inst.eecs.berkeley.edu/~eecs151

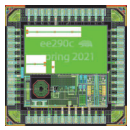
EECS151 : Introduction to Digital Design and ICs

Lecture 22 – Latches, Flip-Flops

Bora Nikolić

Berkeley engineering students pull off novel chip design in a single semester (Berkeley Engineer, June 17, 2021)

In what could have important implications for engineering education as well as the field of chip design, a class of Berkeley Engineering students has successfully completed the design process — or "tape-out" — for a novel chip that will be manufactured this summer. As part of this spring's Advanced Topics in Circuit Design course, 19 students with no prior experience in chip design went from basic introductions to tape-out by the end of a four-month period.



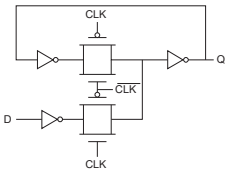
Review

- Binary division is a slow, iterative process
- Non-restoring division speeds it up
- SRT divider, higher radix, redundant number representation
- Timing analysis for early and late signal arrivals
- Flip-flop-based pipelines are a lot easier to analyze than latch-based ones
- Latches are based on positive feedback

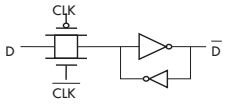
Latches

Writing into a Static Latch

Use the clock as a control signal (to break the positive feedback), that distinguishes between the transparent and opaque states



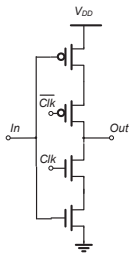
Converting into a MUX



Forcing the state (functionality depends on sizing)

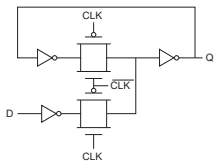
Tri-State Inverter

- Out is Z when Clk=0

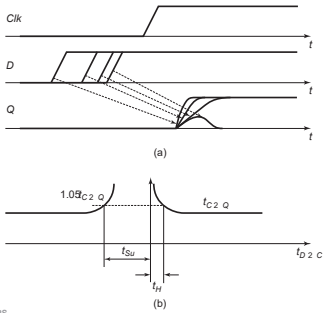


- Latch

Clk-Q Delay

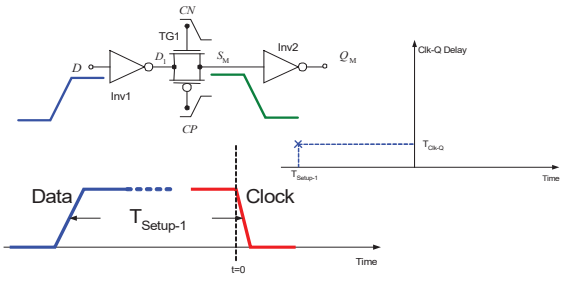


Setup and Hold Times



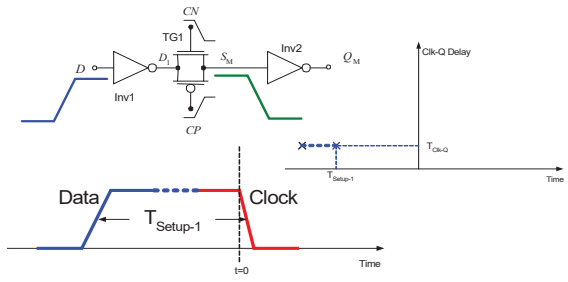
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)



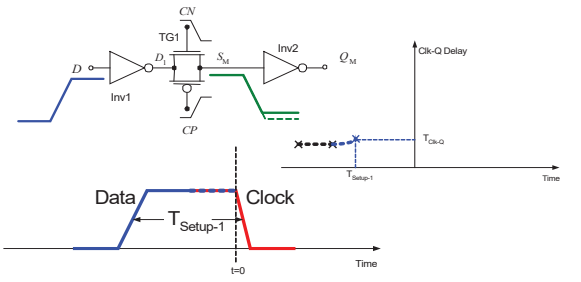
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)



