Synthesis ELEC2570

Louis Devillez

25 décembre 2020

Table des matières

1 Design Flow

1. Concept
2. High-level design : Arch. description and embedded program
3. RTL coding
4. Behavorial simulation and verification : HDL code
5. Logic synthesis : Structural netlist
6. Structural simulation : Masks layout
7. Place and route : Physical netlist
8. Physical simulation and sign-off : Tape-out
9. Fabrication : Silicon wafer (silicon die for 1 unit on the wafer)
10. Dicing and Assembly (packaging and wire bonding) : Chip
11. Testing and sorting : product

Fab-less company : from Concept to Signoff
Pure-play foundry : from Signoff to Testing

2 Module A : From C code to embedded execution

2.1 PPA

The PPA is a good factor to evaluate a digital chip

— Speed **P**erformance in GHz :
  $f_{clk} \rightarrow$ CPI (not enough)
  MIPS (cpu) or GPOS/GFLOPS
— **P**ower consupmiton in mW :
  $P \propto f_{clk} \rightarrow$ Needs to be normalized
  mW/MHz, GOPS/W, pJ/inst, μJ/task
— Fabrication costs $\propto$ 1/ Silicon **A**rea

2.2 Business model map

- IP vendors : produce hardware and software intellectual property
- Semiconductor vendors : produce digital design
- EDA and foundries : implement IC design on silicon
- OEMs and ODMs : Original Equipment Manufacturer and Originad Design manufacturer
- Software developers
- Operators
- Retailers
- Consumer

2.3 MCU

Microcontroller are often used in embedded applications. The embeds various functions (IO,I2C, memory, ADC,DAC, Timer, Power management Unit, CPU). Low cost (high production volume).

Two main CPU : *ARM Cortex-M* and *RISC-V*. Cortex-M0 are predictable (in-order execution, built-in interrupt controller), portable (predetermined memory space)

Thumb instruction set : 16 bits instructions (limit code size). Subset of ARM instruction. Only the branch is conditional.

2.3.1 Processor core

Harvard architecture : 2 bus - 1 for data and 1 for instruction.
Von Neumann architecture : 1 bus for data and instruction.

16x32-bit registers

- R0 - R12 : general use register
- R13 : Main stack pointer/Process stack pointer
- R14 : Link register
- R15 : Program counter

Only 16 registers to stay low poer and have compact instructions.
Fast 32-bit multiplier : single cycle (in SW, 32 instructions) (optional)
Floating point hardware : (optional)

2.3.2 Interrupt

- NVIC : Nested vector interrupt control
- WIC : Wake-up interrupt controller
- NMI : Non-maskable interrupt

2.4 AHB

AMBA : Advanced Microcontroller Bus Architecture
AHB : Advanced High-performance bus, high bandwidth (pipeline, burst transfers, single-cycle master handover)
APB : Advanced Peripheral bus - low bandwith, low complexity
AHB-Lite : Subset of AHB. High performance bus :

— Two-cycle transfer : address phase, then data phase
— Wide data bus configuration (32-bit to 102B-bit data)
— Burst transfers and wait states
— single-clock dege operation
— MUX operation

2.5 RAM and memory

The RAM contains :
Data : static and global variables
Stack : temporary variable and parameters of functions call
Heap : dynamically allocated variables
The stack and heap grow in opposite directions.
The cortex-M0 can generate byte, half-ward, and word transfers

2.6 Architectural design

High-level synthesis : Synthesize RTL code from C/SystemC/Matlab. Improved poductivity but PPA still lagging far behind manuel coding.

Need SoC verification : First-time success is paramount and verefication is key (sooner the error is detected, the quicker a fix is implemented).

Verification : act of reviewing, inspecting and testing a design ir order to establish that it meet the specifications (functional and performance).

2.6.1 Dynamic verification

Dynamic verification : Simulate the design with a given set of stimuli to check thats its state and outputs are correct (performed at all design stages).

The number of states grow too fast ($\log^2$) and worst with CPU (test all SW). A possibility is to emulate the design on a fpga (SW developement before SoC production) but :
— Require time and expertise
— is also subject to human errors
— Dose not capture layout effects

A test bench is used to provide and collect data. Need also a clock and reset signal. (Testbench should be written by a different person than the designer). Waveform inspection or sequential generation of input stimuli is not scalable. Complex DSP need high-levl model to compare the outputs.

