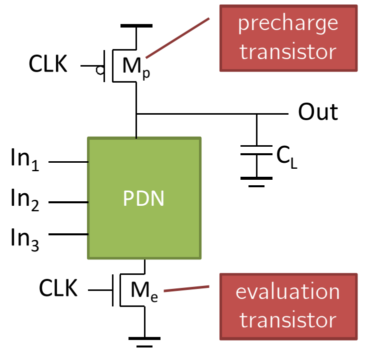
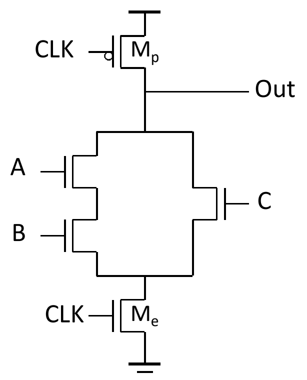
ECE4740: Digital VLSI Design

Lecture 16: Domino logic

Recap: dynamic logic





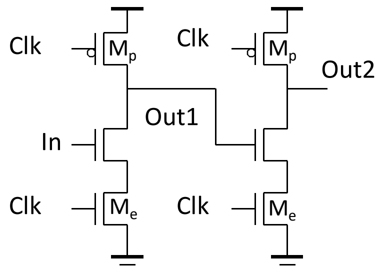
- Two-phase operation:
  - Precharge → CLK=0
  - Evaluation → CLK=1

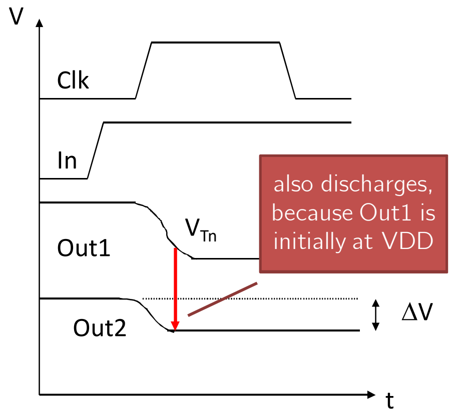
Domino logic

Cascading dynamic gates

Cascading causes problems

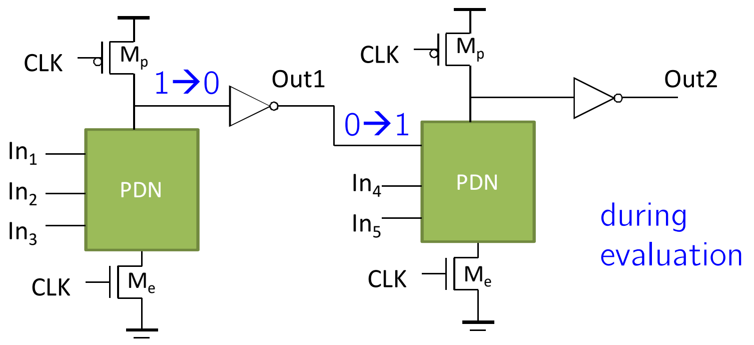
Two inverters:





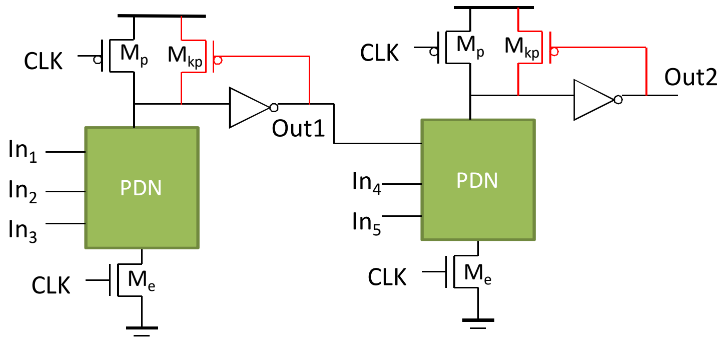
- Only a single 0 → 1 transition allowed at inputs during the evaluation period!

Solution: domino logic (cont'd)

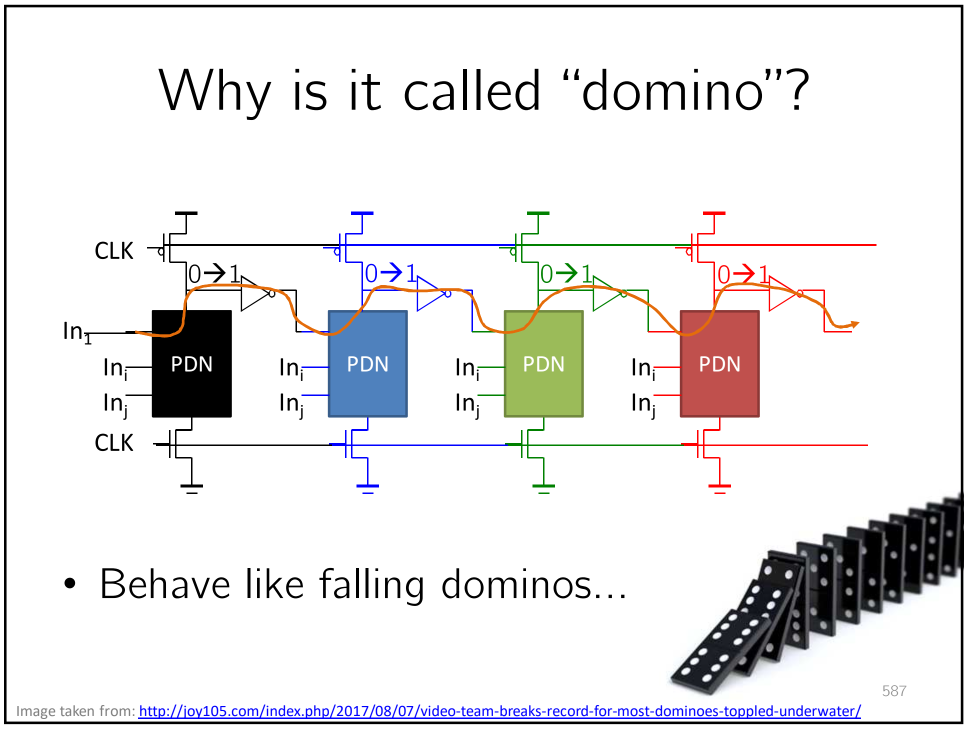


- Only possible transition is $0 \rightarrow 1$, which guarantees signal integrity

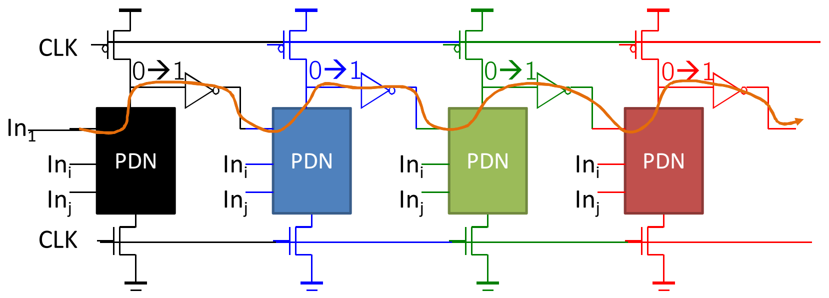
Advantages of static CMOS inverter



- Inverters can drive bleeder (=level keeper)
- Inverters can be optimized for fan-out
- Inverters improve noise immunity



Why is it called "domino"?



