

Class 6: Intro to SRC Simulator Register Transfers and Logic Circuits

SRC Language Conventions

- Chapter 2, Appendix B.5

SRC Simulator Demo

- cond_br.asm (SRC)
- fib_simple.asm (SRC)
 - Iterative fibonacci using only registers
- array accesses (pseudo-Sparc)

cond_br.asm

```
#define Cost 125
if(X<0)
    X = -X;

// Example program
// +6 in register -> false
// - 6 in register -> true
```

fib_simple.asm (C style source)

```
register int current_result = 0; // r1
register int fib_n_1 = 1; // r2 - fib(n-1)
register int fib_n_2 = 0; // r3 - fib(n-2)
register int count = 5; // r4 - loop ctr

while(count != 0)
{
    current_result = fib_n_1 + fib_n_2;
    fib_n_2 = fib_n_1;
    fib_n_1 = current_result;
}
```

Arrays in C

```
int a[5];
int main()
{
    int i=0;
    for(i=0; i<5; i++)
        a[i]=i;
    return 0;
}
```

Arrays in Assembly

```
// Array access example (pseudo-Sparc assembly code)

mov 0, r2      ; i in r2
mov a, r3      ; r3 = address of a
Labell:        ; body of loop
sll r2, 2, r4  ; mult by 4)
               ; r4==i*4, offset calculation
st r2, [r3+r4] ; store i into a + 4*i
add r2, 1, r2  ; add 1 to i
cmp r2, 4      ; compare i to 4
               ; loop will iterate 5 times
ble Labell    ; branch less than equal to
               ; Labell
stop
```

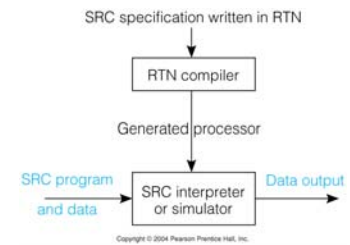
Register Transfer Notation

- Formal description language
 - *Metalinguage*
 - Language used to describe languages
 - Used to describe SRC machine
 - Mathematical notation

Why are Specification Languages Useful?

- **Allows:**
 - Description of **what** without having to specify **how**.
 - **Precise and unambiguous** specifications, unlike natural language.
- **Reduces errors:**
 - Due to **misinterpretation of imprecise specifications** written in natural language
 - Due to **confusion in design and implementation** - "human error."
- **Specifications can be automatically checked and processed by tools.**
 - An RTN specification could be input to a simulator generator that would **produce a simulator** for the specified machine.
 - An RTN specification could be input to a compiler generator that would **generate a compiler** for the language, whose output could be run on the simulator.

Relationship of RTN to SRC



Road Map

How do data transfer operations map to logic circuits?



Assembly language
Architecture
Gate-level logic

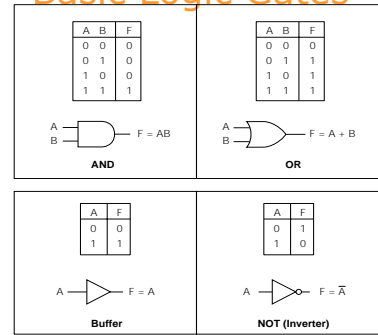
DLD Review

Appendix A

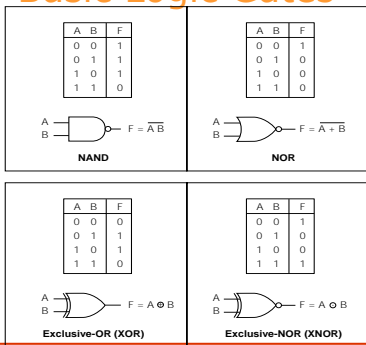
Definitions

- **Combinational logic**
 - a digital logic circuit in which *logical decisions* are made based only on *combinations* of the inputs. e.g. an adder.
- **Sequential logic**
 - a circuit in which decisions are made based on combinations of the current inputs as well as the past history of inputs. e.g. a memory unit.
- **Finite state machine**
 - a circuit which has an *internal state*, and whose outputs are functions of both current inputs and its internal state. e.g. a vending machine controller.

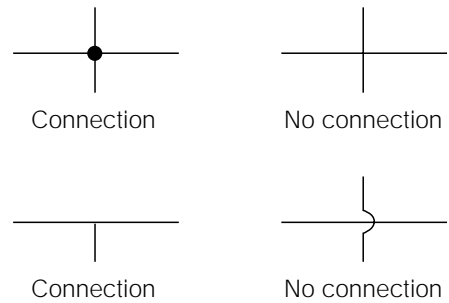
Basic Logic Gates



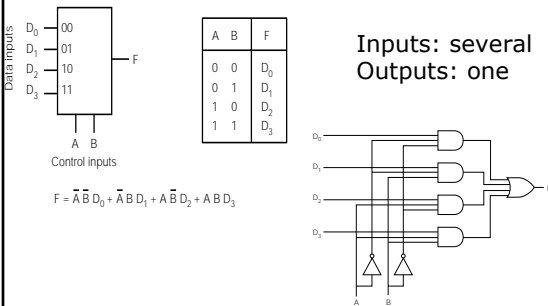
Basic Logic Gates



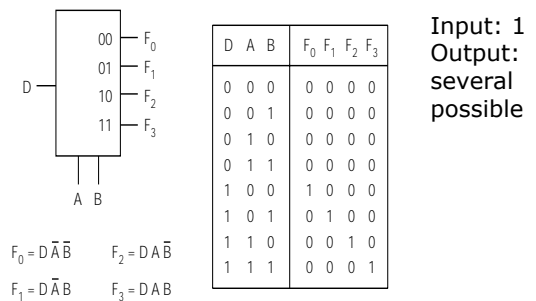
Circuit Intersections



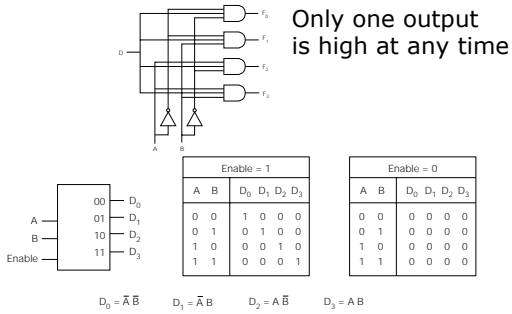
Multiplexer (MUX)



