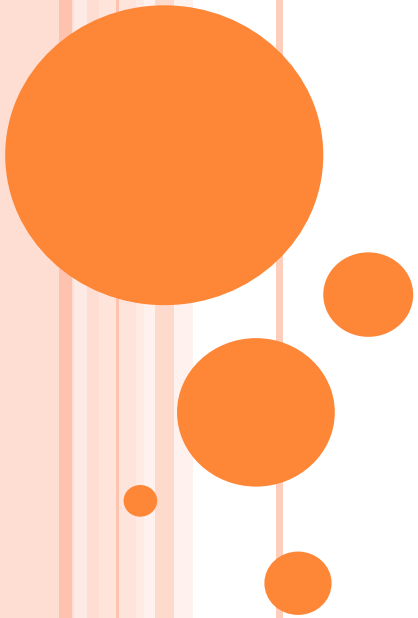


Design of Logic Cores

DESIGN FOR REUSE

- Design-for-reuse is an absolute necessity to:
  - maintain productivity levels,
  - keep the design time within reasonable bounds.
- Good designer
  - 100 gates per day, or 30 lines of RTL code a day.
  - 100K gate ASIC (a typical 1990s design)
    - 1000 designer-days -> 5 person team for about a year
  - 10M gate ASIC design
    - 100,000 designer-days ->500 persons for about a year
  - 100 M gate SoC design
    - *1,000,000 designer-days -> 5,000 persons for about a year! o*r 50 person team about 10years!

Design for Reuse Requirements

- Good functional documentation,
- Good coding practices,
- Carefully designed verification environments
  - thorough test suites,
- Robust and versatile EDA tool scripts
- Effective porting mechanism across various technology libraries (for hard cores).

Design Methodology for Logic Cores

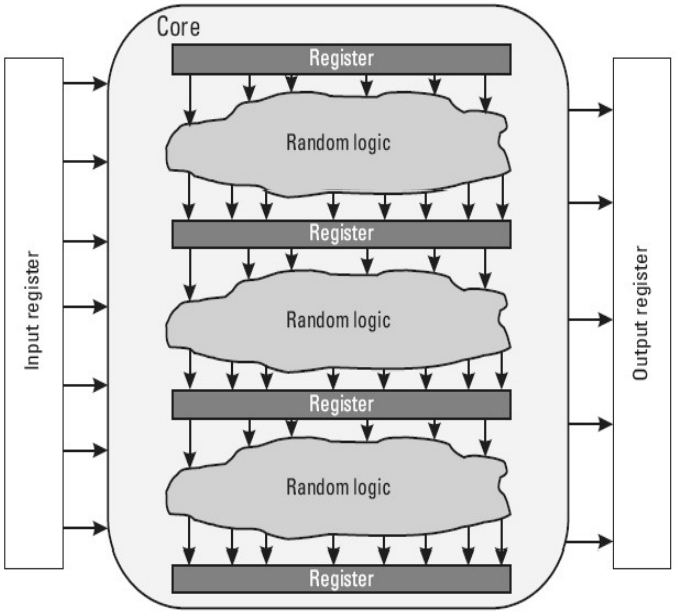
General Guidelines for Design Reuse Design Process for Soft and Firm Cores Design Process for Hard Cores

General Guidelines for Design Reuse

- Synchronous Design
- Memory and Mixed-Signal Design
- On-Chip Buses
- Clock Distribution
- Clear/Set/Reset Signals
- Deliverable Models

SYNCHRONOUS DESIGN

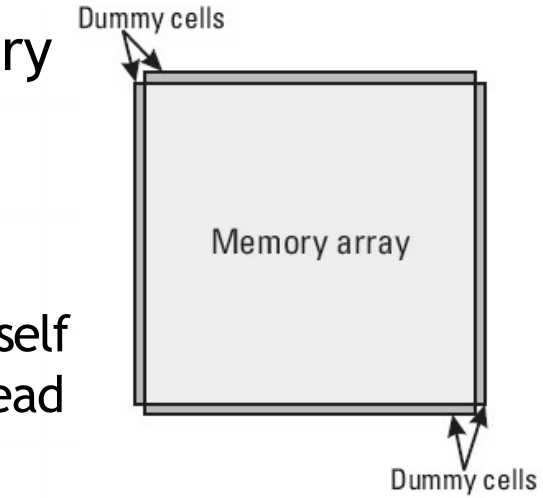
- Use registers for synchronization in core logic and its inputs and outputs to manage core-to-core interaction.
  - Creates a wrapper around a core.
    - portability
    - manufacturing test application
- Avoid latches in random logic
  - Use them only in blocks such as FIFOs, memories, and stacks