- Behave like falling dominos…

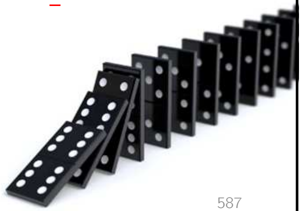
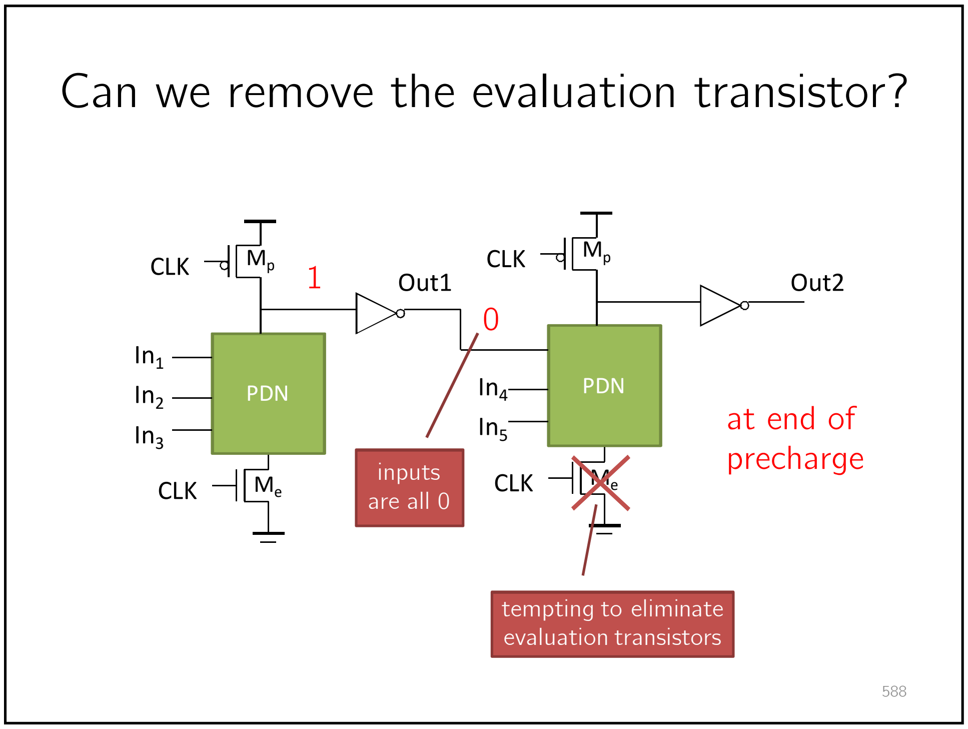
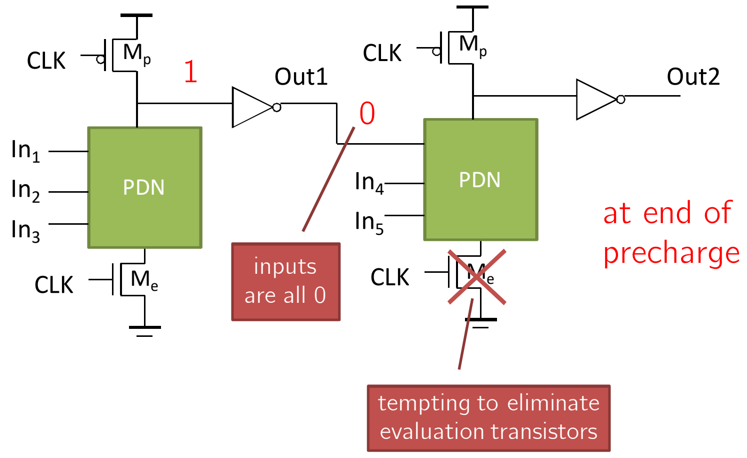


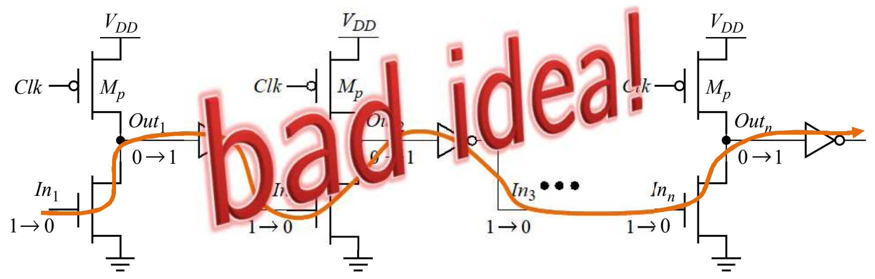
Image taken from: http://joy105.com/index.php/2017/08/07/video-team-breaks-record-for-most-dominoes-toppled-underwater/



Can we remove the evaluation transistor?



Footless domino



- Precharge has to ripple through: $t_{pre}=t_{prop}$
- Extra power dissipation when N+PMOS on

589

Advantages of domino logic

- Extremely fast circuits
  - Fan-in of a dynamic gate is much smaller than for a CMOS gate (only half of the transistors)
  - $t_{pHL}=0$
  - Static inverter can be optimized to match fan-out (separation of in and out capacitances)
- Inverters enable easy use of level restorers
- Fairly small area (at least N+4 transistors)

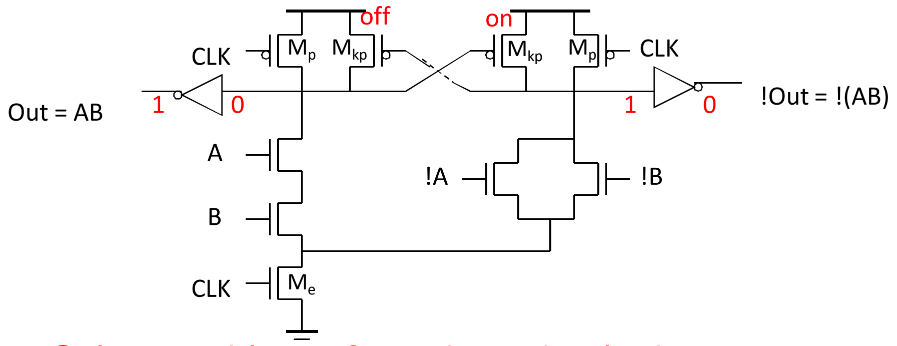
590

Main disadvantage

- Only non-inverting logic can be implemented
- Possible fixes:
  - Use De Morgan or other logic transforms
  - Use differential logic (dual rail)
  - Use np-CMOS (zipper or NORA)