Assertions based verification : a statement that a certain property must be true, which falgs an error if not. It is possible to check block internal state. Can be specified by designer of verificatino teams. Widely adopted by the industry when there is a lot of IP blocks.

ABV can be used for measuring test coverage. The reach a coverage of 100% we need manual testbnech improvement with new mode of operations, function, range of input stimuli.

To cost of verification increase with the miniaturisation.

Metric-driven verification : use coverage-directed automatic random stimuli generation to exercise all parts of the design. Universal Verification Methodology (UVM) is a standardized metric-driven methodoly for RTL digital design with a focus on IP re-use.

2.6.2 Formal verification

Formal verification : Use of mathematical methods to prove that a design meets ets specification. It is defined by its
— Mathematical framework
— Verification technique

Logic equivalence checking : Prove the logic equivalence between golden RTL and varisous RTL during the whole design.
— Performed by a dedicated tool (e.g. Cadence Conformal)
— Works on both combinatorial and sequiential digital circuits
— Complicated by low-power features

2.6.3 Static verification

Static verification : Check that several constraints are met in the design.

3 Module B : From platfrom HDL to gate-level netlist

3.1 Principle

Logic synthesis : Conversion from HDL description to a gate-level netlist
— HDL : Descripton of the design
— Constraints on the design (.sdc)
PPA design intent constraints (e.g. timing)
Boundary and operating conditions
— Target technology and libraries (.lib/.db)
— Output : Verilog structural netlist and Constrain file (.sdc)
— Tool : Synopsys Design Vision / Design compiler | Cadence RTL compiler

Steps :

1. Analysis + elaboration : converts HDL to generic logic netlist
2. Mapping : convert generic netlist into standard cells from the core library
3. Optimization : optimizes the netlist to meet the PPA design constraints (iterative process)
4. Lint : Netlist sanity check to make sure the RTL/netlist is valid

3.2 Design constraints

— Design **intent** constraints : From the designer (e.g. PPA) $\Rightarrow$ .sdc
— Design **rule** constraints : from the foundra or library provider (e.g. thold) $\rightarrow$ .db/.lib

$\Rightarrow$ We meet first the design rule constraints to have a functionnal circuit and then the design intent constraints.

Performance trade-offs : we have a Pareto curve of optimum solutions between energy and Delay.

3.3 Timing closure

Setup timing constrraint : $T_{cycle} > T_{clk2Q} + T_{setup} + T_{hold}$ **Slack :** Difference between the timing of the capture path and of the launch path.

STA : Static timing analysis - method for computed the expected timing behavior of a synchronous logic circuit without simulation.

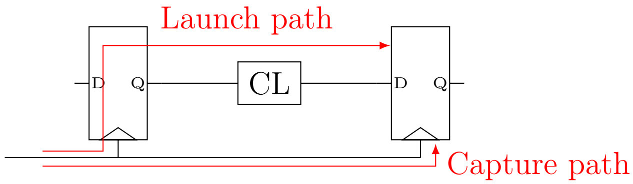

FIGURE 1 – Timing constraints

Timing exceptions : Everything whose timing constrain is not defined by $T_{cycle} > T_{C2Q} + T_{delay} + T_{setup}$. We have Asynchronous I/O or extremely relaxed timing constraint.

3.3.1 Timing closure

To achieve the timing closure
— Net list optimisation (simplify by moving gates)
— Gate mapping optimization (replace a groupe of gate by a gate).
— Pin swapping (invert A and B of gate to go through less transistors)
— Driving strength incerease
— Buffering centralized or distributed
— Pipelining (up to T_{setup}/T_{C2Q})
— Automatic architecture optimizations (e.g. choice the right adder to reach target PPA).
— Register retiming (moving register to break differently the path but can increase area and power if the number of register increase).
— Technology optimization (L8 and L9)

3.4 Standard-cell

Type of cells :
— Combinational cells : INV, NAND, NOR, AND, OR, XOR, MUX
— Sequential cells : DFFQ, LATQ
— Implementation cells : buffers, logic0, logic1
Cells are available in various driving strengths (Driving capacity).

