# Opcode

# OPCODE[0]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | 0 |

Decide source of Register A:

|  |  |
| --- | --- |
| **Bit value** | **Source** |
| 0 | input |
| 1 | Register Y |

# OPCODE[3:1]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | 0 |

Enable group (bit per group)

|  |  |
| --- | --- |
| **Bit value** | **Group** |
| 00**1** | ADD/SUB |
| 0**1**0 | LOGIC |
| **1**00 | SHIFT |

# OPCODE[5:4]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | 0 |

Group function

|  |  |  |
| --- | --- | --- |
| **Group** | **Bit value** | **Function** |
| ADD/SUB | 00 | ADD |
| 01 | SUB |
| LOGIC | 00 | XOR |
| 01 | OR |
| 10 | AND |
| SHIFT | 00 | SHIFT4 |
| 01 | SHIFT1 |
| 10 | SHIFT2 |
| 11 | SHIFT3 |

# TOP

Diagram

Description automatically generated with medium confidence

|  |  |
| --- | --- |
| **Logic gate** | **count** |
| INV | 0 |
| AND2 | 0 |
| AND3 | 0 |
| AND4 | 0 |
| OR2 | 0 |
| OR3 | 4 |
| OR4 | 0 |
| XOR2 | 0 |
| XOR3 | 0 |
| DFF | 0 |
| MUX2:1 | 4 |
| MUX3:1 | 0 |
| MUX4:1 | 0 |

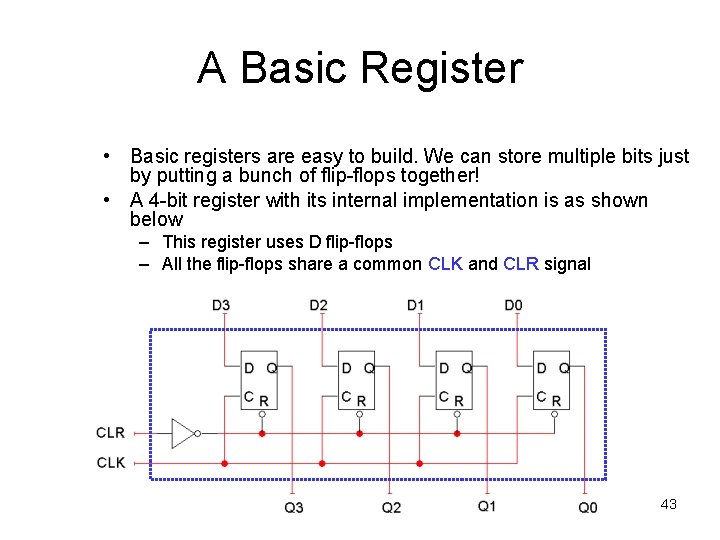
The way we achieve blocking of operations is by an **enable** bit for each logic unit. This is done by placing a switch (nmos transistor) that will block vdd for that block, and a pmos that will pull down the output to 0. This is described in figure below.

Diagram, schematic

Description automatically generated

# Registers A, B and Y

Since not specified otherwise and it is not really needed, we will remove the clear logic.



|  |  |
| --- | --- |
| **Logic gate** | **count** |
| INV | 0 |
| AND2 | 0 |
| AND3 | 0 |
| AND4 | 0 |
| OR2 | 0 |
| OR3 | 0 |
| OR4 | 0 |
| XOR2 | 0 |
| XOR3 | 0 |
| DFF | 4 |
| MUX2:1 | 0 |
| MUX3:1 | 0 |
| MUX4:1 | 0 |

**add\_sub**

s: ‘0’ to ADD, ‘1’ to SUB (this takes care of converting B to 2’s complement)

Diagram

Description automatically generated

Inside the **4-bit Adder** block above (carry look ahead adder, based on HAYES p.54)

NOTE: since not specified otherwise, we remove the carry-out logic.

|  |  |
| --- | --- |
| **Logic gate** | **count** |
| INV | 0 |
| AND2 | 7 |
| AND3 | 2 |
| AND4 | 1 |
| OR2 | 5 |
| OR3 | 1 |
| OR4 | 1 |
| XOR2 | 4 |
| XOR3 | 4 |
| DFF | 0 |
| MUX2:1 | 0 |
| MUX3:1 | 0 |
| MUX4:1 | 0 |

Diagram

Description automatically generated

# and\_or\_xor

|  |  |
| --- | --- |
| **OP[1:0]** | **Function** |
| 00 | XOR |
| 01 | OR |
| 10 | AND |

Diagram, engineering drawing

Description automatically generated

|  |  |
| --- | --- |
| **Logic gate** | **count** |
| INV | 0 |
| AND2 | 4 |
| AND3 | 0 |
| AND4 | 0 |
| OR2 | 4 |
| OR3 | 0 |
| OR4 | 0 |
| XOR2 | 4 |
| XOR3 | 0 |
| DFF | 0 |
| MUX2:1 | 0 |
| MUX3:1 | 4 |
| MUX4:1 | 0 |

# barrel\_shift

NOTE: we need to shift by 1,2,3 or 4. Shifting by 4 is equal to shifting by 0Diagram

Description automatically generated

|  |  |
| --- | --- |
| **Logic gate** | **count** |
| INV | 0 |
| AND2 | 0 |
| AND3 | 0 |
| AND4 | 0 |
| OR2 | 0 |
| OR3 | 0 |
| OR4 | 0 |
| XOR2 | 0 |
| XOR3 | 0 |
| DFF | 0 |
| MUX2:1 | 0 |
| MUX3:1 | 0 |
| MUX4:1 | 4 |

# Floor plan

* Registers are close together to minimize delays between them
* Square ALU allows minimizing delays between all blocks

A picture containing shape

Description automatically generated

**Logic gates that will be used**

A picture containing schematic

Description automatically generated

**Total logic gates count:**

Graphical user interface, application, table, Excel

Description automatically generated

Estimated ALU4 area: 1334 um^2 = 1.33 mm^2