04 - Basic Arithmetic Blocks and Operations

- Adders
  - Half adder, Full adder, Word adder
  - Carry, ripple carry, look-ahead carry
- Subtraction
  - Using adders; carry logic
- Multiply
  - Parallel and serial multiply
- Division
  - Concept; implementation
- ALU
  - Accumulator, status flags

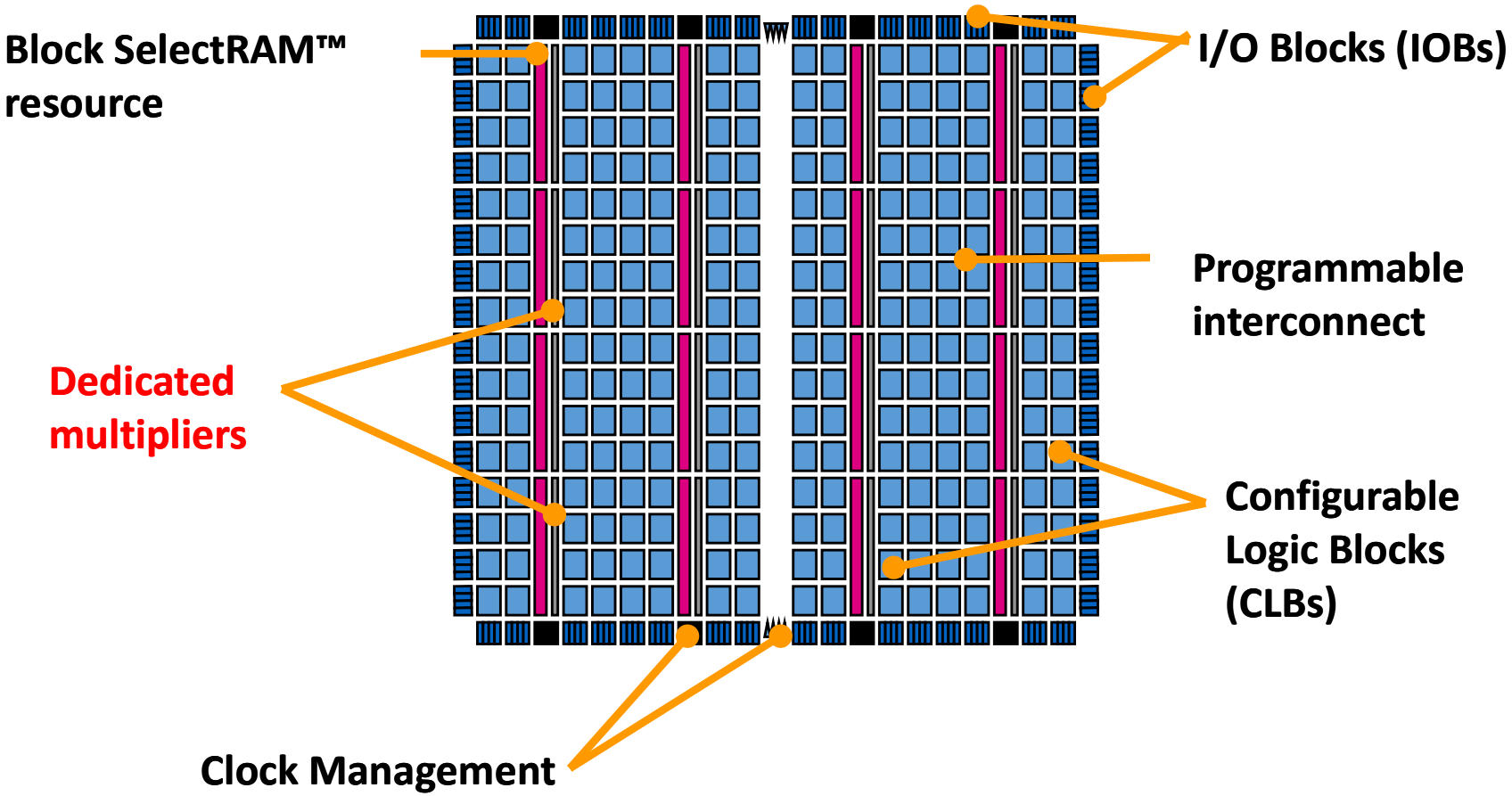
VHDL code for an 8x8 combinational multiplier

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity mul_8x8 is
   port ( A: in  unsigned ( 7 downto 0);
          B: in  unsigned ( 7 downto 0);
          P: out unsigned (15 downto 0));
end mul8x8;