Operating constraints (**PVT** corner)
— **Process** : slow - typical - fast. Trade off with leakage current
— **Voltage :** trade off speed - power consumption
— **Temperature :** trade off speed - leakage current
Strategies : Typical conditions (design evaluation) or worst-cas conditions (design sign-off).

3.5 Robust HDL coding

Non-synthesizable statements :
— Initial statements (not physical) ⇒ restable register
— Delay statements ⇒ constraints in the .sdc OR delay the signal by clock cycles
Undesired logic :

— Unwanted latch ⇒ add default case in *if/case* statements
— Combinatorial feedback loops ⇒ break the loop with register

3.6 Clock design

Multiple clocks can be
— Synchronous
— Logically exclusive
— Asynchronous

— Must be clean (no glitch)
— Generated by a crystal oscillator or a PLL
— *create_ clock* constraint in the .sdc defines how the timing analysis will be run

A multiple clock can be generated from a PLL and a clock division with a CLK gen (clock generated on chip). We need a synchronization FF at output of divided-clock to kills glitches. The C2Q delay induces a skew and there is a timing violation ⇒ annotate a zero delay for structural simulation.

Clock inversion, Glitch reduction and clock selection must be used with strong care (may require specific constraints).

If we want a selection between exclusive clock we need a glitch-free clock MUX.

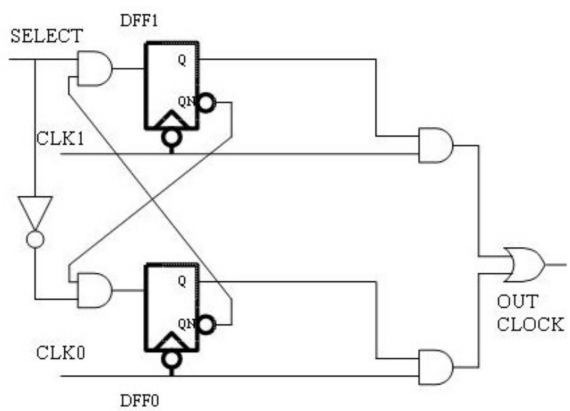

FIGURE 2 – Glitch-free clock MUX

Metastability : Time window T_W where setup/hold is violated. Two lip-flops in series form a double-latech barrier and improve the MTBF (Need to specify to the tool te be tolerant on _meta nodes

Mean-time between failure : $\dfrac{e^{S/\tau}}{T_W F_C F_D}$
— F_C : synchronizing clock frequency
— F_D : data changing frequency
— T_W : probability to enter Metastability

3.7 Reset design

— Synchronous reset : more resistant to glitches
— Asynchornous reset : easier design
— Massive reset : high fanout and bit capacity to drive
— to prevent timing violations with an external reset, the reset input shold be synchronized

3.8 Logic paths

— IN2REG
— REG2REG
— REG2OUT
— IN2OUT

IN2OUT are very rare and a well-balanced design should have REG2REG critical paths

Boundary conditions on input/output ports impact the timing and power of the design (slew rate and capacitance). To resolve I/O setup constrains we can forward the clock (have the same delay between capture and lauch path).

3.9 Power

Average power

$$P_{avg} = \frac{1}{T}\int_0^T i_{DD}(t)V_{DD}dt = P_{dyn}(f_{clk}) + P_{stat} \tag{1}$$

Switchig power (charging a node)

$$P_{SW} = C_L V_{dd}^2 f_{clk} \alpha_f \tag{2}$$

Short-circuit(during input transition when both transistors are ON)

$$P_{SC} = C_L V_{dd}^2 f_{clk} \alpha_f \beta_{SC} = P_{SW}\beta_{SW} \tag{3}$$

Leakage power

$$P_{Leak} = V_{dd} * W/L * \mu * C_{DEP*} U_{th}^2 * 10^{(Vgs-Vt)/S*}(1 - e^{-Vds/Uth}) \tag{4}$$
$$= V_{dd}\beta_{sub} * 10^{(Vgs-Vt)/S*}(1 - e^{-Vds/Uth}) \tag{5}$$

To reduce I_{leak} we can change the Vt

The internal power is the sum of the switching power and the short-circuit power.
The switching activity need to be annotated to have accurate power reports (.saif)

P_{sw} and P_{leak} have opposite trends $\Rightarrow$ minimu energy point (**MEP**) around 0.3 - 0.4V. We can reduce f_{clk} and V_{dd} to reduce the P_{idle} (reduce by V_{dd}^2). But it is limited by memories.