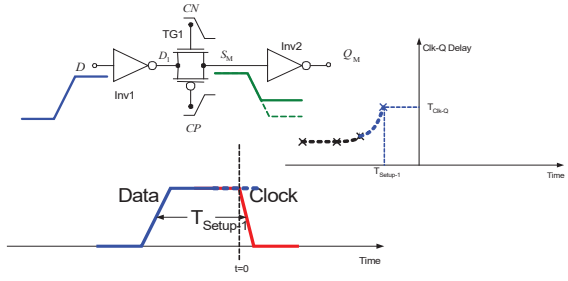
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)



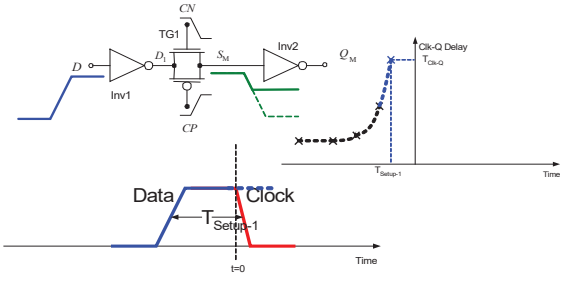
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)



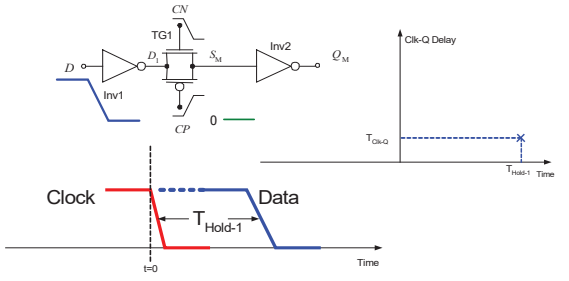
Setup-Hold Time Illustrations

Circuit before clock arrival (Setup-1 case)