591

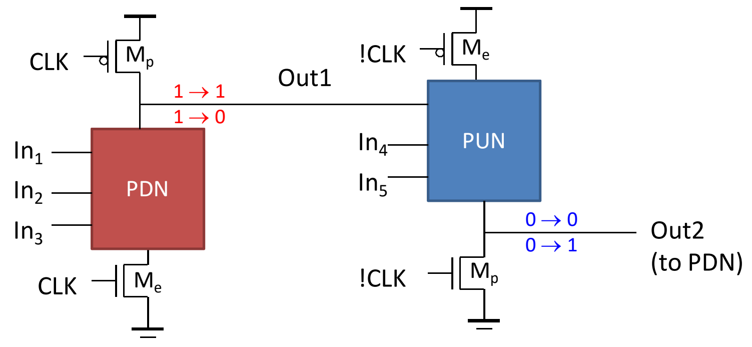
Differential (dual-rail) domino



- Solves problem of non-inverting logic
- High-performance → used in microprocessors
- Unratio'ed (even with cross-coupled PMOSs)
- High power consumption (always a transition)

592

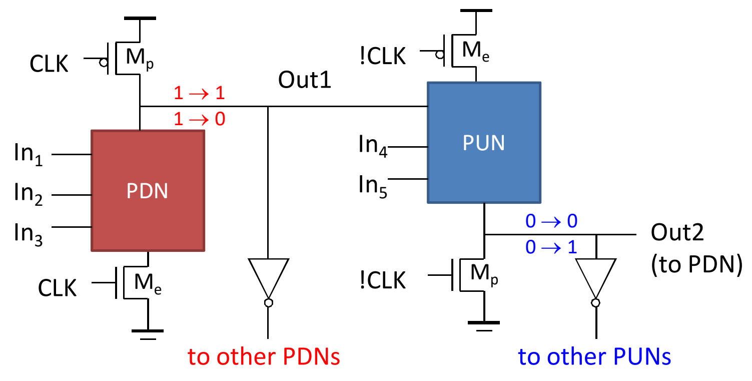
np-CMOS: zipper



- Only 0→1 transitions at inputs of PDN
- Only 1→0 transitions at inputs of PUN
- Advantage: Allows extremely dense logic

593

np-CMOS: NORA = no race

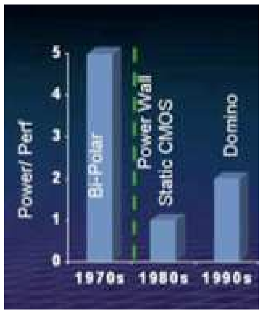


- Careful with sizing PUN to match PDN delay
- Reduced noise margins

594

Advantage of domino logic

- Allows much faster switching frequencies

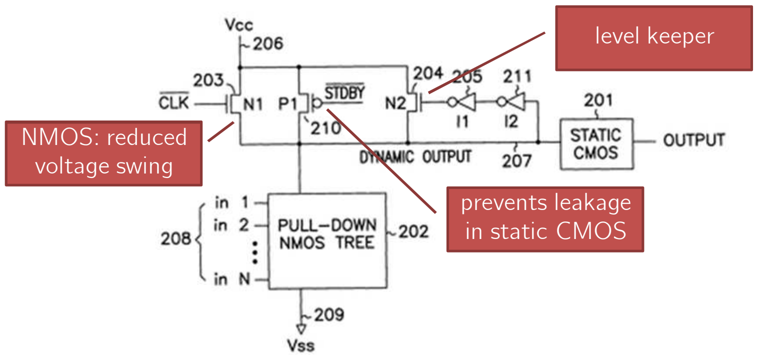


- Some of the fastest processors used domino logic gates in a full-custom-design process
- E.g.: Domino logic was used in 1990s by Intel

Source: Intel/http://www.anandtech.com/show/2594/12

Some tricks by Intel in 2003

- Intel NMOS precharge domino logic patent



United States Patent 6529045, 2003

Disadvantages

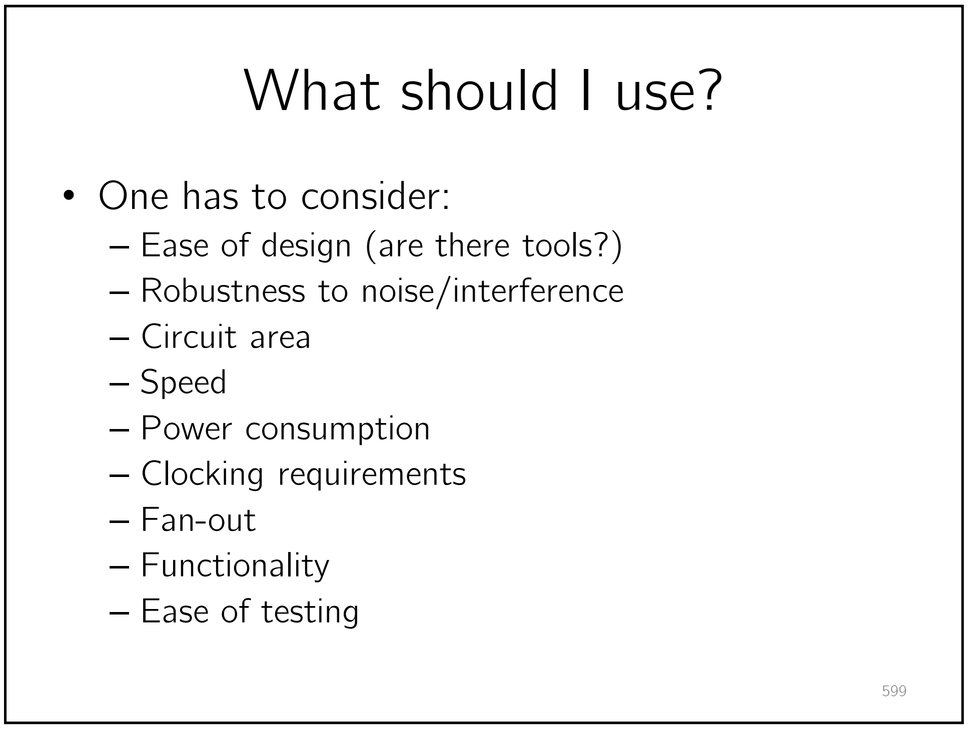
- Circuits are usually larger than static CMOS
- Requires full-custom design process
  - No good tool support available to the public
- Resolving timing/noise issues requires
  - large design teams
  - multiyear design cycles
  - hundreds of millions of dollars
- Higher power consumption that CMOS

597

The case for static CMOS

How to choose a logic style?

