Registers

Register transfer

Control function (MOV) : Destination register (R2) <- Source (R1)

Micro-ops

Register transfer is an example of a **Micro-Operation**

- A micro-operation is an operation which can be accomplished within a small number of gate propagation delays upon data stored in adjacent registers and memory.
- Those commonly encountered in digital systems divide naturally into four groups
  - Transfer or identity mico-ops copy data e.g
  - Arithmetic micro-ops provide the elements of arithmetic e.g R0<-R1+R2
  - logic micro-ops provide per bit operation e.g R<-R2 or R2
  - Shift micro-ops provide bit rotations e.g R1<- sr R2, R0 <- rol R1
- These are operations which can be accomplished with a full-adder, which with **carry lookahead** logic can be made to deliver a substantial result e.g 64-bit in just a few gate delays

Arithmetic Micro-ops

Symbolic CLA Inputs Function
micro-op A B C S
R0<-R1+R2 R1 + R2 + 0 Addition
R0<-R1-R2 R1 + R2 + 1 Subtraction
R0<-R1+1 R1 + 0…0 + 1 Increment
R0<-R1-1 R1 + 1…1 + 0 Decrement
R0<-R2 0…0 + R2 + 0 1’s Complement
R0<-R2 0…0 + R2 + 1 2’s Complement\

The first two of these operations may be accomplished by the addition of an XOR gate to the B-input of each full-adder.

Overflow

When a fixed sized register is used for arithmetic operations there is the hazard at each micro-op of overflow. If the Register R stores an n-bit 2's compliment number then:

$-2^{n-1} \leq R \leq 2^{n-1} - 1$

We can detect overflow for all the previous operations by simply recording the status of bits C = carry and V = overflow

$K_1 : C \Leftarrow C_n, V \Leftarrow C_n \bigoplus C_{n-1}$

This is then the basis for the adder-subtractor on the next page which uses control input X to select addition and X for subtraction.

$X.K_1 : R1 \Leftarrow R1 + R2$

$X.K_1 : R1 \Leftarrow R1 + R2 + 1$

$K_1 : C \Leftarrow C_n, V \Leftarrow C_n \bigoplus C_{n-1}$

Logic Micro-Operations

The aim here is to provide an effective set of bit-wise functions. A typical basic set follows:

Symbolic Description\

$R0 \Leftarrow R1$ Logical bitwise NOT (1’s compliment)
$R0 \leftarrow R1 \bigwedge R2$ Logical bitwise AND (clears bits)
$R0 \leftarrow R1 \bigvee R2$ Logical bitwise OR (sets bits)

$R0 \leftarrow R1 \bigoplus R2$ Logical bitwise XOR (complements bits)

George Bode Prof. of mathematics in UCC introduced the notation $\bigwedge$ and $\bigvee$ in 1854.

Shift Micro-operations

These provide lateral bitwise shift which are essential for many basic arithmetic algorithms e.g multiplication, division, square root...
$R \leftarrow srR \equiv R_i \Leftarrow R_{i+1} i = 0, n-2, R_{n-1} \Leftarrow 0$
$R \leftarrow slR \equiv R_i \leftarrow R_i - 1 i = 1, n-1, R_0 \leftarrow 0$

From these logical shifts different variants can be developed to handle the end bits differently (Arithmetic shift, rotates, LSL, LSR....)

Two different path choices are available

- Multiplexer-based transfer for speed
- Bus-based transfers for flexibility and economy

The if-then else control structure, when applied to identity micro-ops, results in the destination register requiring selective access to two different source registers.
$K : R0 \leftarrow R1, K_1 K_2 : R0 \leftarrow R2$

To route two sources to one destination we can use a 2:1 MUX with control input S, data input $D0 \bigvee D1$, and R0 must have a control $LOAD_{R0}$. From the RT description, we deduce the following functions for these control inputs

$$LOAD_R0 = K_1 + K_1 K_2 = K_1 + K_2$$
$$S = K_1 \rightarrow D1 to R1 \bigvee D0 = R2$$

Bus-based transfers

Digital systems typically have a considerable number of registers N.

- Typically 8 < N < 256
- Programmer need to be able to make transfers between any pair of them.
- Consider the N=1 and use 2:1 mux to interconnect R0, R1, R2.

Register transfer via MUX

This is a very flexible system for it can make up to three independent transfers in one clock period.

$R2 \leftarrow R1$ 1 X X 1 0 0 Point-to-Point
$R2 \leftarrow R1, R1 \leftarrow R2$ 1 1 X 1 1 0 Reg. Exchange
$R2 \leftarrow R1, R1 \leftarrow R0$ 1 0 0 1 1 1 Reg. Rotate
$R2 \leftarrow R0, , R1 \leftarrow R0$ 0 0 X 1 1 0 Reg. broadcast

This is very costly in terms of interconnections requiring 6*n MUX input connections. (General is N*n-bit registers will require (N-1)*N*n wires)

Register Transfer via MUX & Bus

To reduce the amount of interconnects we can use 3-1 MUXs with a single MUX-to-Register bus connection. This results in some loss of flexibility.

$R0 \leftarrow R2$ 1 0 0 0 1 Point-to-Point $R0 \leftarrow R1, R2 \leftarrow R1$ 0 1 1 0 1 Reg. Broadcast $R0 \leftarrow R1, R1 \leftarrow R0$ IMPOSSIBLE

MUX input connections have been reduced from 6*n to 3*n. For N Registers we need only N*n wires.

Tri-state Bus

Tri-state buffers provide the means to construct a wired-or of arbitrary fan-in, with which we can effectibely disperse the MUX on the previous slide right back to the register latches. That is we build our 3:1 MUX as:

This results in a solution seen below (tri-state) which has the same functionality as the MUX based solution, using just a bi-directional bus

$R0 \leftarrow R2$ 1 0 0 0 0 1 Point-to-Point $R0 \leftarrow R1, R2 \leftarrow R1$ 0 1 0 1 0 1 Reg. Broadcast $R0 \leftarrow R1, R1 \leftarrow R0$ IMPOSSIBLE Single source only

With this arrangement it is possible to connect N*n-bit registers with N-1 paths of width n.

Memory Transfers

- Typically processor memories are addressed by a number of address registers:
  - **PC, MAR, IOR**
- Address register information is transmitted over an address bus to memory M.
- Similarly data is transferred to/from a number of data reigsters over a data bus.
  - **Data register: IR, DR, BUF**

DataPath

- In all digital systems we can partition the structure into two sections:
  1. The data path which performs data-processing operations on the data stream
  2. The control unit which determines the schedule for these data processing operations.

Register File and Functional Unit

- In practice the three functional micro-ops are implemented in one compact circuit, the **Function Unit** composed of the **ALU** and the **Shifter**.
- Data for this **Functional Unit** comes from a physically adjacent **Register file**, with dual MUX-busses able to supply two operands per clock cycle.
- Since the register file itself is modest in size, there must be provision to send and receive data to/from the main memory system via **DATA IN** and **DATA OUT**.