architecture arch of mul8x8 is
begin
     P <= A * B;
end arch;
```

Xilinx FPGAs Architecture

- All Xilinx FPGAs contain the same basic resources
  - Logic Resources
    - Slices (grouped into CLBs)
      - Contain combinatorial logic and register resources
    - Memory
    - Multipliers
  - Interconnect Resources
    - Programmable interconnect
    - IOBs
      - Interface between the FPGA and the outside world
  - Other resources
    - Global clock buffers
    - Boundary scan logic

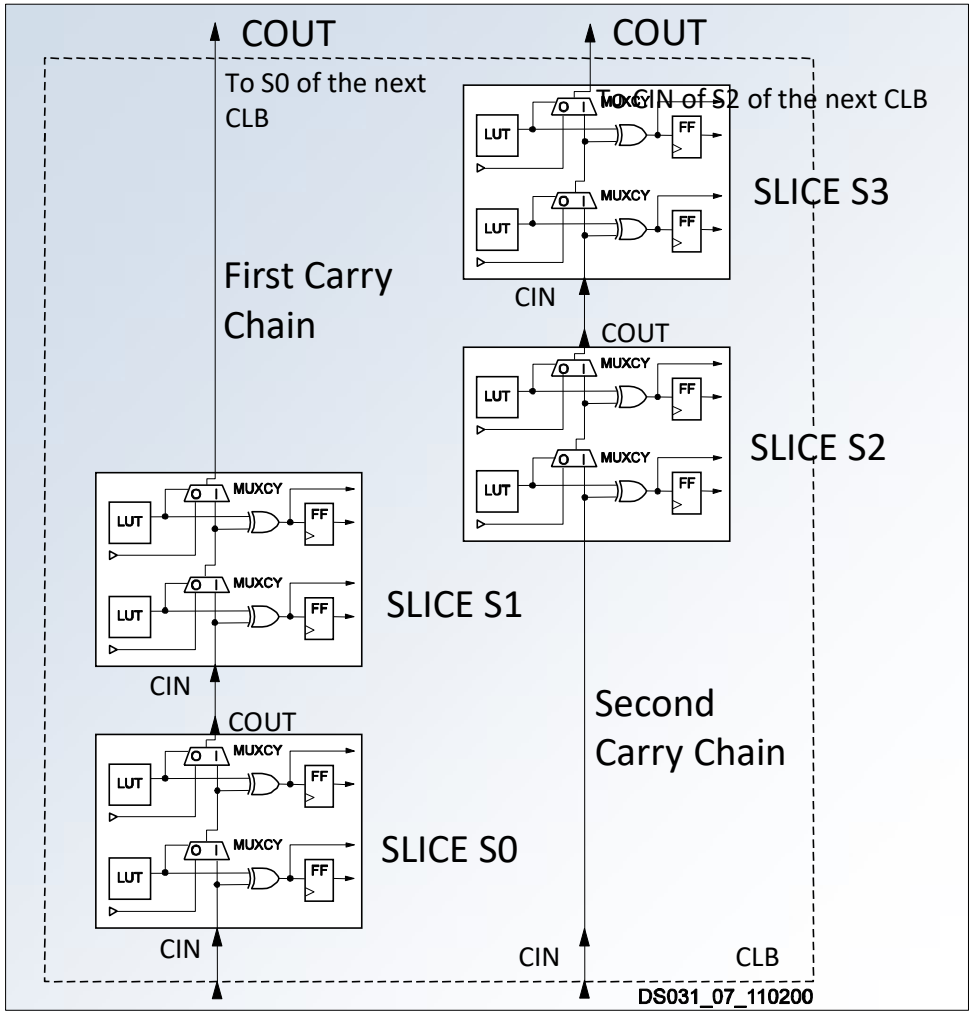
Virtex-II Archtecture



Virtex™–II architecture's core voltage operates at 1.5V

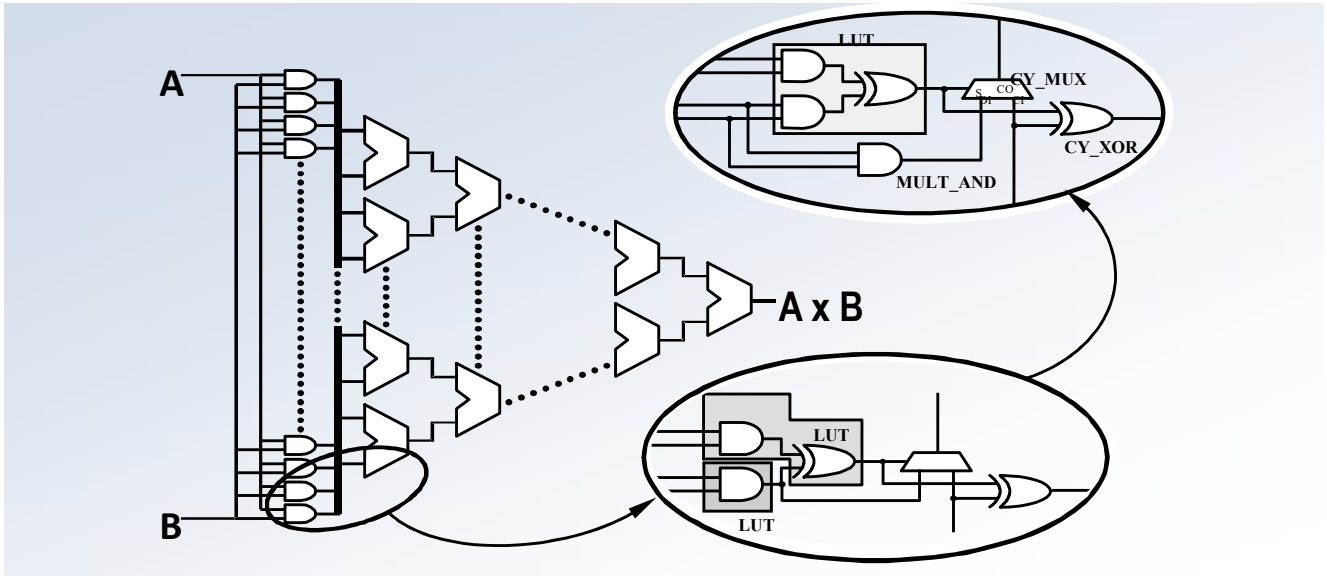
Carry Chains

- Simple, fast, and complete arithmetic Logic
