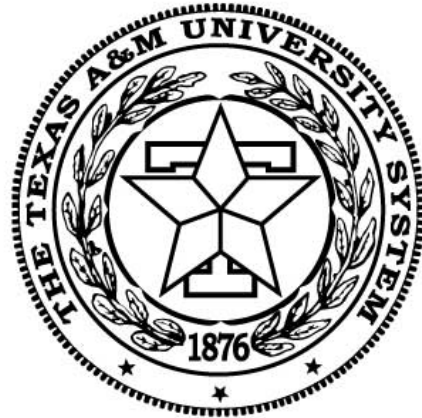


# ECEN 449 – Microprocessor System Design

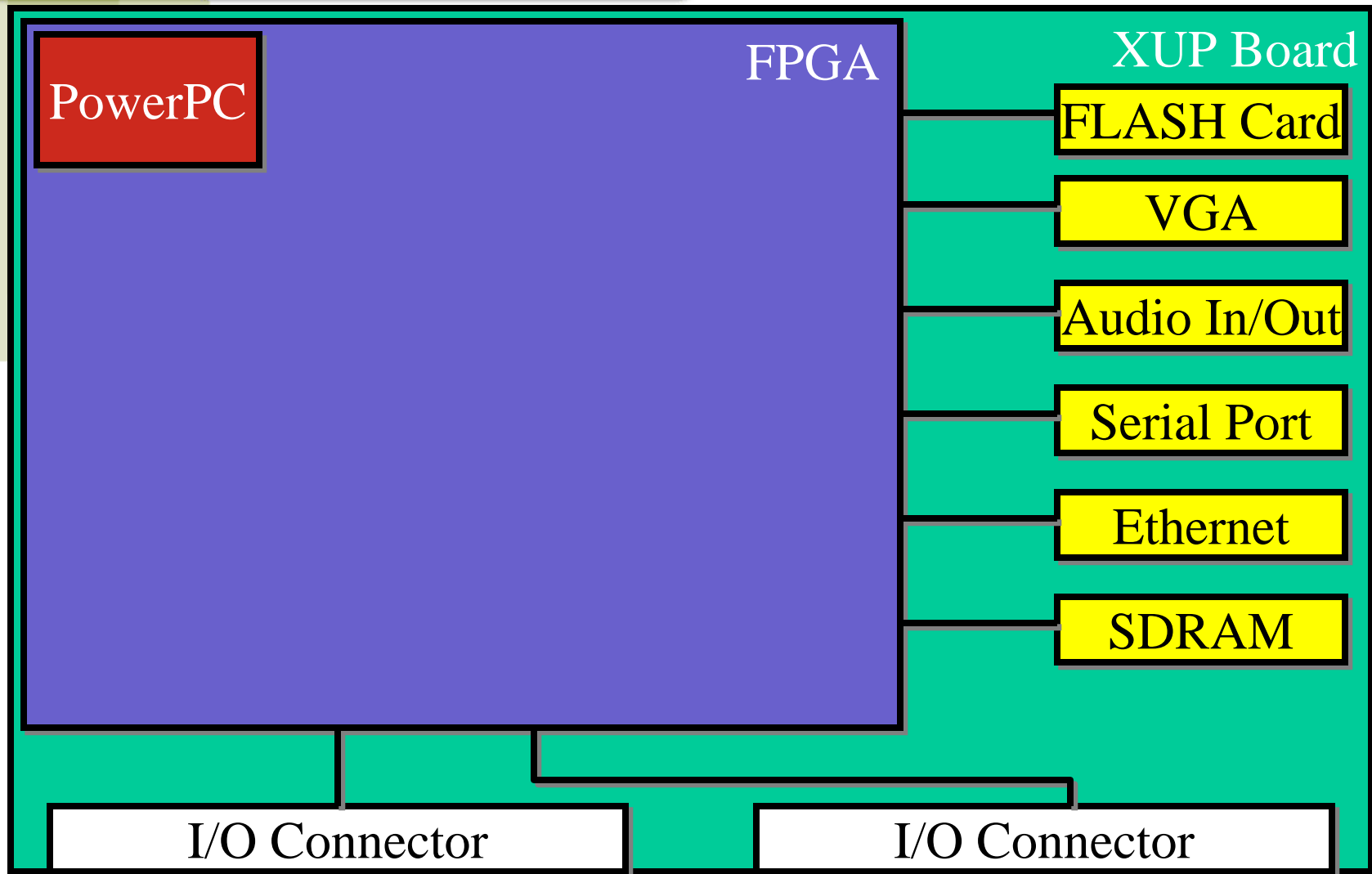


## The Xilinx Framework