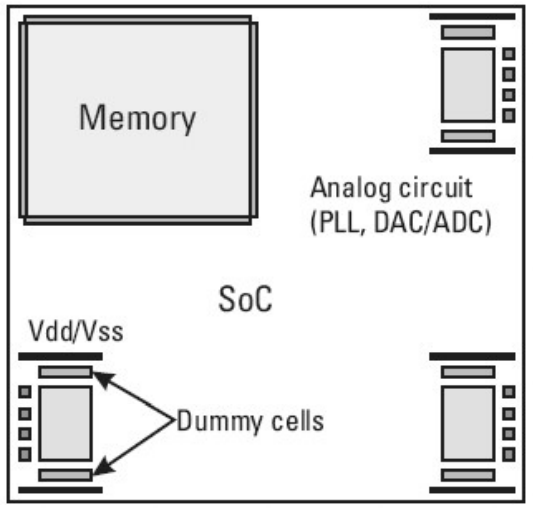


- Avoid asynchronous loops, internal pulse generator circuits, direct combinational paths from block inputs to outputs

MEMORY AND MIXED-SIGNAL DESIGN

- Large memories: different parasitics at boundary cells and a cell in the middle of an array.
  - Include rows and columns of dummy cells at the periphery of large memories
  - Make these rows and columns part of the built-in self repair (BISR) mechanism, to minimize area overhead
- most commonly used analog/mixed-signal circuits used in SoC: PLLs, ADCs/DACs, and temperature sensors.
  - extremely sensitive to noise and technology parameters
    - ❖ place them at the corners





ON-CHIP BUSES

- On-chip buses and data transaction protocol must be designed prior to the core selection process.
- Core providers cannot envision all possible interfaces.
  - Parameterized interfaces should be used in core design.
    - FIFO-based interfaces are flexible and versatile in handling varying data rates between cores and the system buses
- Organizations (VSI Alliance, …) develop on-chip bus and core interface standards/specifications.
  - support multiple masters, separate identity for data and control signals, fully synchronous and multiple cycle transactions, bus request-and-grant protocol

Clock Distribution

- Use the smallest number of clock domains.
- Isolate each clock in an independent domain.
- Use buffers at the clock boundary.
- Avoid metastability between clock domains interface
- Use synchronization method at the clock boundaries.
  - E.g., clock buffering and dual stage FFs or FIFOs at the clock boundary.
- Distribute a low-frequency chip-level synchronization clock when cores contain local PLLs.
  - Each core's local PLL should lock to this clock and generate required frequency for the core.

CLEAR/SET/RESET SIGNALS

- Document all reset schemes for the entire design:
  - Synchronous/asynchronous, internal/external power-on-resets,
  - any software reset schemes used,
  - does any functional block has locally generated resets,
  - whether resets are synchronized with local clocks, …
- Use synchronous reset if possible
  - avoids race conditions on reset,
  - static timing analysis difficult with asynchronous resets,
  - designer has to evaluate the reset pulse width at every FF
    - to make sure it becomes inactive synchronously to clocks

DELIVERABLE MODELS

- Design reuse depends on quality of deliverable models:
  - behavioral or instruction set architecture (ISA) model,
  - bus functional model for system-level verification,
  - fully functional model for timing and cycle-based logic simulation/emulation,
  - physical design models: floor planning, timing, and area
- Might be delivered in encrypted form to restrict piracy and reverse engineering.
  - create a top-level module and instantiate the core model inside it.
    - the top-level module behaves as a wrapper and hides the whole netlist, floor planning, and timing of the core

Deliverable Models (Need and Usage)

