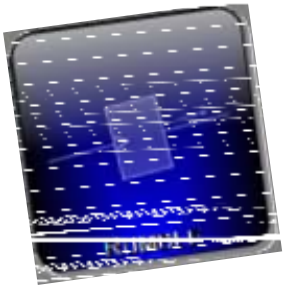
Displacement Minimization Objective



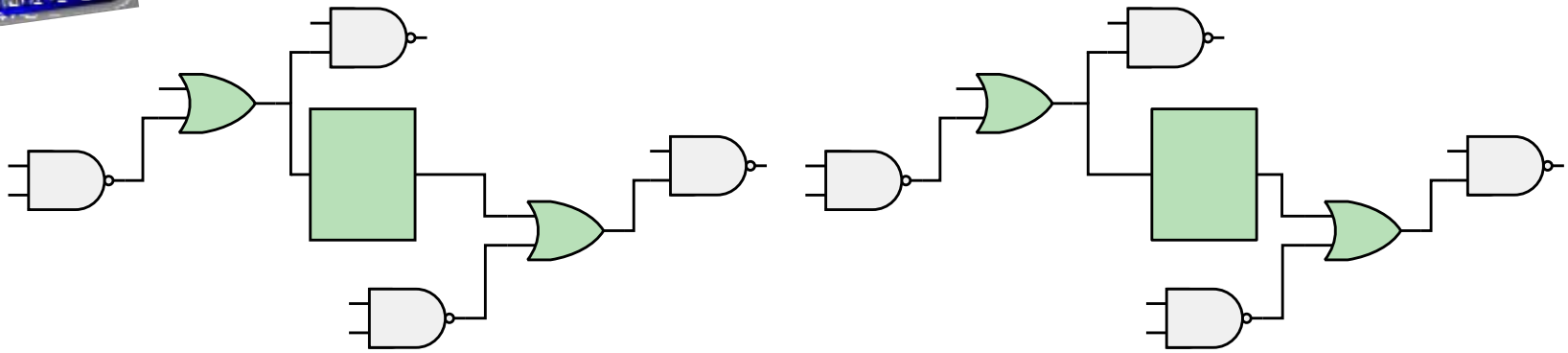


- Many solutions may have optimal slack
- Add a bias to original location
  - Prevents LP solvers' "lower left preference"
  - Mitigates legalization problems
  - Helps preserve initial timing conditions



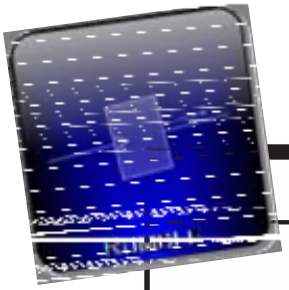


FOM Objective



- FOM = Σ ($slack_n$ – threshold)
- Objective function emphasizes worst slack
  - Worst slack may not be improvable
- Additional term for sum of negative slack
  - Some negative paths may be improved

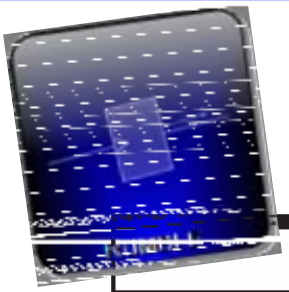
-20



Experiment 1: Keep Buffers

Implications of keeping buffers						
	KEEP-BUFFERS Slack (ps)			RUMBLE Slack (ps)		
Subcircuit	orig	new	imprv.	orig	new	imprv.
latch A0	-1480	-1318	**162**	-1480	26	**1506**
latch A1	-1268	-1066	**202**	-1268	186	**1454**
latch A2	-1020	-939	**80**	-1020	-791	**229**
latch A3	-953	-766	**187**	-953	-390	**563**
latch A4	-897	-677	**220**	-897	356	**1253**
latch A5	-848	-746	**101**	-848	-278	**570**
latch A6	-690	-690	**0**	-690	395	**1085**
latch A7	-645	-586	**59**	-645	-19	**626**
latch A8	-633	-560	**74**	-633	290	**923**
latch A9	-610	-466	**144**	-610	262	**872**
avg	*-904*	*-782*	*123*	*-904*	*4*	*908*

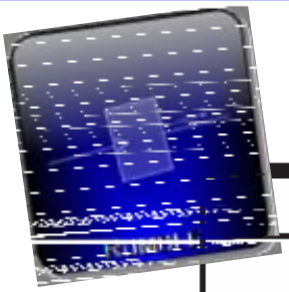
- *Rebuffering dramatically improves performance*