  - Dedicated XOR gate for single-level sum completion
  - Uses dedicated routing resources
  - All synthesis tools can infer carry logic



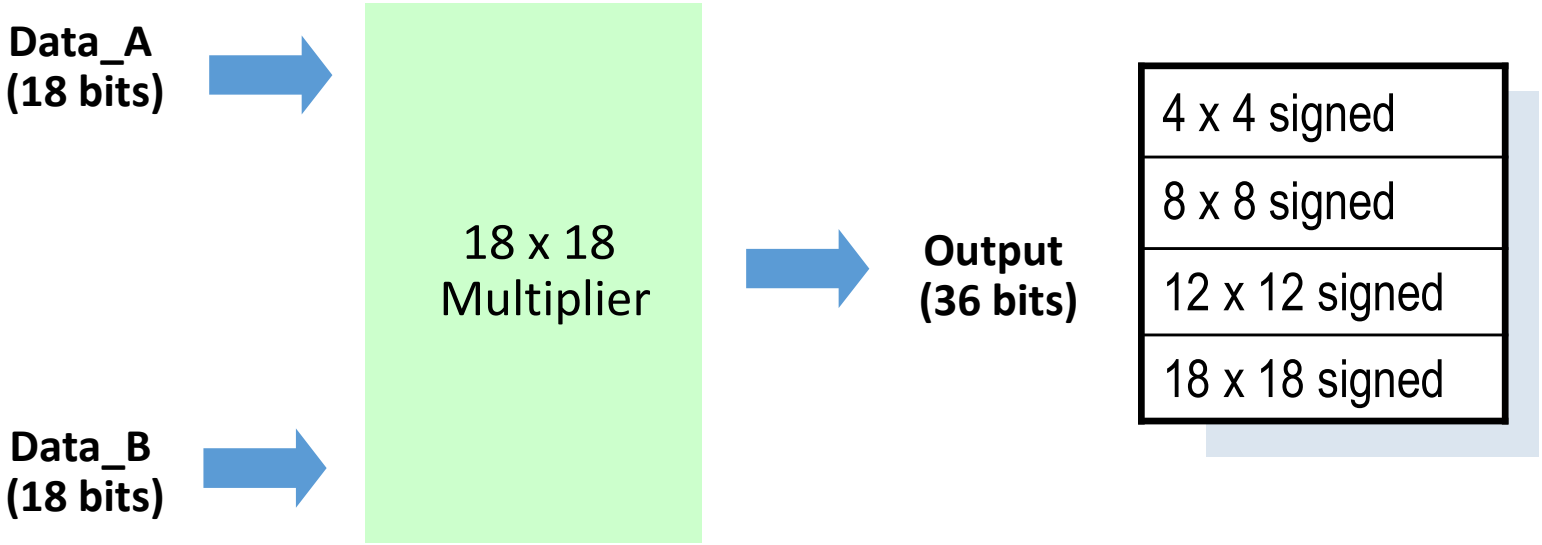
Multiplier AND Gate

- Highly efficient multiply and add implementation
  - Earlier FPGA architectures require two LUTs per bit to perform the multiplication and addition
  - The MULT_AND gate enables an area reduction by performing the *multiply* and the *add* in one LUT per bit



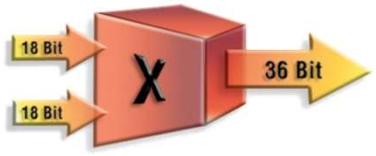
Embedded Multiplier Blocks

- 18-bit twos complement signed operation
- Optimized to implement Multiply and Accumulate functions
- Multipliers are physically located next to block SelectRAM™ memory