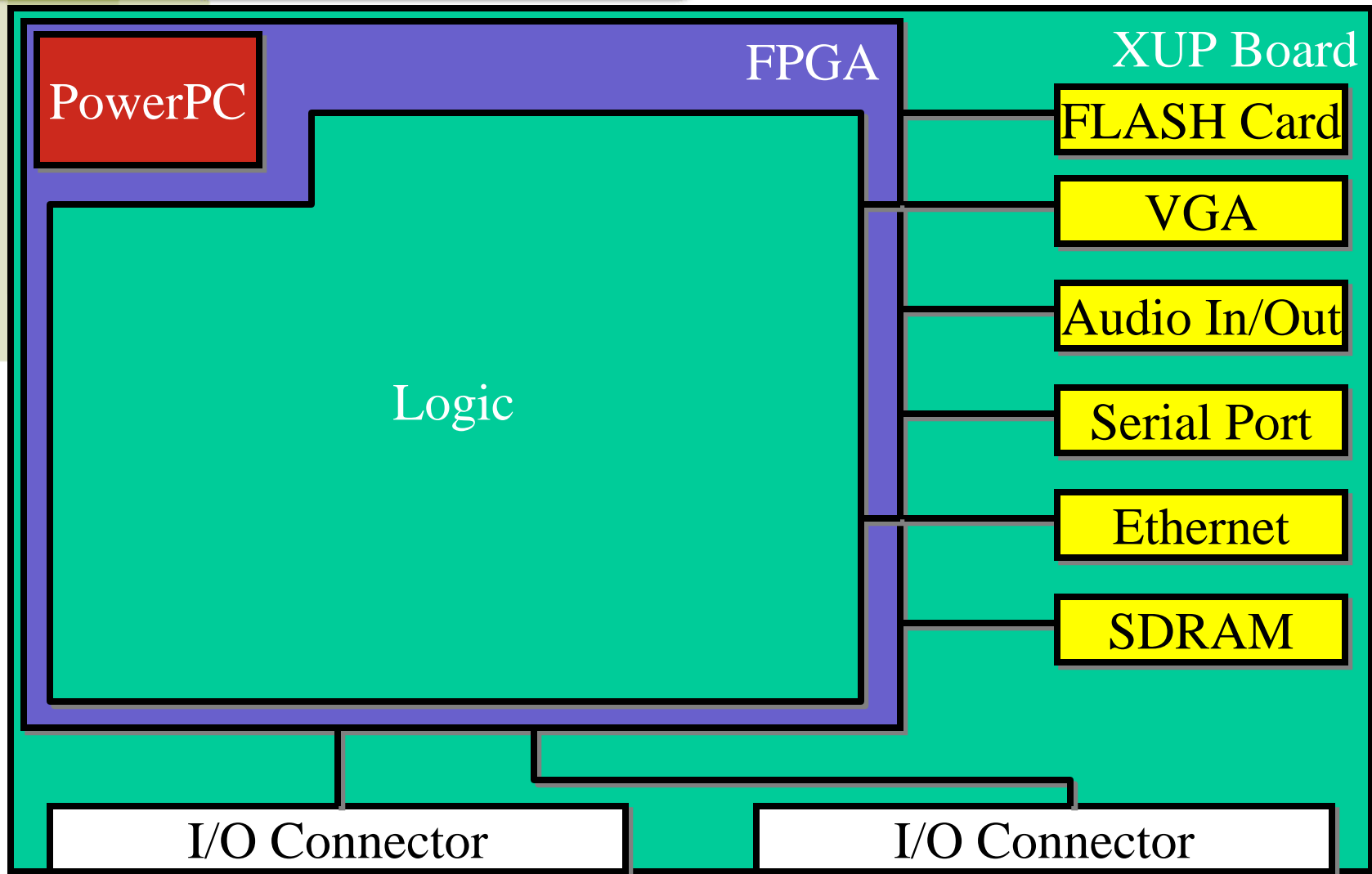
## Summary of this Unit

- Overview of XUP board / resources available on the board
- Discussion of the Virtex-II Pro's BlockRAMs