Experiment 2: Linear Delay Model vs. Industrial Timer

Model timing vs. reference timing						
	Model slack (ps)			Subcircuit slack (ps)		
Subcircuit	orig	new	imprv.	orig	new	imprv.
latch A0	-1799	-48	**1751**	-1480	26	**1506**
latch A1	-1509	65	**1574**	-1268	186	**1454**
latch A2	-1113	-868	**245**	-1020	-791	**229**
latch A3	-1147	-527	**620**	-953	-390	**563**
latch A4	-1090	180	**1269**	-897	356	**1253**
latch A5	-945	-295	**650**	-848	-278	**570**
latch A6	-920	320	**1241**	-690	395	**1085**
latch A7	-886	49	**935**	-645	-19	**626**
latch A8	-913	213	**1126**	-633	290	**923**
latch A9	-800	397	**1198**	-610	262	**872**
avg	*-1112*	*-51*	*1061*	*-904*	*4*	*908*

97% correlation



Experiment 3: How good is RUMBLE ?

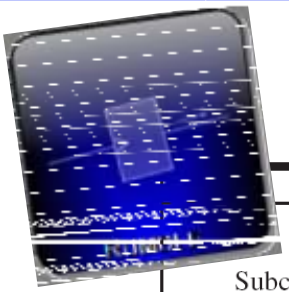
Center-of-gravity vs. RUMBLE

Subcircuit	COG Slack (ps) orig	COG Slack (ps) new	COG Slack (ps) imprv.	RUMBLE Slack (ps) orig	RUMBLE Slack (ps) new	RUMBLE Slack (ps) imprv.
latch A0	-1480	-527	**953**	-1480	26	**1506**
latch A1	-1268	-203	**1065**	-1268	186	**1454**
latch A2	-1020	-800	**219**	-1020	-791	**229**
latch A3	-953	-615	**338**	-953	-390	**563**
latch A4	-897	-78	**819**	-897	356	**1253**
latch A5	-848	-319	**529**	-848	-278	**570**
latch A6	-690	-690	**0**	-690	395	**1085**
latch A7	-645	-645	**0**	-645	-19	**626**
latch A8	-633	-633	**0**	-633	290	**923**
latch A9	-610	67	**677**	-610	262	**872**
avg	*-904*	*-444*	*460*	*-904*	*4*	*908*

“Do no harm”
preventing degradation

20.9% Cycle time

41.3% Cycle time



Experiment 4: Multi-move RUMBLE vs. Single

Iterated RUMBLE vs. RUMBLE: 1-hop

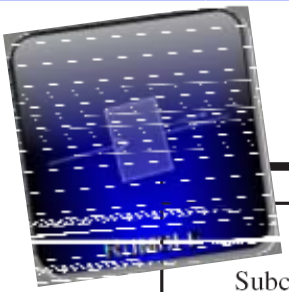
Subcircuit	Iterated single-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.	Multi-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.
subcircuit B0	-1542	-1542	**0**	-6091	-6091	**0**	-1542	-130	**1412**	-6091	-130	**5962**
subcircuit B1	-1501	-277	**1223**	-5924	-277	**5647**	-1501	55	**1556**	-5924	0	**5924**
subcircuit B2	-1240	-1240	**0**	-4354	-4354	**0**	-1240	-980	**261**	-4354	-4044	**310**
subcircuit B3	-848	-278	**569**	-2523	-812	**1710**	-848	-279	**569**	-2523	-813	**1709**
subcircuit B4	-690	-79	**612**	-4090	-79	**4011**	-690	202	**893**	-4090	0	**4090**
subcircuit B5	-690	48	**739**	-2053	0	**2053**	-690	290	**980**	-2053	0	**2053**
subcircuit B6	-645	-18	**627**	-1921	-32	**1889**	-645	301	**945**	-1921	0	**1921**
subcircuit B7	-595	86	**681**	-1937	0	**1937**	-595	503	**1098**	-1937	0	**1937**
subcircuit B8	-444	-444	**0**	-889	-889	**0**	-444	-92	**352**	-889	-191	**698**
subcircuit B9	-418	-46	**372**	-857	-46	**811**	-418	6	**424**	-857	0	**857**
avg	*-861*	*-379*	*482*	*-3064*	*-1258*	*1806*	*-861*	*-12*	*849*	*-3064*	*-518*	*2546*