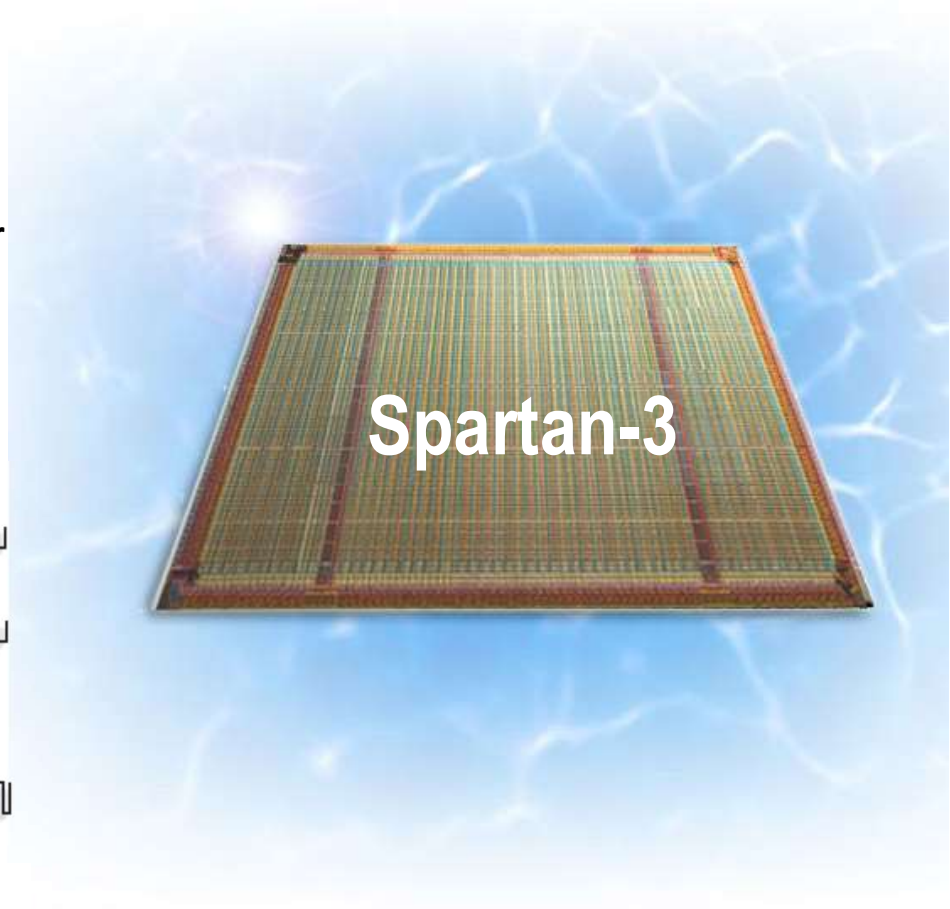


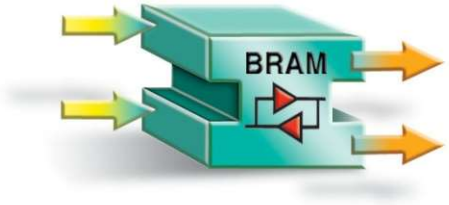
The Spartan-3 Family

Built for high volume, low-cost applications



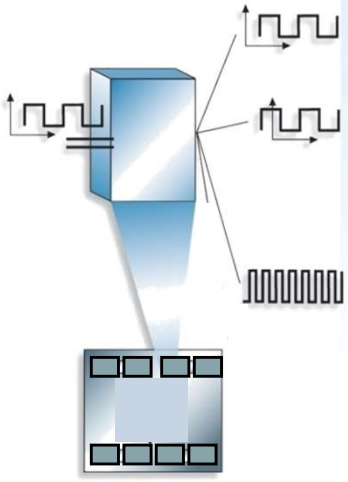
18x18 bit Embedded Pipelined Multipliers for efficient DSP





Configurable 18K Block RAMs + Distributed RAM

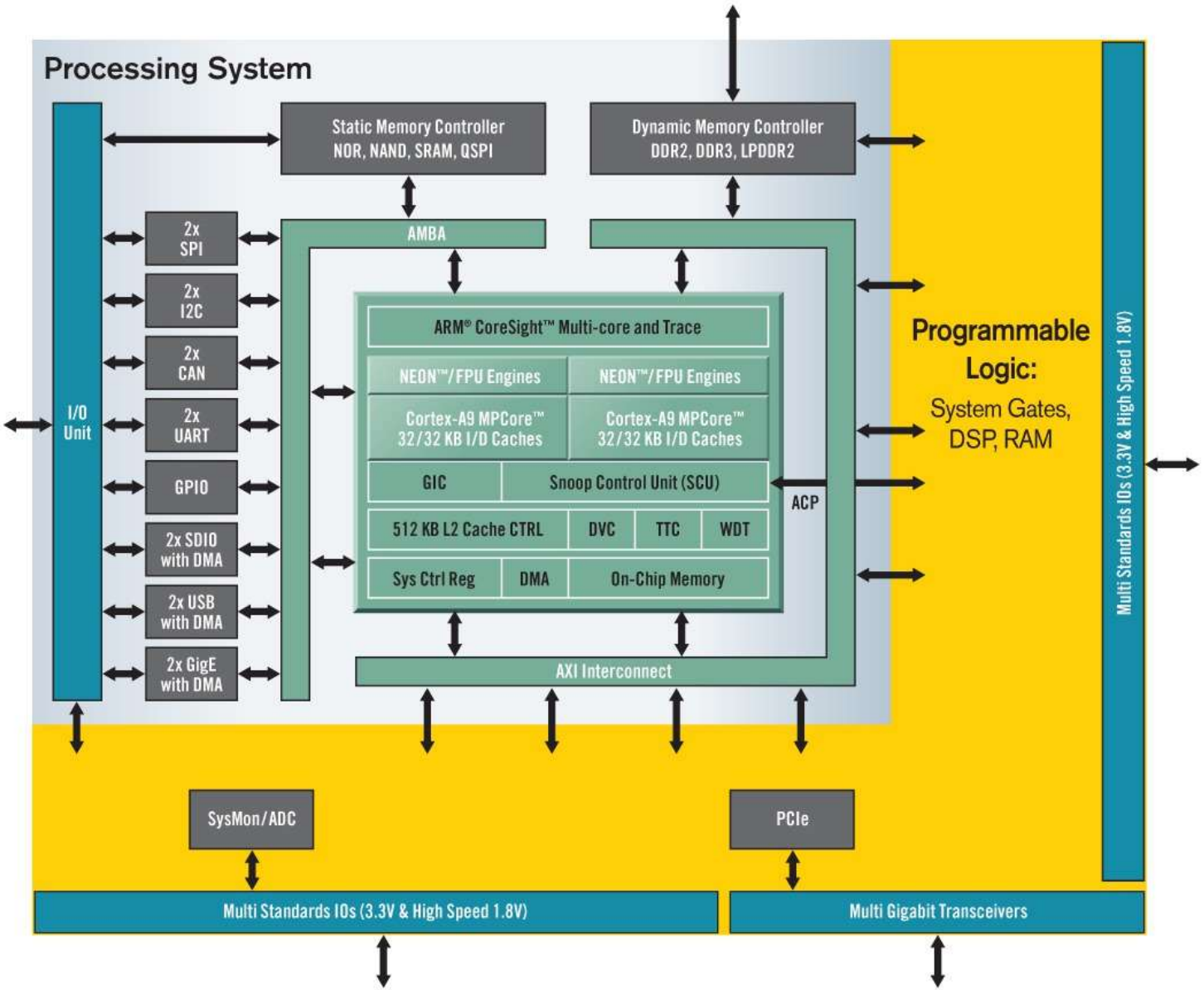
4 I/O Banks, Support for all I/O Standards including PCI, DDR333, RSDS, mini-LVDS



Up to eight on-chip Digital Clock Managers to support multiple system clocks

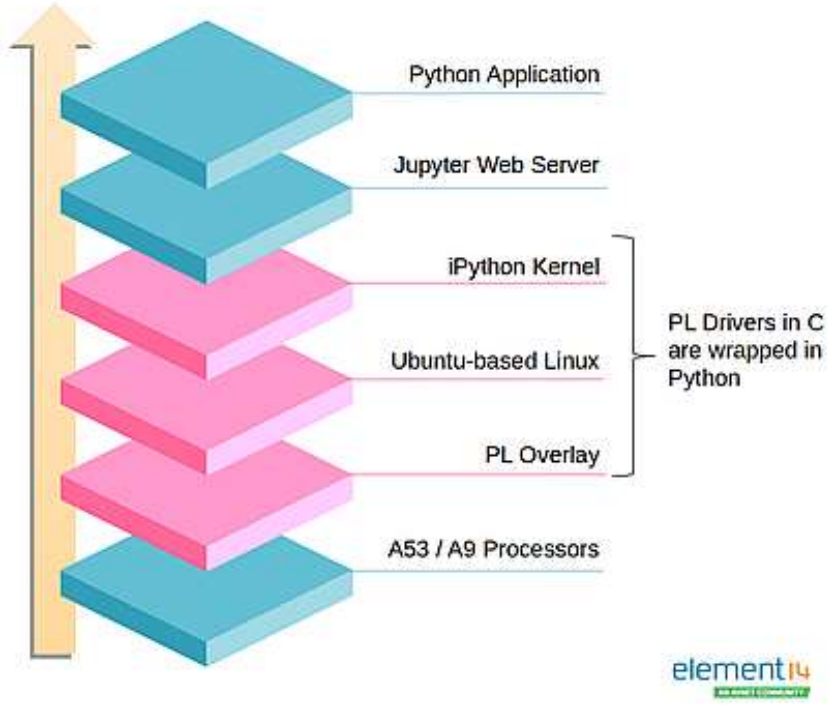
Xilinx Spartan 6 FPGA Architecture

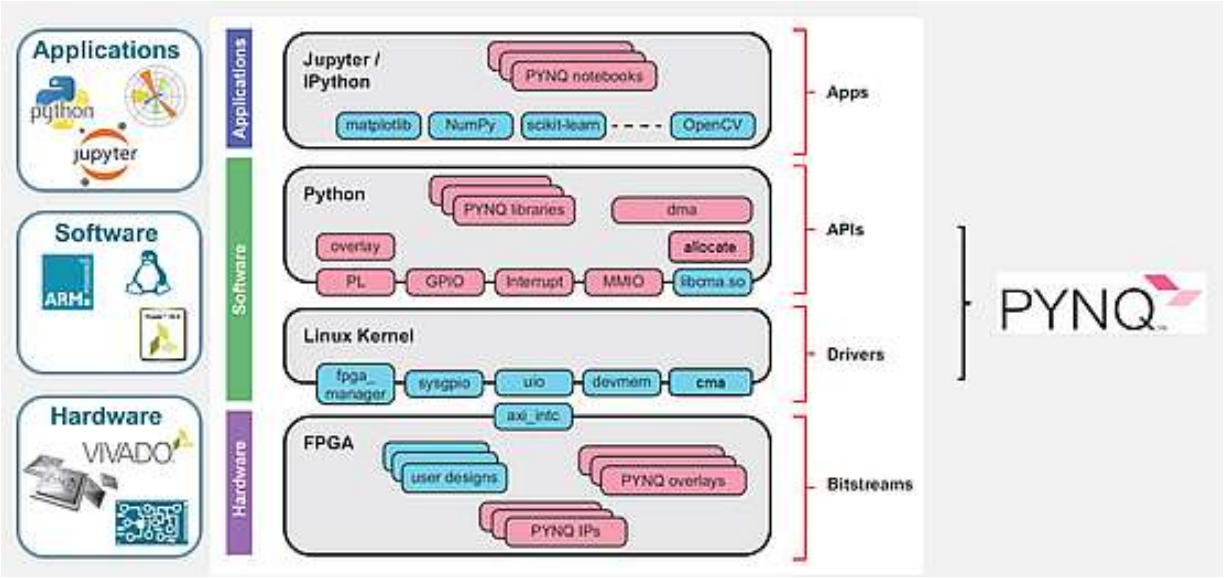
Zynq All Programmable SoC System Architecture