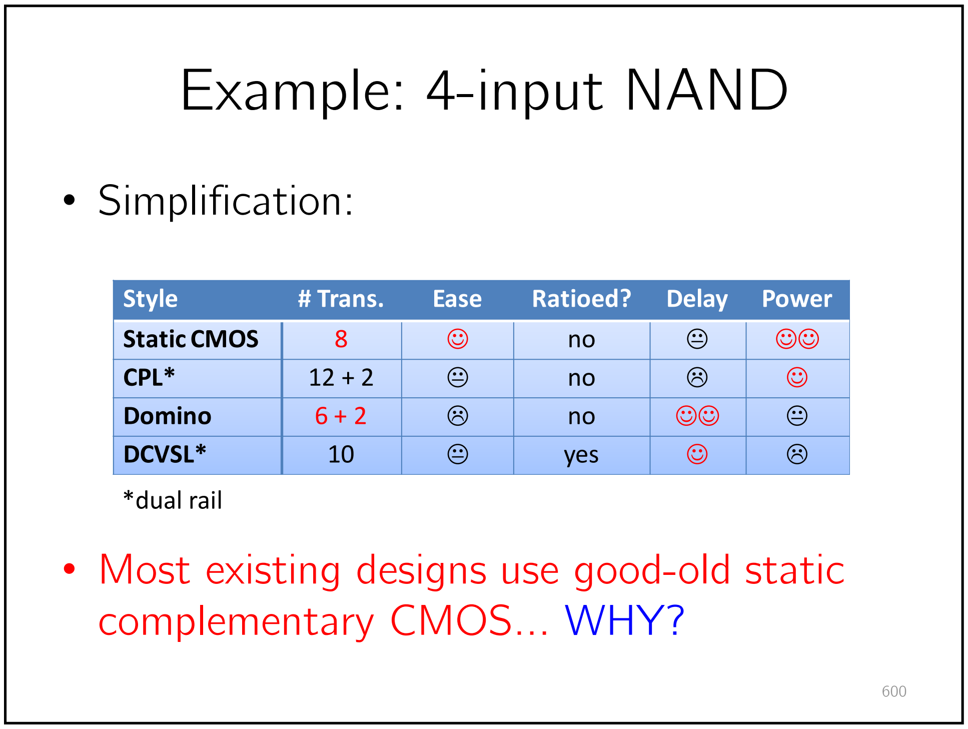
598



What should I use?

- One has to consider:
  - Ease of design (are there tools?)
  - Robustness to noise/interference
  - Circuit area
  - Speed
  - Power consumption
  - Clocking requirements
  - Fan-out
  - Functionality
  - Ease of testing

599



Example: 4-input NAND

- Simplification:

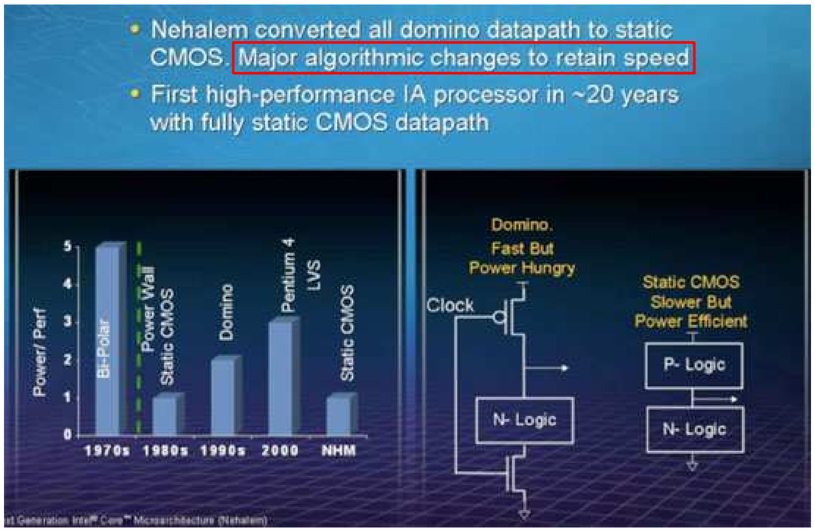
Style	# Trans.	Ease	Ratioed?	Delay	Power
Static CMOS	8	☺	no	😐	☺☺
CPL*	12 + 2	😐	no	☹	☺
Domino	6 + 2	☹	no	☺☺	😐
DCVSL*	10	😐	yes	☺	☹

*dual rail

- Most existing designs use good-old static complementary CMOS… WHY?

600

What is Intel doing?



- Nehalem converted all domino datapath to static CMOS. Major algorithmic changes to retain speed
- First high-performance IA processor in ~20 years with fully static CMOS datapath

Power/ Perf
5
4
3
2
1
0
Bi-Polar
Power Wall
Static CMOS
Domino
Pentium 4 LVS
Static CMOS
1970s
1980s
1990s
2000
NHM
Domino, Fast But Power Hungry
Clock
N- Logic
Static CMOS Slower But Power Efficient
P- Logic
N- Logic
Next Generation Intel® Core™ Microarchitecture (Nehalem)

Source: Intel/http://www.anandtech.com/show/2594/12

601

What are the reasons?

- The use of domino logic based circuits has dominated the design of microprocessors for the past twenty years
- Domino logic circuits exhibit smaller parasitic capacitances which allow higher switching frequencies; higher switching frequencies, however, result in high power
- To save power, Intel circuit designers decided to switch from domino logic to static CMOS based logic when implementing Nehalem
- CMOS based logic circuits consume substantially lower power than domino logic, but power savings are not free
- CMOS technology is slower with respect to switching frequency since it has much larger parasitic capacitances

Source: Intel

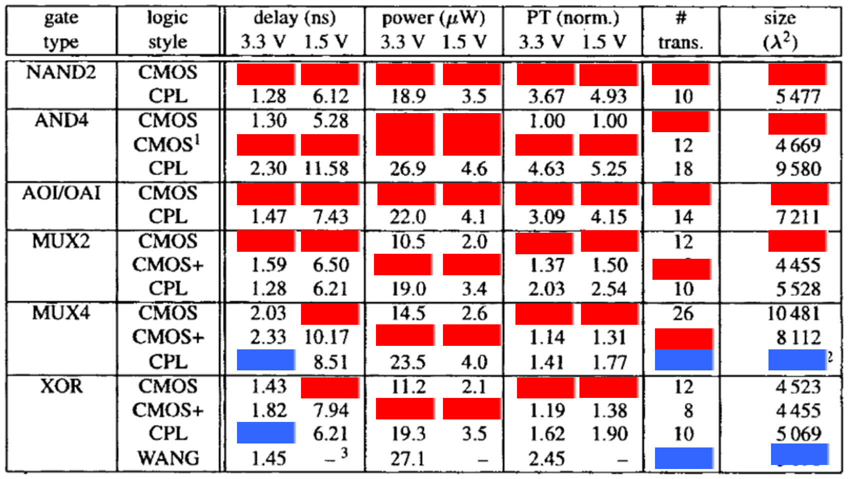
602

CPL is also energy efficient!