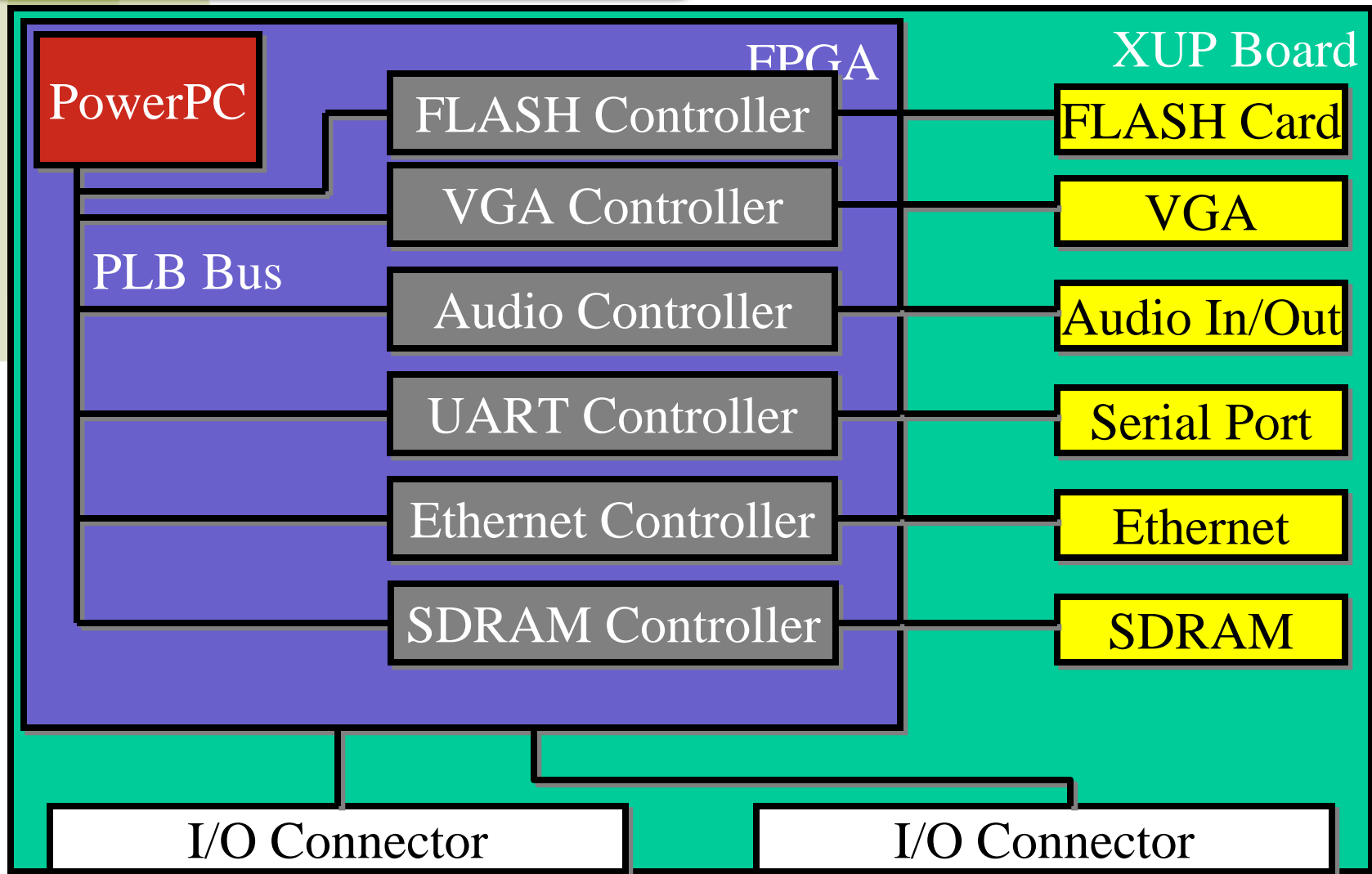
# The XUP Board



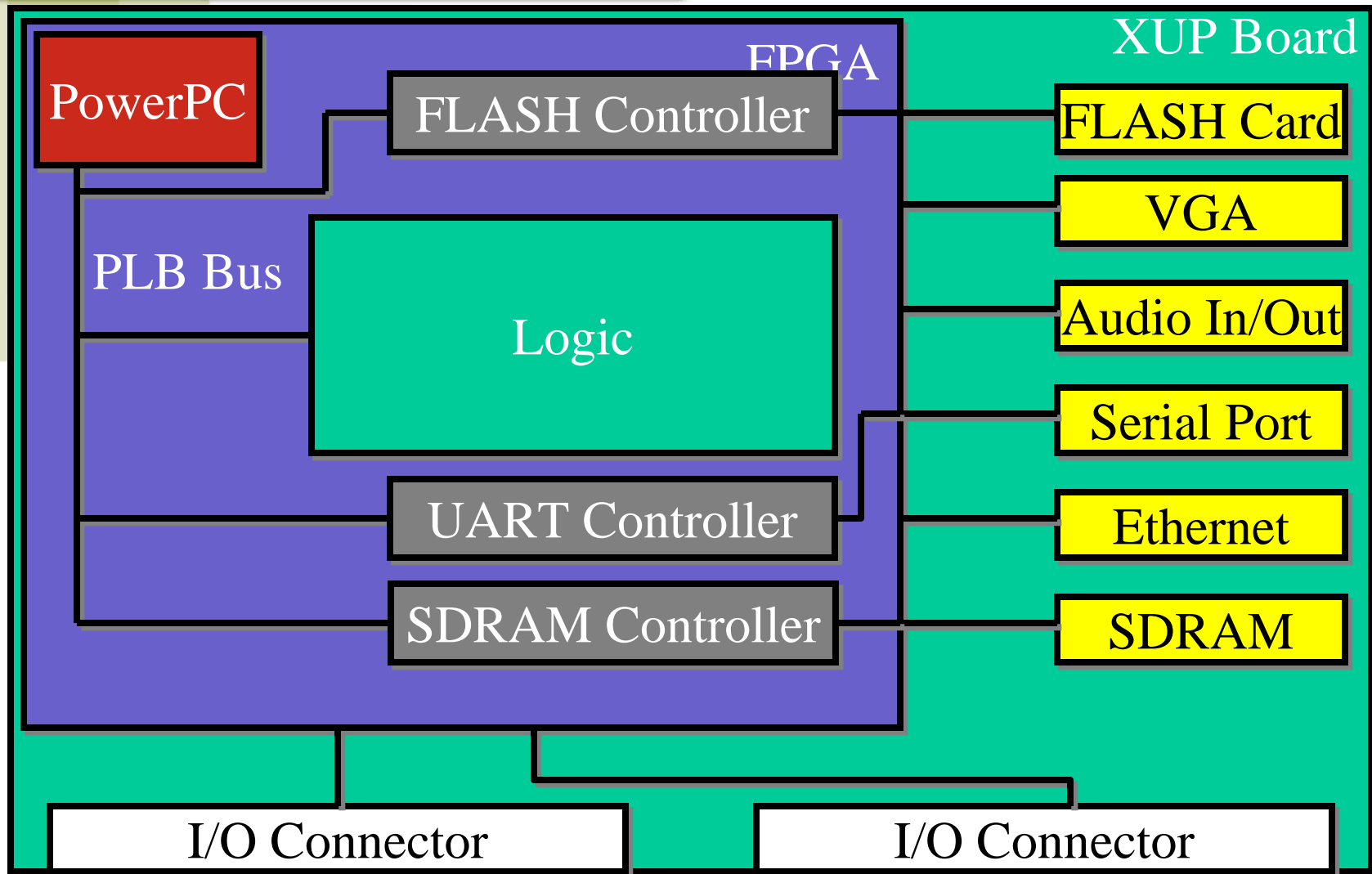
# All-hardware Design



# XUP Board as PC



# HW/SW System



## Creating a HW/SW System

- Custom logic needs to conform to bus/interface protocols
  - If Xilinx “IP cores” used for communication, the cores take care of this.
- Assemble custom logic, IP cores, software into a package that can be downloaded onto the board
  - EDK tool creates bit file for FPGA
  - SystemACE hardware loads Linux/other software off of FLASH card
  - Can also pre-load BlockRAMs with data as part of FPGA programming