PYNQ: PYTHON PRODUCTIVITY

- http://www.pynq.io/board.html





ALU

Arithmetic Logic Unit - ALU

- Mathematician John von Neumann proposed the ALU concept in 1945.
- an ALU is a digital circuit that performs:
  - arithmetic operations and
  - logical operations

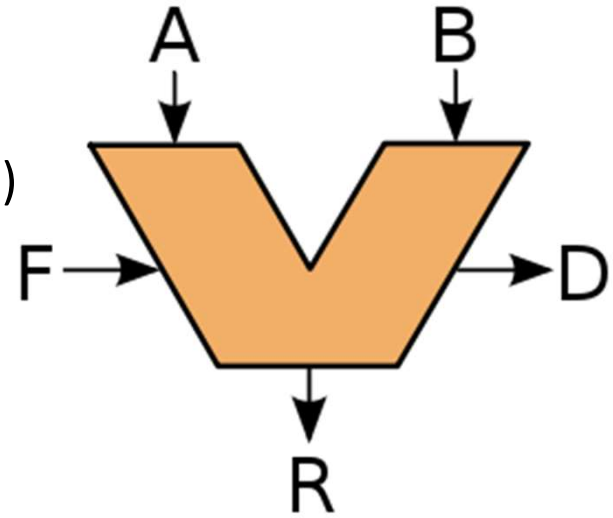
Legend

A, B – Operands (Registers, Data)

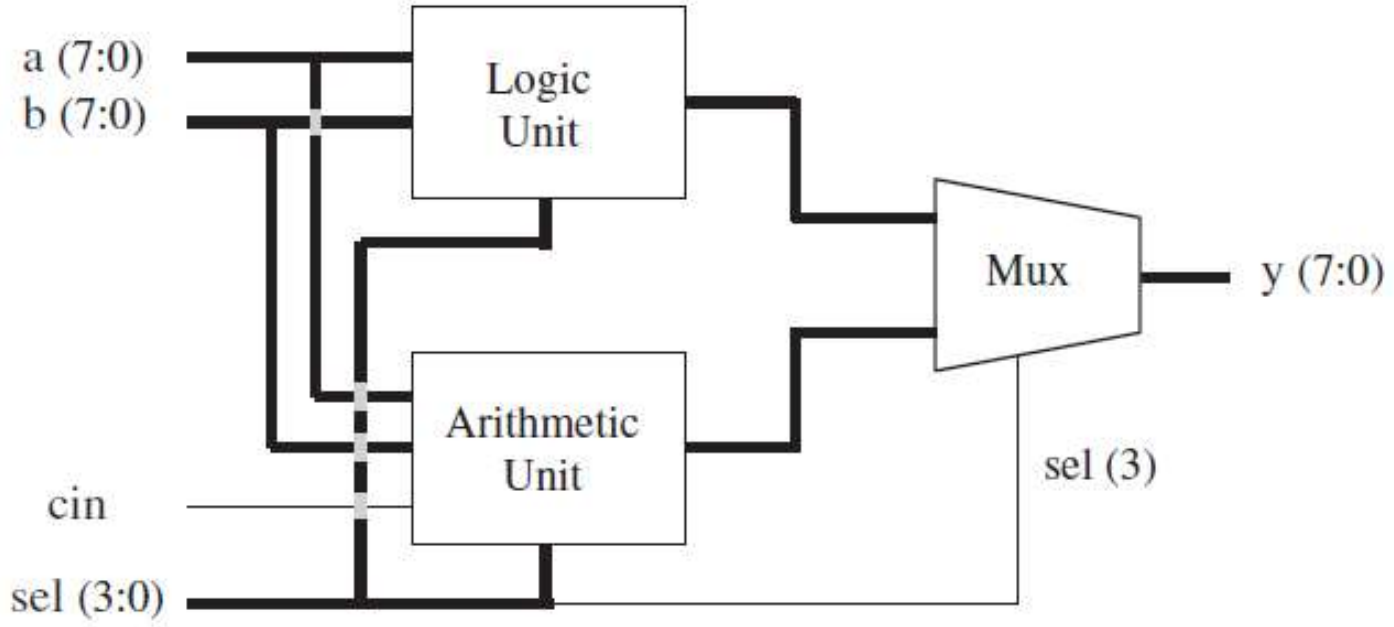
R – Result

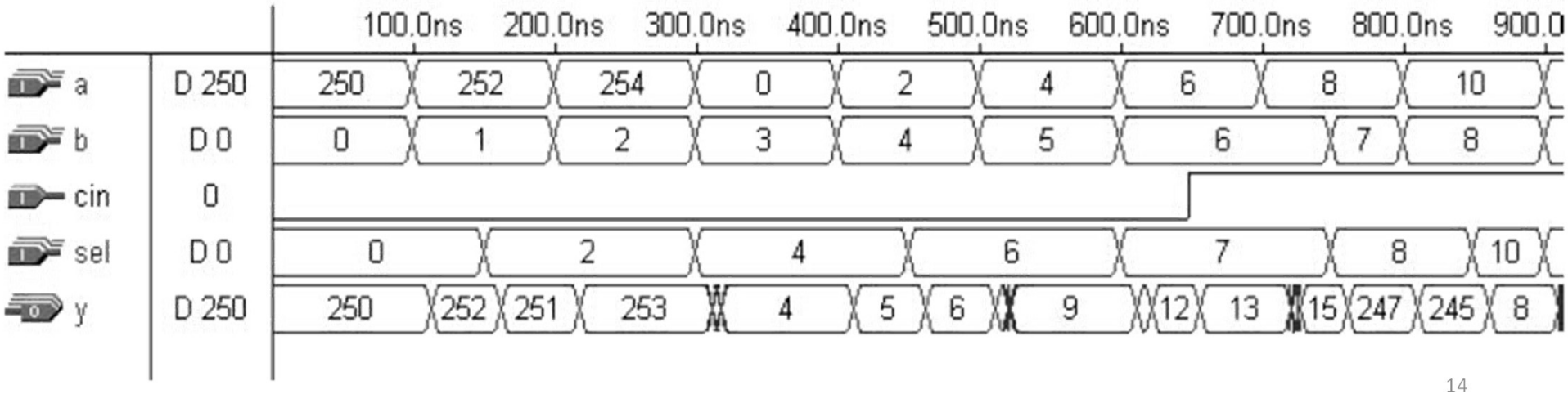
F – Instruction code

D – Output status (Flags)



Simple ALU – Block Diagram





ALU

sel	Operation	Function	Unit
0000	y <= a	Transfer a	Arithmetic
0001	y <= a + 1	Increment a	
0010	y <= a - 1	Decrement a	
0011	y <= b	Transfer b	
0100	y <= b + 1	Increment b	
