**Harold’s Flip-Flops**

**Cheat Sheet**

13 June 2020

|  |
| --- |
| **S-R Flip-Flop (Edge-Triggered)** |
| **Style** | NAND-NAND | AND-NOR |
| **Circuit** |  | Flip-flop (electronics) - Wikipedia |
| **Symbol** |  |
| **Truth Table** |

|  |  |  |  |
| --- | --- | --- | --- |
| **S** | **R** | **Qnext** | **Action** |
| 0 | 0 | Q | No change, Hold |
| 0 | 1 | 0 | Reset (Q 🡪 0) |
| 1 | 0 | 1 | Set (Q 🡪 1) |
| 1 | 1 | X | Invalid, Not allowed |

 |
| **Boolean Equation** | $$Q\_{next}=\overbar{R}Q\_{prev}+\overbar{R}S=\overbar{R}(Q\_{prev}+S) $$ |
| **Name Origin** | SR for Set-Reset |
| **Observations** | A **Flip-Flop** is a **Latch** with 2 AND/NAND gates added for clock input to trigger data flow from left to right |
| **Applications** | * Storing a single bit of data, 1 or 0
 |
| **TTL Chips** | 74x71, 74Lx74 |
| **D Flip-Flop (Edge-Triggered)** |
| **Style** | NAND-NAND | AND-NOR |
| **Circuit** | Schematic diagram |  |
| **Symbol** | Digital Logic Part 4 - Data SignalsRheingold Heavy  |
| **Truth Table** |

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Outputs** | **Action** |
| **D** | **CLK** | **Qnext** | **Q’next** |
| 0 | ↑ | 0 | 1 | Reset (Q 🡪 0) |
| 1 | ↑ | 1 | 0 | Set (Q 🡪 1) |

 |
| **Boolean Equation** | $$Q\_{next}=D$$ |
| **Name Origin** | D for Delays, since it delays the signal until the next active clock transition |
| **Observations** | * Made with S-R flip-flop with input S inverted for input R
* Stores a single bit after the edge-triggered clock pulse
 |
| **Applications** | * Storing Bits (memory) in a pipeline
* Event Detection
 |
| **TTL Chips** | 74x74, 74x79, 74x171, 74x173 |



|  |
| --- |
| **J-K Flip-Flop (Edge-Triggered)** |
| **Style** | NAND-NAND | AND-NOR |
| **Circuit** | J K Flip Flop Explained in Detail - DCAClab Blog | JK & T Flip-Flops | Room 514 |
| **Symbol** | Introduction to Flip-Flops - luisdanielhernandezengineeringportfolio JK-Flip Flop Evolution |
| **Truth Table** |

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Outputs** |  |
| **J** | **K** | **CLK** | **Qnext** | **Q’next** | **Action** |
| 0 | 0 | ↓ | Q | Q’ | Hold, No change |
| 0 | 1 | ↓ | 0 | 1 | Reset (Q 🡪 0) |
| 1 | 0 | ↓ | 1 | 0 | Set (Q 🡪 1) |
| 1 | 1 | ↓ | Q’ | Q | Toggle, Change (1 🡪 0) |

 |
| **Boolean Equation** | $$Q\_{next}=J\overbar{Q} \_{prev}+\overbar{K}Q\_{prev}$$ |
| **Name Origin** | None, other than J and K are adjacent letters in the alphabet |
| **Observations** | * Same as S-R flip-flop except 2 feedback lines added
* Fixes the invalid 1-1 state
 |
| **Applications** | * Frequency Division: If $J=K=HIGH,$ then clock frequency divider $\left(^{f}/\_{2}\right)$
* Counting: If cascaded with QA wired to JKB CLK, then QA = LSB and QB=MSB
* Sequence Detection: If cascaded with QA🡪JB and Q’A🡪KB, then tap Q/Q’s for 1/0 pattern, then AND for output
 |
| **TTL Chips** | 74x68, 74x69, 74x70, 74x73, 74x76, 74x101, 74x102, 74X103, 74x107 |
| **T Flip-Flop (Edge-Triggered)** |
| **Style** | NAND-NAND | AND-NOR |
| **Circuit** | Designing of T Flip Flop | T flip-flop - CircuitVerse |
| **Symbol** |  Designing of T Flip Flop |
| **Truth Table** |

|  |  |  |
| --- | --- | --- |
| **Inputs** | **Outputs** | **Action** |
| **T** | **CLK** | **Qnext** | **Q’next** |
| 0 | ↑ | Q | Q’ | No change, Hold |
| 1 | ↑ | Q’ | Q | Change, Toggle |

 |
| **Boolean Equation** | $$Q\_{next}=\overbar{Q}\_{prev}$$ |
| **Name Origin** | T for Toggle, since it changes state on the triggering edge of the clock pulse |
| **Observations** | * Made with J-K flip-flop with input T connected to both J and K
* Implements the two middle rows of the J-K flip-flop truth table
 |
| **Applications** | * Frequency Division: If $J=K=HIGH,$ then clock frequency divider $\left(^{f}/\_{2}\right)$
 |
| **TTL Chips** | 74x374 or use J-K flip-flop chips |

|  |
| --- |
| **J-K Flip-Flop Applications** |
| **Frequency Division** | If $J=K=HIGH,$ then clock frequency divider $\left(^{f}/\_{2}\right)$ |
|  |
| **Counting** | If cascaded with QA wired to JKB CLK, then QA = LSB and QB=MSB |
|  |

|  |  |
| --- | --- |
| **Sequence Detection** | If cascaded with QA🡪JB and Q’A🡪KB, then tap Q/Q’s for 1/0 pattern, then AND for output |
|  |

Credit: Diagrams taken from “ECPI University EET 230 – Digital Systems II”, Wikipedia, and Google images.