# Interfaces to the PowerPC Cores

- Processor Local Bus
  - Three 64-bit busses that allow data to be moved to/from the instruction and data caches by off-chip hardware.
- OCM Controller
  - Allows memory (BlockRAMs) to be accessed at rates comparable to the caches
  - Great way to build data buffers
- Interrupt Controller
- Device Control Register
  - Allows creation of a register file that is “shared” among all of the devices connected to the PowerPC
- Clock, Power Management
- JTAG Port

## Other Resources

- SDRAM
  - Lots of space, complex interface requirements
  - Controlled via IP core
  - Your hardware can interface with core, simplifying things
  - Essential if the amount of data you need to access is large (SDRAM size is 512MB)
- BlockRAM
  - Small, on-chip memory
  - Can be configured in a number of ways
  - Fast, simple interface
  - Works when amount of data you need to access is smaller than 2.4Mbit (the size of the on-chip BRAM for the XC2VP30 device)

## Virtex Block SelectRAM

- 18Kb capacity and configurable at build time to be either:
  - 1 x 16K
  - 2 x 8K
  - 4 x 4K
  - 8 x 2K
  - 16 x 1K
  - 32 x 512
- 136 of these on each XC2VP30 FPGA for total of 2.4Mb total
- Dual ported, can be aggregated to form larger structures
- Parity bits, possible to pre-load with data in VHDL

# Interface Signals

*Table 3: Block RAM Interface Signals*

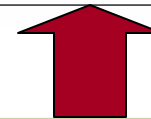
Signal Description	Single Port	Dual Port		Direction
		Port A	Port B	
Data Input Bus	DI	DIA	DIB	Input
Parity Data Input Bus (available only for byte-wide and wider organizations)	DIP	DIPA	DIPB	Input
Data Output Bus	DO	DOA	DOB	Output
Parity Data Output (available only for byte-wide and wider organizations)	DOP	DOPA	DOPB	Output
Address Bus	ADDR	ADDRA	ADDRB	Input
Write Enable	WE	WEA	WEB	Input
Clock Enable	EN	ENA	ENB	Input
Synchronous Set/Reset	SSR	SSRA	SSRB	Input
Clock	CLK	CLKA	CLKB	Input