0101	y <= b - 1	Decrement b	
0110	y <= a + b	Add a and b	
0111	y <= a + b + Cin	Add a and b with carry	
1000	y <= not a	Complement a	Logic
1001	y <= not b	Complement b	
1010	y <= a and b	AND	
1011	y <= a or b	OR	
1100	y <= a nand b	NAND	
1101	y <= a nor b	NOR	
1110	y <= a xor b	XOR	
1111	y <= a xnor b	XNOR	

ALU – VHDL Code (Entity)

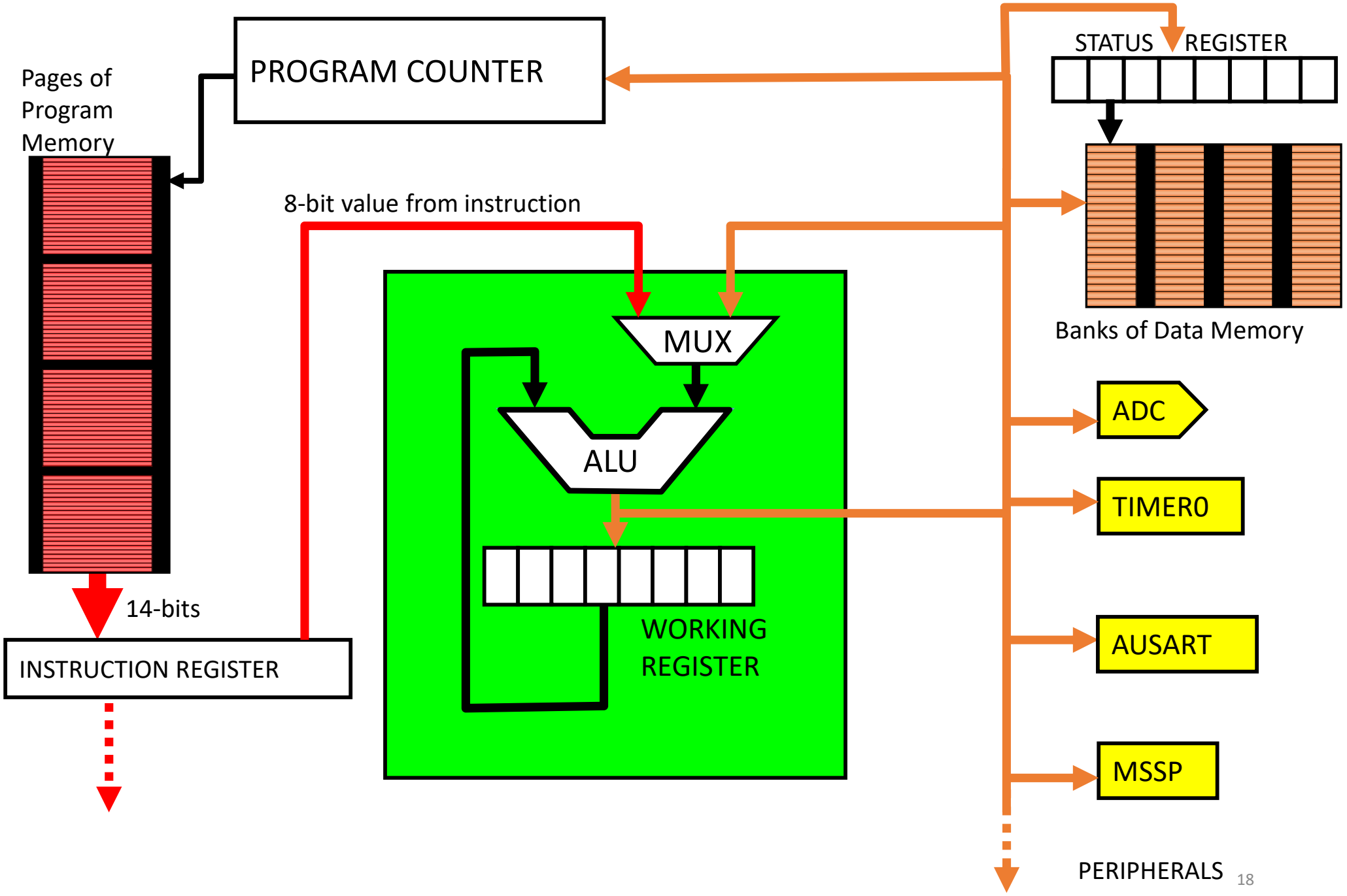
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity ALU is
port (a, b: in  std_logic_vector (7 downto 0);
       sel: in  std_logic_vector (3 downto 0);
       cin: in  std_logic;
         y: out std_logic_vector (7 downto 0));
end ALU;

architecture dataflow of ALU is
  signal arith, logic: std_logic_vector (7 downto 0);
begin
```