Demultiplexer (DEMUX)

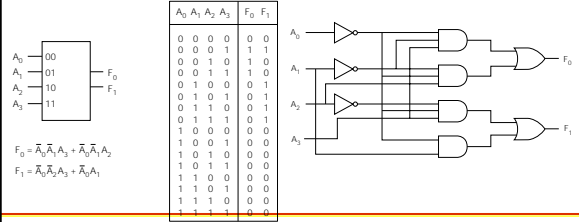


Decoder

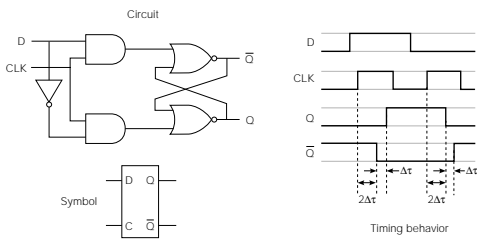


Priority Encoder

- Translates set of inputs into binary encoding
- Used to arbitrate among devices competing for same resource

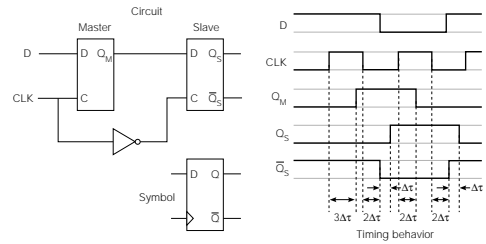


Clocked D (Data) Flip-Flop



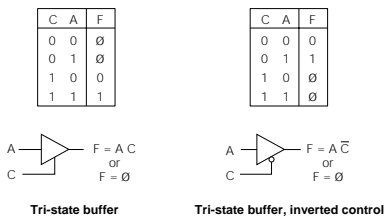
Sometimes called a latch, has a potential problem: If D changes while the clock is high, the output will also change. The Master-Slave flip-flop solves this problem:

The Master-Slave Flip-Flop



The rising edge of the clock clocks new data into the Master, while the slave holds previous data. The falling edge clocks the new Master data into the Slave.

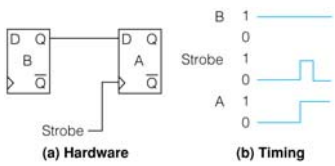
Tri-state Buffers



- There is a third state: High impedance. This means the gate output is essentially disconnected from the circuit.
- This state is indicated by Ø in the figure.

Implementing Register Transfers in Logic

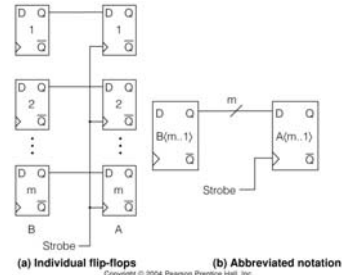
One-bit Register Transfer



Data moves to A when *Strobe* is activated

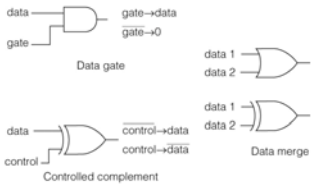
Multiple Bit Register Transfer

- Implementing $A(m..1) \leftarrow B(m..1)$



Data Transmission View of Logic Gates

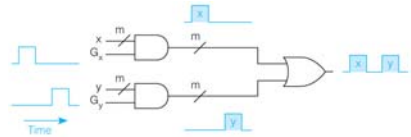
- Logic gates can be used to control the transmission of data:



Copyright © 2004 Pearson Prentice Hall, Inc.

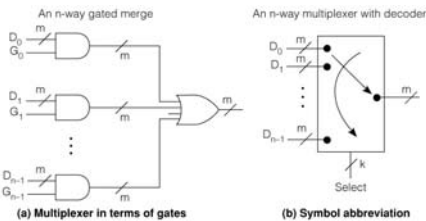
Multiplexer as a 2 Way Gated Merge

- Data from multiple sources can be selected for transmission



Copyright © 2004 Pearson Prentice Hall, Inc.

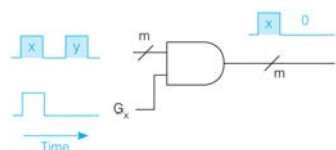
m-bit Multiplexer and Symbol



Copyright © 2004 Pearson Prentice Hall, Inc.

- Multiplexer gate signals G_i may be produced by a binary to one-out-of-n decoder

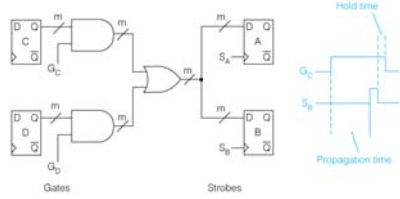
Separating Merged Data



Copyright © 2004 Pearson Prentice Hall, Inc.

- Merged data can be separated by gating at the right time
- It can also be strobed into a flip-flop when valid

Multiplexed Register Transfers using Gates and Strobes



Copyright © 2004 Pearson Prentice Hall, Inc.

- Selected gate and strobe determine which Register is Transferred to where.
- $A \leftarrow C$, and $B \leftarrow C$ can occur together, but not $A \leftarrow C$, and $B \leftarrow D$