- It was common belief around 2000 that complementary pass-transistor logic (CPL) is an alternative in terms of power & area
- CPL is known to be very efficient for XOR and MUX circuits
  - Important in adder structures!
- But CMOS is the main choice nowadays…

603

CPL vs. CMOS



gate type	logic style	delay (ns) 3.3 V	delay (ns) 1.5 V	power (μW) 3.3 V	power (μW) 1.5 V	PT (norm.) 3.3 V	PT (norm.) 1.5 V	# trans.	size (λ^2)
NAND2	CMOS	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	CPL	1.28	6.12	18.9	3.5	3.67	4.93	10	5 477
AND4	CMOS	1.30	5.28	🟥	🟥	1.00	1.00	🟥	🟥
	CMOS[1]	🟥	🟥	🟥	🟥	🟥	🟥	12	4 669
	CPL	2.30	11.58	26.9	4.6	4.63	5.25	18	9 580
AOI/OAI	CMOS	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
	CPL	1.47	7.43	22.0	4.1	3.09	4.15	14	7 211
MUX2	CMOS	🟥	🟥	10.5	2.0	🟥	🟥	12	🟥
	CMOS+	1.59	6.50	🟥	🟥	1.37	1.50	🟥	4 455
	CPL	1.28	6.21	19.0	3.4	2.03	2.54	10	5 528
MUX4	CMOS	2.03	🟥	14.5	2.6	🟥	🟥	26	10 481
	CMOS+	2.33	10.17	🟥	🟥	1.14	1.31	🟥	8 112
	CPL	🟦	8.51	23.5	4.0	1.41	1.77	🟦	🟦[2]
XOR	CMOS	1.43	🟥	11.2	2.1	🟥	🟥	12	4 523
	CMOS+	1.82	7.94	🟥	🟥	1.19	1.38	8	4 455
	CPL	🟦	6.21	19.3	3.5	1.62	1.90	10	5 069
	WANG	1.45	–[3]	27.1	–	2.45	–	🟦	🟦

R. Zimmermann and W. Fichtner, "Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic," IEEE JSSC, Vol. 32, No. 7, July 1997

604

In many cases, CMOS is the choice

- Ease of design (are there tools?) ✓
- Robustness to noise/interference ✓
- Circuit area ✗
- Speed ✗
- Power consumption ✓
- Clocking requirements ✓
- Fan-out ✗
- Functionality ✓
- Ease of testing ✓

If you don't care about power, design time, etc. but you care about *area and speed* go for dynamic logic or DCVSL

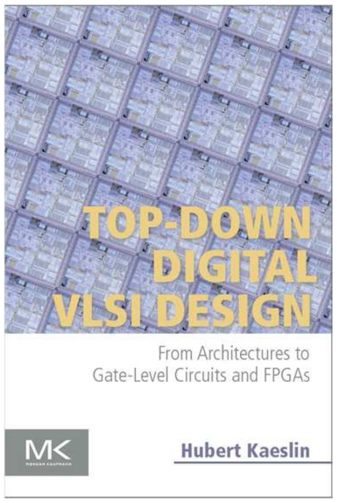
605

How to build state machines

Sequential logic

606

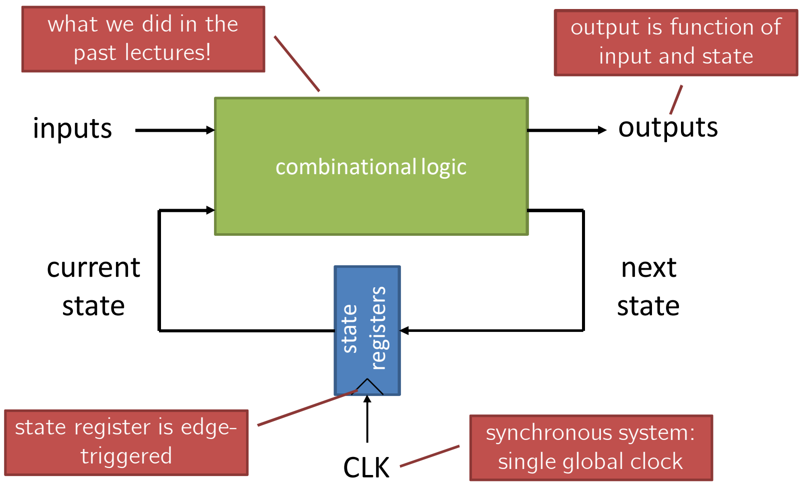
I will use parts of this book


TOP-DOWN DIGITAL VLSI DESIGN
From Architectures to Gate-Level Circuits and FPGAs
Hubert Kaeslin

- Follows a top-down approach:
  - starts with architectures and goes to transistors
  - Useful stuff first ☺
- Has excellent chapters on timing and architectures

607

Sequential logic


what we did in the past lectures!
output is function of input and state
inputs
outputs
combinational logic
current state
next state
state registers
state register is edge-triggered
CLK
synchronous system: single global clock

608

Some naming conventions

- Finite state machines (FSMs)
  - Mealy: output determined by input & state
  - Moore: output only depends on state
  - (Medvedev: output = current state)
- Latch = level sensitive
- Flip-flop = edge triggered
- Register = anything that stores temporary data locally and in small amounts

609

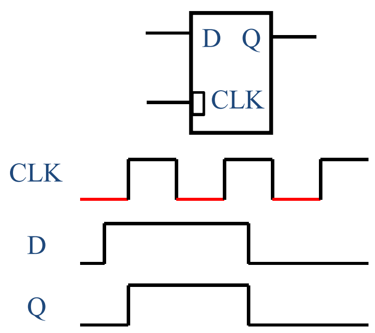
Static vs. dynamic storage

- Static storage (mostly latches & flip-flops)
  - Preserves state as long power is on
  - Positive feedback (regeneration) with internal connection between output and input
  - Useful when updates are infrequent (power!)
- Dynamic storage (e.g., DRAM)
  - Store state on (parasitic) capacitances
  - Only hold state for micro/milliseconds
  - Requires periodic refresh or data is lost
  - Usually smaller, faster, and lower power