Iterated RUMBLE vs. RUMBLE: 2-hop

Subcircuit	Iterated single-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.	Multi-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.
subcircuit C0	-719	-719	**0**	-8313	-8313	**0**	-719	-675	**44**	-8313	-5028	**3285**
subcircuit C1	-719	-719										**517**
subcircuit C2	-690	-79										**090**
subcircuit C3	-690	-79										**090**
subcircuit C4	-681	-349										**707**
subcircuit C5	-645	-91										**767**
subcircuit C6	-645	-33										**767**
subcircuit C7	-318	-318										**940**
subcircuit C8	-490	227										**966**
subcircuit C9	-217	-217	**0**	-652	-652	**0**	-217	60	**277**	-652	0	**652**
avg	*-581*	*-238*	*344*	*-3846*	*-1877*	*1968*	*-581*	*92*	*673*	*-3846*	*-957*	*2888*

Simultaneous mo
is better th
one latch at a

Simultaneous movement
is better than
one latch at a time



Experiment 4: Multi-move RUMBLE vs. Single

Iterated RUMBLE vs. RUMBLE: 1-hop

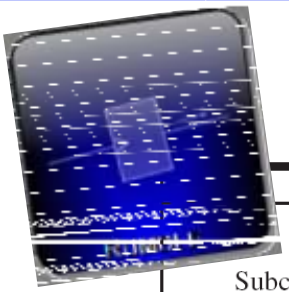
Subcircuit	Iterated single-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.	Multi-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.
subcircuit B0	-1542	-1542	**0**	-6091	-6091	**0**	-1542	-130	**1412**	-6091	-130	**5962**
subcircuit B1	-1501	-277	**1223**	-5924	-277	**5647**	-1501	55	**1556**	-5924	0	**5924**
subcircuit B2	-1240	-1240										
subcircuit B3	-848	-278										
subcircuit B4	-690	-79										
subcircuit B5	-690	48										
subcircuit B6	-645	-18										
subcircuit B7	-595	86										
subcircuit B8	-444	-444										
subcircuit B9	-418	-46										
avg	*-861*	*-379*	*482*	*-3064*	*-1258*	*1806*	*-861*	*-12*	*849*	*-3064*	*-518*	*2546*

Simultaneous m is better t one latch at

Simultaneous movement is better than one latch at a time

Iterated RUMBLE vs. RUMBLE: 2-hop

Subcircuit	Iterated single-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.	Multi-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.
subcircuit C0	-719	-719	**0**	-8313	-8313	**0**	-719	-675	**44**	-8313	-5028	**3285**
subcircuit C1	-719	-719	**0**	-8004	-8004	**0**	-719	-653	**66**	-8004	-4386	**3617**
subcircuit C2	-690	-79	**612**	-4090	-79	**4011**	-690	314	**1004**	-4090	0	**4090**
subcircuit C3	-690	-79	**612**	-4090	-79	**4011**	-690	337	**1027**	-4090	0	**4090**
subcircuit C4	-681	-349	**333**	-3865	-349	**3516**	-681	158	**524**	-3865	-158	**3707**
subcircuit C5	-645	-91	**554**	-3767	-306	**3462**	-645	371	**1015**	-3767	0	**3767**
subcircuit C6	-645	-33	**612**	-3767	-52	**3716**	-645	324	**969**	-3767	0	**3767**
subcircuit C7	-318	-318	**0**	-940	-940	**0**	-318	531	**848**	-940	0	**940**
subcircuit C8	-490	227	**716**	-966	0	**966**	-490	466	**956**	-966	0	**966**
subcircuit C9	-217	-217	**0**	-652	-652	**0**	-217	60	**277**	-652	0	**652**
avg	*-581*	*-238*	*344*	*-3846*	*-1877*	*1968*	*-581*	*92*	*673*	*-3846*	*-957*	*2888*



Experiment 4: Multi-move RUMBLE vs. Single

Iterated RUMBLE vs. RUMBLE: 1-hop

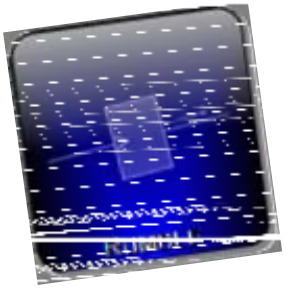
Subcircuit	Iterated single-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.	Multi-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.
subcircuit B0	-1542	-1542	**0**	-6091	-6091	**0**	-1542	-130	**1412**	-6091	-130	**5962**
subcircuit B1	-1501	-277	**1223**	-5924	-277	**5647**	-1501	55	**1556**	-5924	0	**5924**
subcircuit B2	-1240	-1240	**0**	-4354	-4354	**0**	-1240	-980	**261**	-4354	-4044	**310**
subcircuit B3	-848	-278	**569**	-2523	-812	**1710**	-848	-279	**569**	-2523	-813	**1709**
subcircuit B4	-690	-79	**612**	-4090	-79	**4011**	-690	202	**893**	-4090	0	**4090**
subcircuit B5	-690	48	**739**	-2053	0	**2053**	-690	290	**980**	-2053	0	**2053**
subcircuit B6	-645	-18	**627**	-1921	-32	**1889**	-645	301	**945**	-1921	0	**1921**
subcircuit B7	-595	86	**681**	-1937	0	**1937**	-595	503	**1098**	-1937	0	**1937**
subcircuit B8	-444	-444	**0**	-889	-889	**0**	-444	-92	**352**	-889	-191	**698**
subcircuit B9	-418	-46	**372**	-857	-46	**811**	-418	6	**424**	-857	0	**857**
avg	*-861*	*-379*	*482*	*-3064*	*-1258*	*1806*	*-861*	*-12*	*849*	*-3064*	*-518*	*2546*