## Supported Configurations

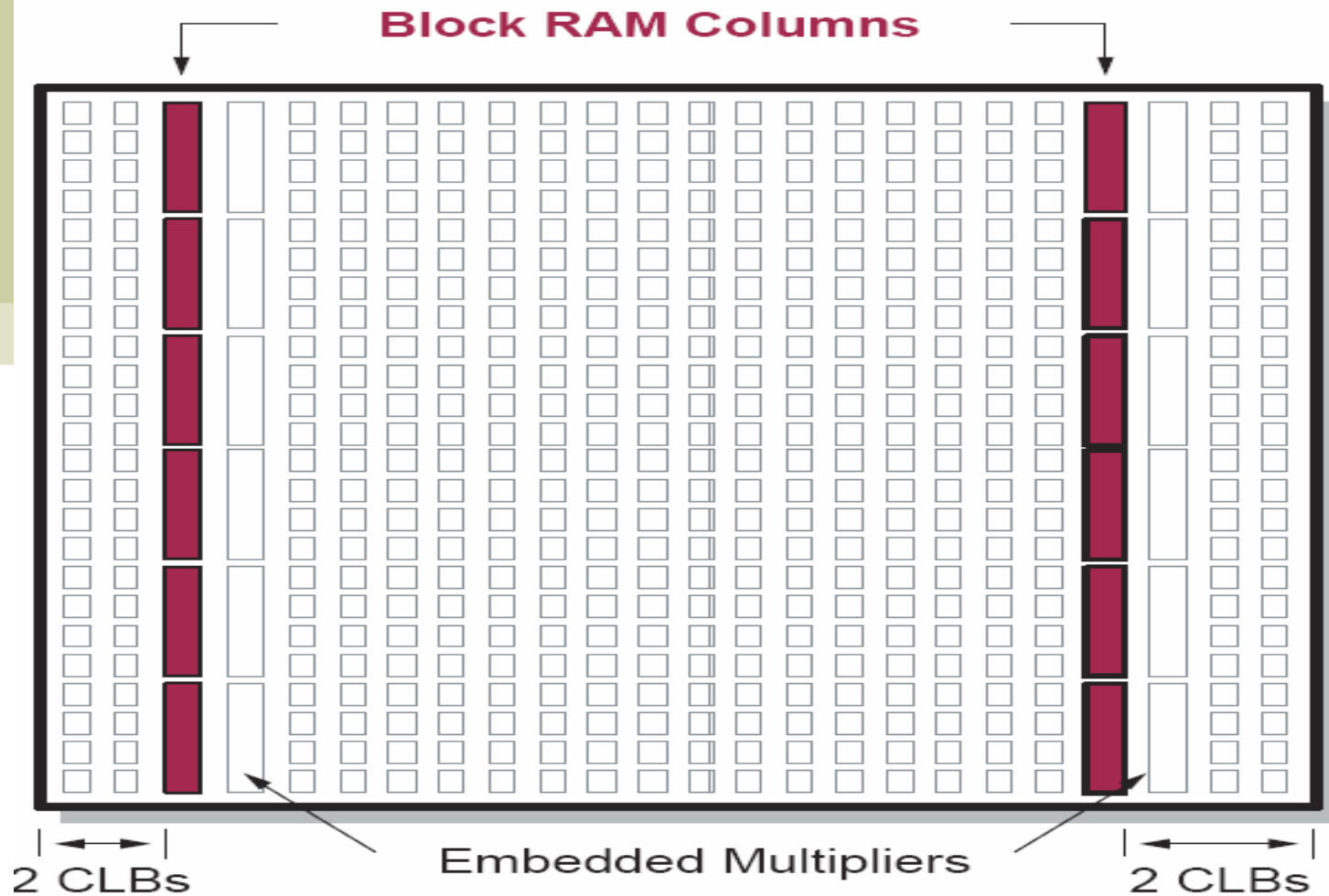
- Your VHDL will instantiate using *primitives*
- Each reference is an individual BRAM
- Could form larger memory block by assembling number of primitives, routing delay would determine total access time.

Table 5: Block RAM Data Organizations/Aspect Ratios

Organization	Memory Depth	Data Width	Parity Width	DI/DO	DIP/DOP	ADDR	Single-Port Primitive	Total RAM Kbits
512x36	512	32	4	(31:0)	(3:0)	(8:0)	RAMB16_S36	18K
1Kx18	1024	16	2	(15:0)	(1:0)	(9:0)	RAMB16_S18	18K
2Kx9	2048	8	1	(7:0)	(0:0)	(10:0)	RAMB16_S9	18K
4Kx4	4096	4	-	(3:0)	-	(11:0)	RAMB16_S4	16K
8Kx2	8192	2	-	(1:0)	-	(12:0)	RAMB16_S2	16K
16Kx1	16384	1	-	(0:0)	-	(13:0)	RAMB16_S1	16K



# Physical Location



## Instantiating BlockRAMs

- Declare as components, can instantiate multiple copies of a component

```
component RAMB16_s9  
    Port ( clk : in std_logic; ... );  
end component;
```

## Uses for BlockRAMs

- “Scratchpad” memories in designs
- ROM arrays -- can pre-load BlockRAMs with fixed values
  - CLBs are reasonably efficient at implementing ROMs, so don't need to use BlockRAMs unless you have a very big ROM
  - CLB's are very inefficient at implementing RAMs, want to use a BlockRAM any time you need more than a few words of RAM/register
- Buffer for Linux/hardware communication
  - One simple way to move data between Linux and hardware is to use BlockRAM buffers for data, interrupt/register write to signal when data is ready.