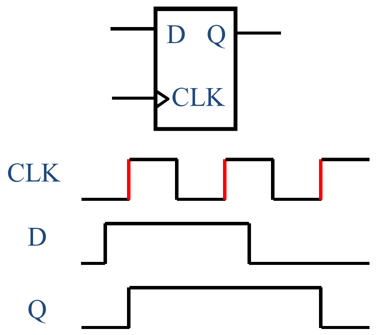
610

Latch vs. flip-flop

- Latch
  - Stores data when CLK signal is low



- Register/flip-flop
  - Stores data when CLK signal rises



611

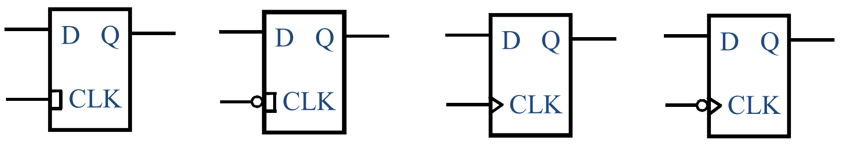
Latch vs. flip-flop (cont'd)

- Latch
  - Stores data when CLK signal is falling
  - Level sensitive
  - Transparent if CLK is high
  - Holds output if CLK is low
- Register/flip-flop
  - Stores data when CLK signal rises
  - Edge sensitive
  - Built using latches (e.g., master-slave flip-flops)

also opposite devices exist:
→ latch holds at high
→ flip-flop storing at falling

612

Common flip-flop and latch symbols



positive latch
1-transparent
0-hold

negative latch
0-transparent
1-hold

rising-edge
triggered FF

falling-edge
triggered FF

- Real-world flip-flops (and latches) may have additional inputs and outputs:
  - Reset in, enable in, scan in, and !Q out

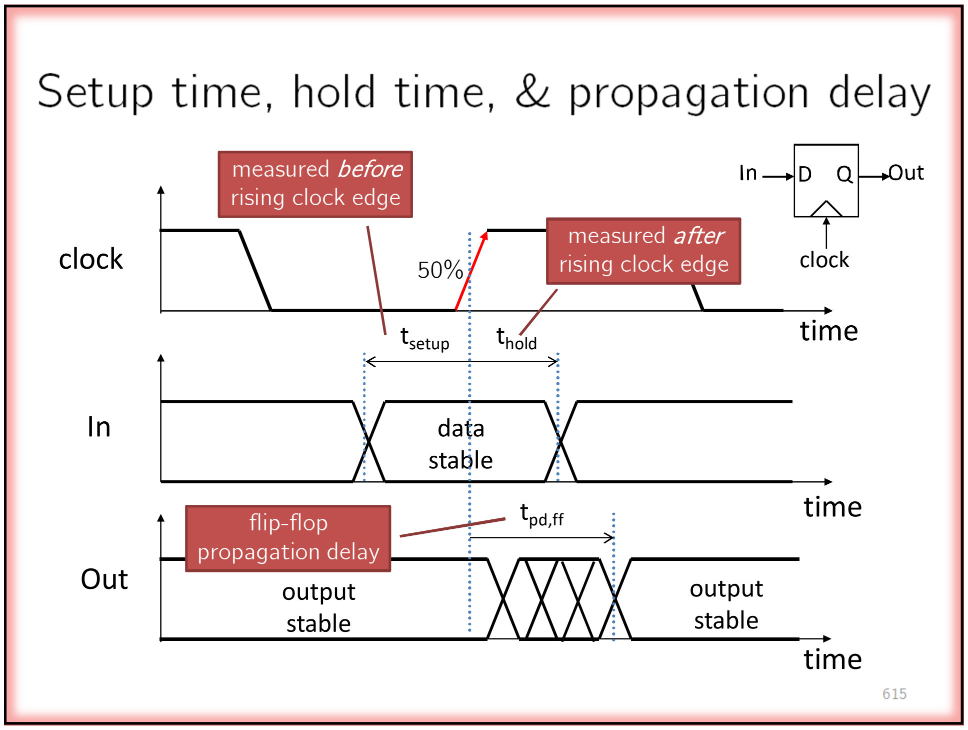
613

Extremely important

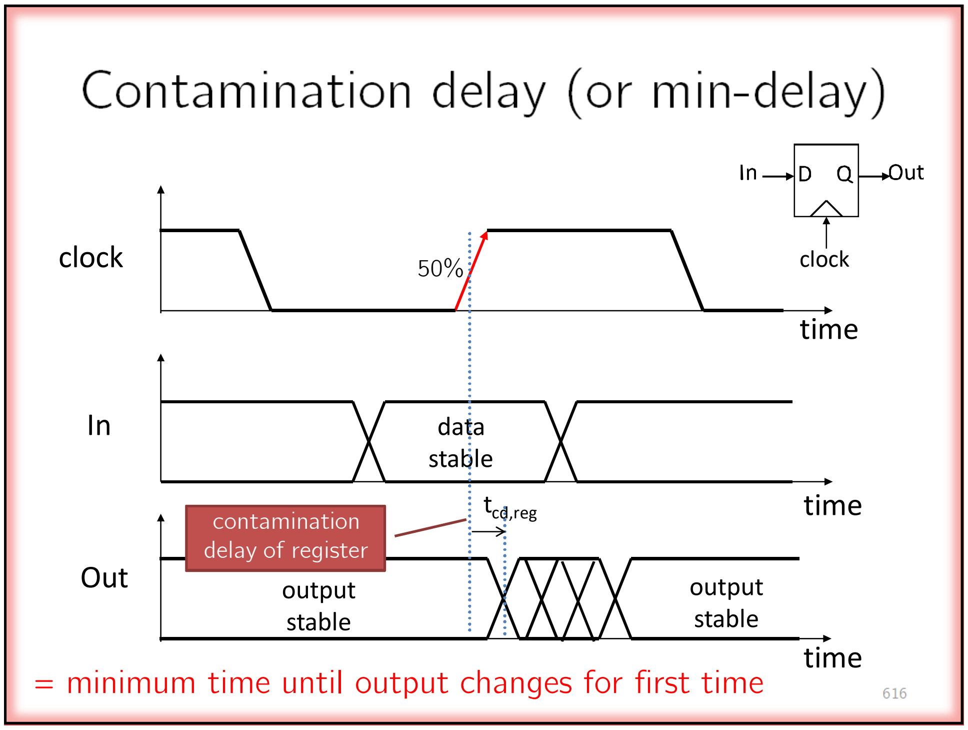
Timing!