ALU – VHDL Code (Architetcure)

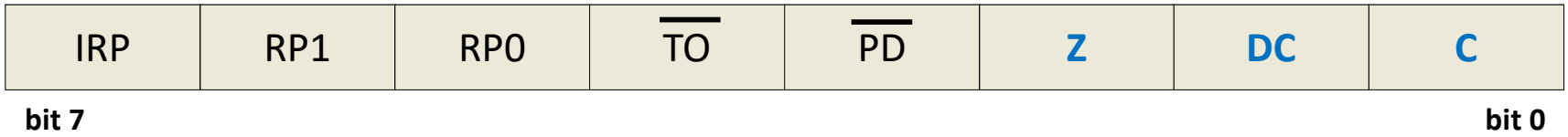
```
----- Arithmetic unit: ------
with sel(2 downto 0) select
   arith <= a when "000",
            a+1 when "001",
            a-1 when "010",
             b  when "011",
            b+1 when "100",
            b-1 when "101",
            a+b when "110",
        a+b+cin when others;

-------- Mux: ---------------
with sel(3) select
     y <= arith when '0',
          logic when others;
end dataflow;
```

```
----- Logic unit: -----------
with sel(2 downto 0) select
  logic <= not a when "000",
           not b when "001",
         a and b when "010",
          a or b when "011",
        a nand b when "100",
         a nor b when "101",
         a xor b when "110",
   not (a xor b) when others;
```