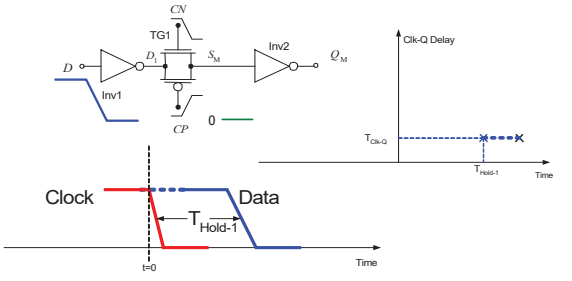
Setup-Hold Time Illustrations

Hold-1 case



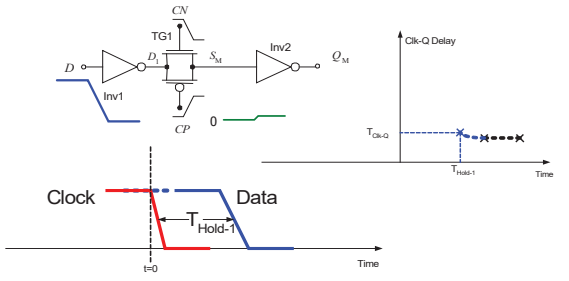
Setup-Hold Time Illustrations

Hold-1 case



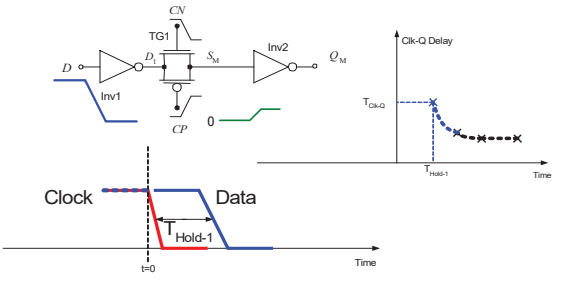
Setup-Hold Time Illustrations

Hold-1 case

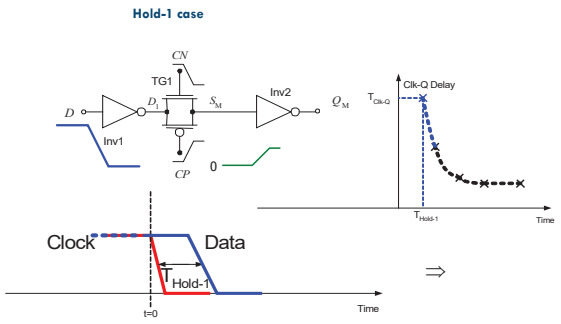


Setup-Hold Time Illustrations

Hold-1 case



Setup-Hold Time Illustrations

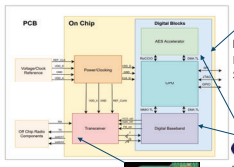


Administrivia

- Midterm 2 scores released
  - Final can clobber either midterm!
- Homework 9 posted on Friday, due 11/15
  - One more homework before Thanksgiving
- Project checkpoints #2 this week
- Thursday is a holiday (Veterans' Day)

The 'Tapeout' Class

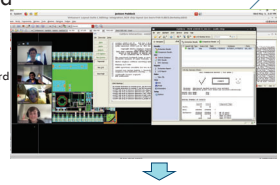
- EE194/290C, Spring'21 Pister, Nikolic, Niknejad

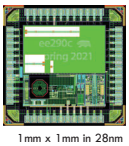


Berkeley Analog Generator (BAG) - generated SAR ADC

CHIPYARD
https://github.com/ucb-bar/chipyard
Processor core, interfaces
Software tools

CHISEL
https://www.chisel-lang.org/
Custom BLE digital baseband, accelerator wrappers





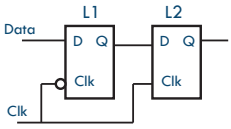
1mm x 1mm in 28nm

- **Spring'22 class will use Intel 16**
  - **2mm x 2mm**

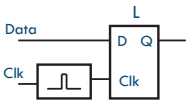
Flip-Flops

Types of Flip-Flops

Latch Pair
(Master-Slave)

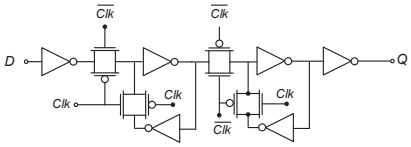


Pulse-Triggered Latch



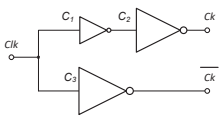
Transmission Gate Flip-Flop

- Two back-to-back latches

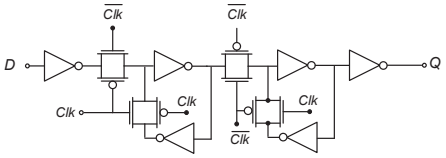


Aside: Inverter Fork

- Often found in flip-flops: equalize Ck, Ckb delays
  - Logical effort = ?

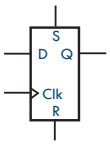


Clk-Q, Setup and Hold Times



Set, Reset

- Set and reset can be synchronous or asynchronous
  - Always watch for additional timing paths!



D-flip-flop with synchronous reset

```
module dff_sync_clear(
  input d, r, clk,
  output reg q);

  always @(posedge clk)
    begin
      if (!r) q <= 1'b0;
      else q <= d;
    end
endmodule
```

`always` block entered only at each positive clock edge

D-flip-flop with asynchronous reset

```
module dff_async_clear(
  input d, r, clk,
  output reg q);

  always @(negedge r or posedge clk)
    begin
      if (!r) q <= 1'b0;
      else q <= d;
    end
endmodule
```

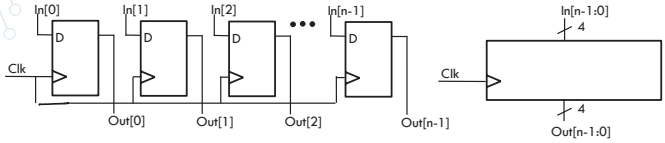
`always` block entered immediately when (active-low) r is asserted