Image taken from: http://disney.wikia.com/wiki/White_Rabbit%27s_Watch

614

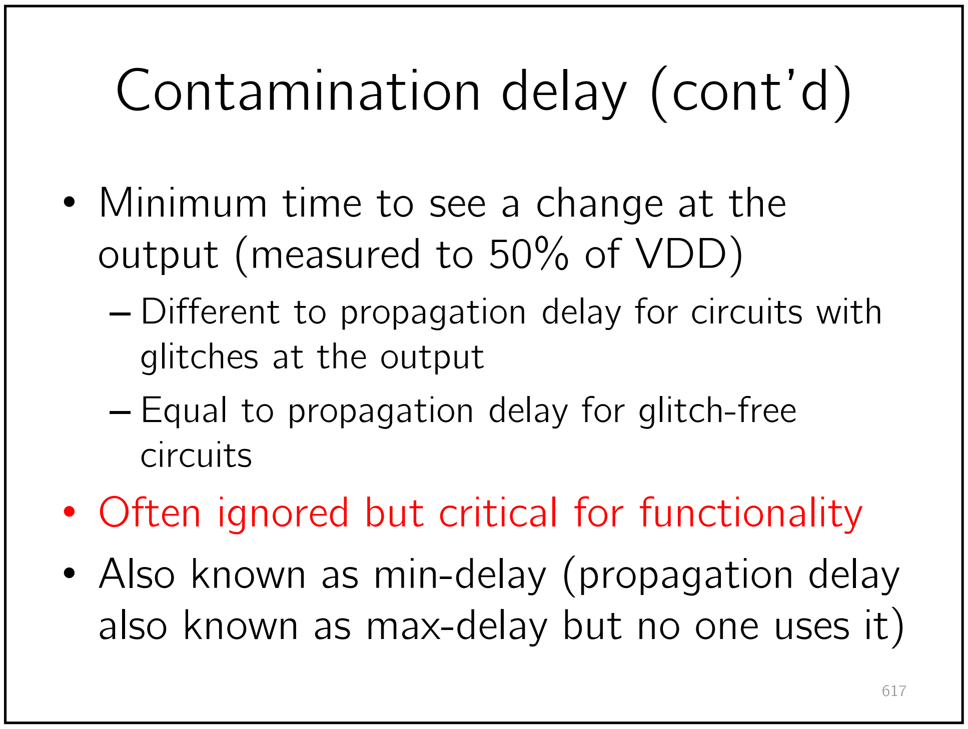


Setup time, hold time, & propagation delay



Contamination delay (or min-delay)

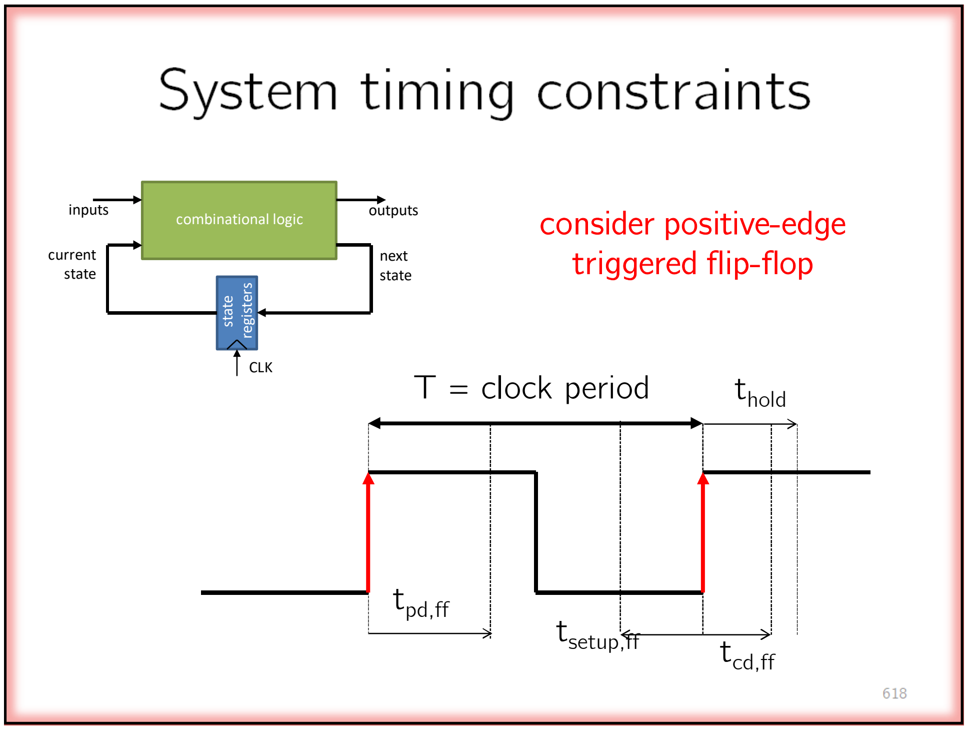
= minimum time until output changes for first time



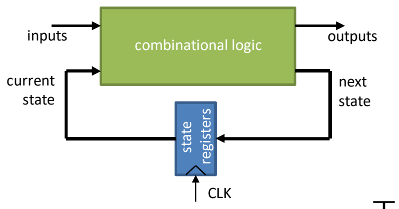
Contamination delay (cont'd)

- Minimum time to see a change at the output (measured to 50% of VDD)
  - Different to propagation delay for circuits with glitches at the output
  - Equal to propagation delay for glitch-free circuits
- Often ignored but critical for functionality
- Also known as min-delay (propagation delay also known as max-delay but no one uses it)