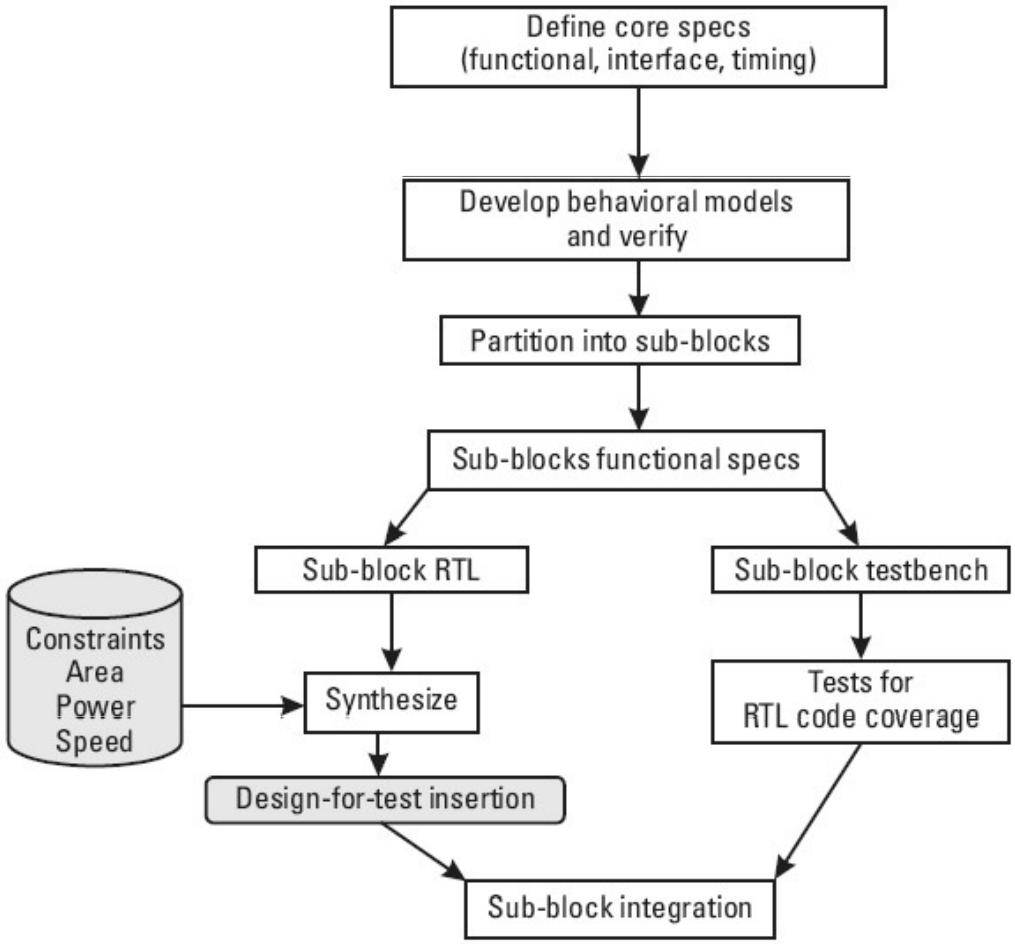
Model Type	Development Environment	Need	Usage
ISA	C, C++	Microprocessor based designs, hw/sw cosimulation	High-speed simulation, application run
Behavioral	C, C++, HDL	Nonmicroprocessor designs	High-speed simulation, application run
Bus functional	C, C++, HDL	System simulation, internal behavior of the core	Simulation of bus protocols and transactions
Fully functional	HDL	System verification	Simulation of cycle-by-cycle behavior
Emulation	Synthesized HDL	High-speed system verification	Simulation of cycle-by-cycle behavior
Timing	Stamp, Synopsis.do, SDF	Required by firm and hard cores	Timing verification
Floor plan/area	LEF format	Required by hard cores only	SoC-level integration and physical design

Design Process for Soft and Firm Cores

- Design Flow
- Development Process for Soft/Firm Cores
- RTL Guidelines
- Soft/Firm Cores Deliverables

DESIGN FLOW

- Design with a conventional EDA RTL synthesis flow.
- Reusability requirement
  ⓪
  multiple configuration tests should be developed and run.



Development Process for Soft/Firm Cores

- Required design specifications at every step in development process:
  1. Functional (purpose and operation)
  2. Physical (packaging, area, power, technology libraries, …)
  3. Design requirements (architecture and block diagrams with data flow)
  4. Interface requirements to specify signal names and functions, timing diagrams, and DC/AC parameters
  5. Test and debug (testing, DFT methodology, test vector generation method, fault grading, …)
  6. Software requirements (software drivers and models for hardware blocks)

RTL GUIDELINES

- RTLcoding style determines:
  - Portability
  - Reusability
  - Area and performance of the core after synthesis.
- So, develop RTL code that is:
  - Simple and easy to understand,
  - structured,
  - uses simple constructs and consistent naming conventions
  - Easy to verify and synthesize.
- Consult Verilog/VHDL books for good coding guidelines.

Soft/Firm Cores Deliverables

- Product files
  - Synthesizable source code
  - Application notes with HDL design example
  - Synthesis scripts & timing constraints
  - Scripts for scan insertion and ATPG
  - Reference library
  - Installation scripts
- Verification files
  - Bus functional model/monitors used in testbench
  - Testbench files including representative verification tests

SOFT/FIRM CORES DELIVERABLES(CONTD..)

- Documentation
  - User guide/Functional specification
  - Datasheet
- System integration files/tools
  - Cycle-based/emulation models as appropriate for macro and/or its testbenches and BFMs
  - Compilers, debuggers, real-time operating systems and software drivers for programmable processor IP
- Additional for firm cores:
  - gate-level netlist, description of the technology library, timing model, area, and power estimates.

Design Methodology for Logic Cores

- General Guidelines for Design Reuse
- Design Process for soft and firm cores
- Design Process for Hard Cores
  - Clock and Reset
  - Porosity, Pin Placement, and Aspect Ratio

CLOCK AND RESET

- Implementation of clock and reset should be independent of SoC clock and reset.
  - Since SoC-level information not available at the time of core design.
- Clock and reset require buffering and minimum wire loading.
- Clock must be available on an output pin of the core.
  - Used for synchronization with other SoC-level on-chip clocks.

Porosity, Pin Placement, and Aspect Ratio

- During SoC-level integration, often desirable to r
  core or through a core.
  - Hard core should have porosity, i.e., some routing chan
    the core should be made available; or
  - limit number of metal layers in the core to 1-2 less than
    maximum allowable by the process.
    - Deliverables for the core should include a blockage map to
      areas where SoC-level routing may cause errors due to cros
- Core pin placement affect SoC-level floor plan ar
  - Large logic cores are normally placed on one corner of t
    - Vdd/Gnd pins should be placed on one or, at most, two sid
      distributing them along all four sides,
    - Also, signals that remain primary I/Os at SoC level: e.g. US