Mid-Range PIC Block Diagram



STATUS Register



IRP: Register Bank Select (used for Indirect addressing)

0 = Bank 0, 1 1 = Bank 2, 3

RP1:RP0: Register Bank Select Bits (used for direct addressing)

00 = Bank 0, 01 = Bank 1, 10 = Bank 2, 11 = Bank 3

$\overline{\text{TO}}$: Time-out bit

0 = A WDT time-out occurred

$\overline{\text{PD}}$: Power-down bit

0 = SLEEP instruction executed

Z: Zero bit

1 = Result of arithmetic operation is zero

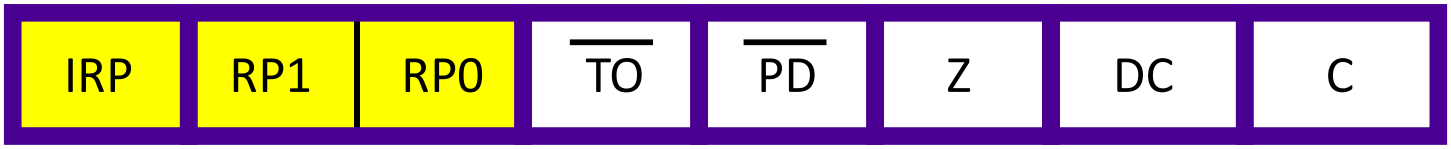
DC: Digit cary / $\overline{\text{borrow}}$ bit

1 = Carry out of 4th low order bit occurred / No borrow occurred

C: Carry / $\overline{\text{borrow}}$ bit

1 = Carry out of MSb occurred / No borrow occurred

Status Register



RP1	RP0	
0	0	BANK0
0	1	BANK1
1	0	BANK2
1	1	BANK3

CCR = Condition Code Register

- Contains:
  - Arithmetic status of the ALU
  - The RESET status
  - Bank select bits for data memory

Indirect Register Bank Select bit:
(used for indirect addressing)

1 = Bank 2,3

0 = Bank 0,1