less worst-slack improvement on larger subcircuits

ed RUMBLE vs. RUMBLE: 2-hop

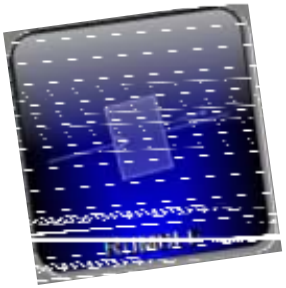
	move RUMBLE FOM (ps) orig	new	imprv.	Multi-move RUMBLE Slack (ps) orig	new	imprv.	FOM (ps) orig	new	imprv.
sub	-8313	-8313	**0**	-719	-675	**44**	-8313	-5028	**3285**
sub	-8004	-8004	**0**	-719	-653	**66**	-8004	-4386	**3617**
subcircuit C2	-4090	-79	**4011**	-690	314	**1004**	-4090	0	**4090**
su	4090	-79	**4011**	-690	337	**1027**	-4090	0	**4090**
su	-3865	-349	**3516**	-681	-158	**524**	-3865	-158	**3707**
su	-3767	-306	**3462**	-645	371	**1015**	-3767	0	**3767**
su	-3767	-52	**3716**	-645	324	**969**	-3767	0	**3767**
su	-940	940	**0**	-318	531	**848**	-940	0	**940**
su	-966	0	**966**	-490	466	**956**	-966	0	**966**
su	-652	-652	**0**	-217	60	**277**	-652	0	**652**
	-3846	*-1877*	*1968*	*-581*	*92*	*673*	*-3846*	*-957*	*2888*

greater FOM improvement on larger subcircuits



Conclusions

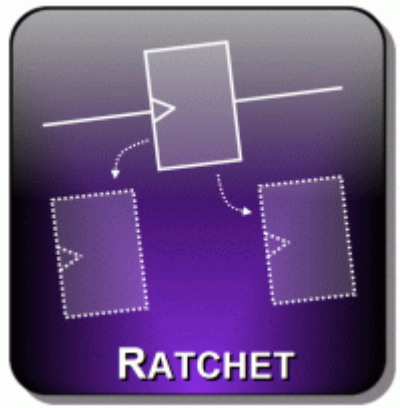
- Challenge:
  - Aggressive performance requirements + technology scaling = **increased use of pipeline latches**
  - Traditional placement leads to **poor timing-driven latch placement**
- Proposed Techniques:
  - **Remove repeaters** to allow freedom to move
  - Use a **linear wire-delay model** that is accurate for long wires
    - Solve using **LP** to place subcircuits
  - Primary objective: worst slack
    - Secondary objectives: displacement and FOM
- RUMBLE improves upon existing solutions because it **sees through repeaters**
- The "**do no harm" philosophy** is crucial to prevent unforeseen degradation



Future Work

- **Handling blockages more intelligently**
  - There can be many optimal locations, choose a legal one
  - Formulate and solve for the best location disallowing blocked locations
- **Smarter subcircuit selection**
  - Only explore the critical region
  - Only rip out the buffers you need to

Also, stay tuned for DAC08…



Questions?

Backup

```
RUMBLE (Gate movable)
1     subcircuit = Build-Subcircuit-From-Seed(movable, 0)
2     before_timing = measure_timing(subcircuit)
3     initial_solution.create_interconnect_cache(subcircuit)
4     initial_solution.before_locs = get_locations(movables)
5     Build LP the RUMBLE linear program for subcircuit
6     after_locs = LP.solve()
7     set_gates_locations(movables, after_locs)
8     initial_solution.rip_up_buffers()
9     phys_syn_opt(movables, initial_solution.get_nets());
10    after_timing = measure_timing(subcircuit)
11    if(after_timing worse than before_timing)
12      set_locations(movables, initial_solution.before_locs)
13      initial_solution.restore_interconnect()
```