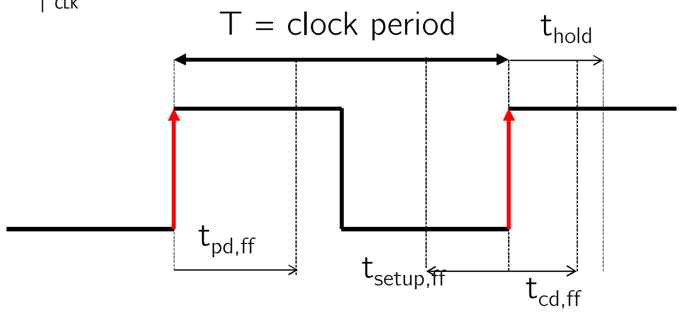
617



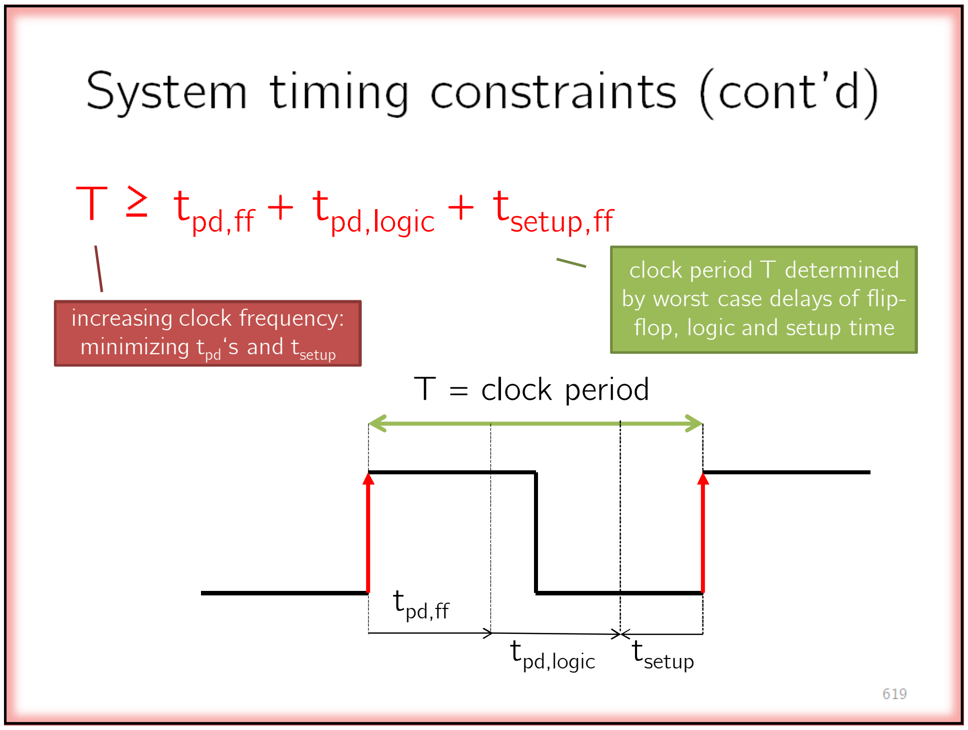
System timing constraints



consider positive-edge triggered flip-flop



618



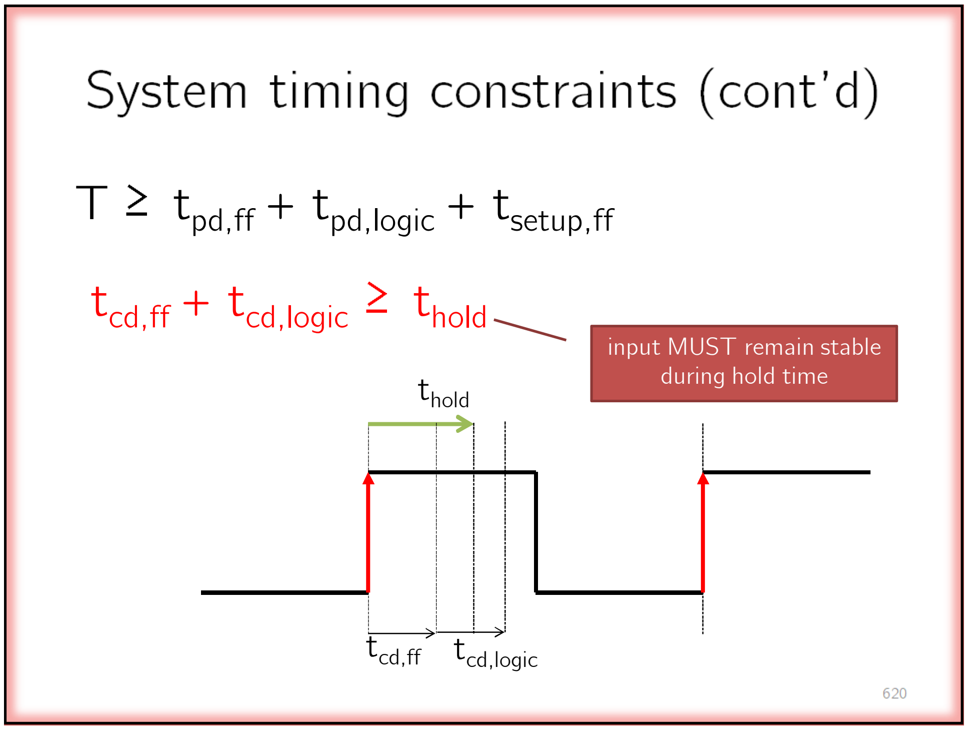
System timing constraints (cont'd)

$T \geq t_{pd,ff} + t_{pd,logic} + t_{setup,ff}$

increasing clock frequency: minimizing t_{pd}'s and t_{setup}

clock period T determined by worst case delays of flip-flop, logic and setup time

619



System timing constraints (cont'd)

$T \geq t_{pd,ff} + t_{pd,logic} + t_{setup,ff}$

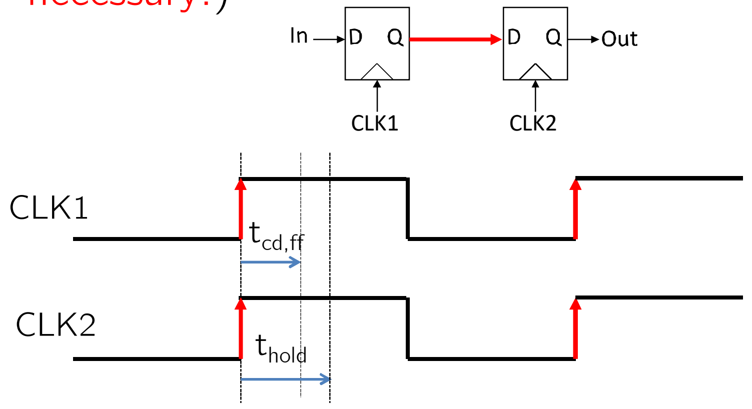
$t_{cd,ff} + t_{cd,logic} \geq t_{hold}$

input MUST remain stable during hold time

620

Careful* with hold condition

- Cascading flip-flops is dangerous (but often necessary!)

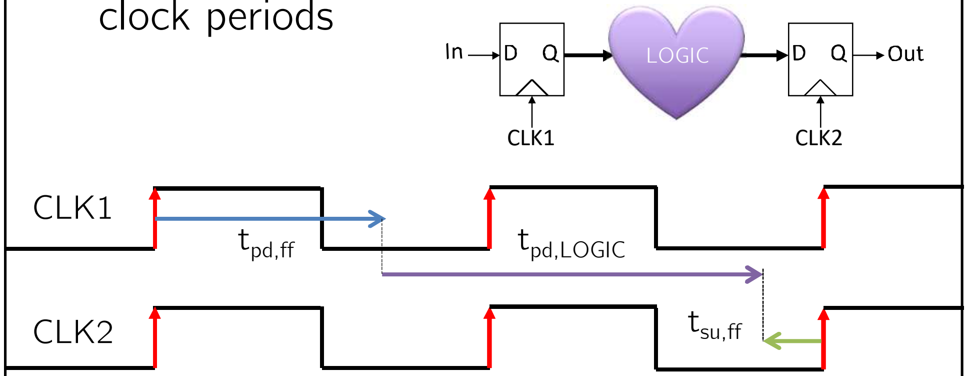


*and that's not even the full story

621

Even worse*

- Timing constraints can be for different clock periods



*may happen for very fast clock frequencies, very deep logic, and/or excessive clock skew

622