Flip-Flop Timing Characterization

- Combinational logic delay is a function of output load and input slope
- Sequential timing (flip-flop):
  - $t_{clk\text{-}q}$ is function of output load and clock rise time
  - t_{Su}, t_H are functions of D and Clk rise/fall times

Registers, Register files

- Register is often built out of flip-flops



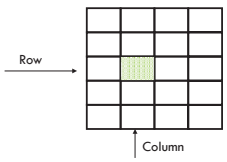
- Register file can be built out of registers
  - Ok for small register files
  - Large register files are generally built with latches and custom designed (like memory arrays)

SRAM

Random Access Memory Architecture

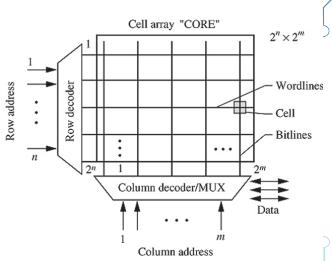
- Conceptual: Linear array of addresses
  - Each box holds some data
  - Not practical to physically realize – millions of 32b/64b words

- Create a 2-D array
  - Decode Row and Column address to get data

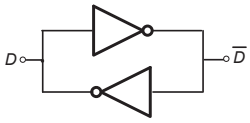


Basic Memory Array

- Core
  - Wordlines to access rows
  - Bitlines to access columns
  - Data multiplexed onto columns
- Decoders
  - Addresses are binary
  - Row/column MUXes are 'one-hot' - only one is active at a time



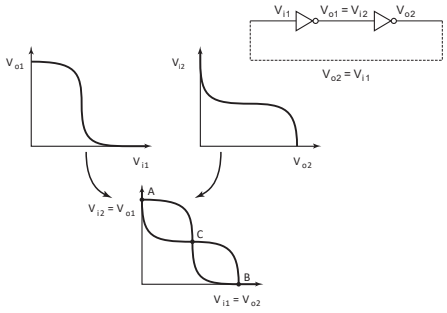
Basic Static Memory Element



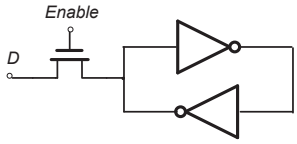
- If D is high, $\overline{D}$ will be driven low
  - Which makes D stay high
- Positive feedback
- Same principle as in latches

Positive Feedback: Bi-Stability

- As in latches

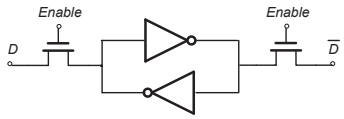


Writing into a Cross-Coupled Pair



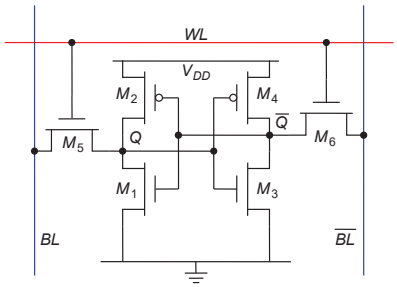
- This is a 5T SRAM cell
  - Access transistor must be able to overpower the feedback; therefore must be large
  - Easier to write a 0, harder to write 1
- Can implement as a transmission gate as well; single-ended 6T cell
- There is a better solution…

SRAM Cell



Since it is easier to write a 0 through NMOS, write only 0s, but on opposite sides! When reading, measure the difference

6-transistor CMOS SRAM Cell



- Wordline (WL) enables read/write access for a row
- Data is written/read differentially through shared BL, $\overline{BL}$

Review

- Latches are based on positive feedback
- Clk-Q delay calculated similarly to combinational logic
- Setup, hold defined as D-Clk times that correspond to Clk-Q delay increases
- Flip-flop is typically a latch pair
- Dense memories are built as arrays of memory elements
  - SRAM is a static memory