Clock gating : Disabled register at certain clock cycles to save power (α_F). The best implementation is with a Latch and a AND gate. It can be used to

— Iplement sleep mode and wape up with IRQ
— Disable unused HW acc/memories or Peripheral

Operand isolation : save switching power in unused combinatorial blocks can be done with AND gates or latches (manual or automatically) (α_f).

Power gating : Disable a circuit when not used (power shut-off). But challenges : states retention, output isolation, wake-up time, wake-up rush current.

3.10 Memories

Volatile		Non-Volatile	
Static	Dynamic	Read only	Electricaly programmable
SRAM macro	DRAM macro (trench capacitor)	ROM macro	Flash
RAM synthesized	DRAM macro with gain cell	OTP (eFuse)	Emerging memories
		Rom synthesized	

— Standard cells
— Standalone
— Can be embedded

3.10.1 SRAM

— Write : charge bitline to the value to write
— Read : pre-charge bitline to Vdd/2
— We need stability for read and write : Static Noise margin (SNM, in mV)

Synthesized RAms can be done with registers (for small size $<$ 1kB). The best implementation is to register the input address (not the output).

The bigger mux will reduce the RAT but increase the power and the area.

4 Module C : From gate-level netlist to physical layout

4.1 Physical implementation

Step 1 : The floorplanning

— the core size (dimension or the aspect ratio)
— the core to boundary distance.
— Core row utilisation

High utilization (lower area and possible shorter routing (lower delay - low C)

Low utilization (Routing less complicated, use space to do timing optimization, less heat/area)

— Row Organization : orientation and spacing

Step 2 : Placement of the hard macros
Step 3 : Placement of the well taps
Step 4 : Power rail (width and position $\rightarrow$ care to IR drop)
Step 5 : Clock tree

— short path are most prone to hold time violations
— Hold time violation cannot be fixed with a reduction of clock frequency
— Skew is temporal due to
Clock source
ambient and circuit noise
Supply/ground bouncne

The clock tree improves skew and transition but introduce a delay. It degrade
— REG2OUT setup closure
— IN2REG hold closure

Step 6 : Routing

1. Power nets
2. Clock nets
3. Signal nets (timing-driven)

We use buffer and repeater to lower the capacitance to drive (lower τ).
Crosstalk and Signal Integrity (SI) $\Rightarrow$ Lower the cap
Need decap cells to reduce load to drive

4.2 Timing optimization

— Add/delete buffers
— Resize cells
— Restructure the netlist
— Remap logic
— Swap pins
— Move instances

4.3 Packaging

— Package parsitics
— Wirebond inductance

ground/supply bounce at rising edge of the clock (flip-flop toggle) or whene output pad is changing (large C and large transistor). To prevent that
— Parallel connection (reduce parasitic inductance), Half IO $\rightarrow$ power pads
— Ground plane

4.4 Technology scaling

4.4.1 5V world and happy scaling

Moore's law : The number of transistor per chip doubles every 1.5-2 years
Transistor sclaling : Dividing T_{ox} L W by α
Interconnect : Connection between different metal layer
Interconnect scaling : Dividing H_{met} H_{met} Pitch by α
Limit to constant-voltage scaling : 5V and power
— Area overhead and speed penalty
— High power density
— Total power : battery lifetime concern

	Constant Voltage	Constant field
Dimensions : $W,\ L,\ T_{ox}$	$1/\alpha$	$1/\alpha$
Voltages : $V_{dd},\ V_t$	1	$1/\alpha$
Current par device : I_{on}	α	$1/\alpha$
Capacitance per device : C_{gg}	$1/\alpha$	$1/\alpha$
Area	$1/\alpha^2$	$1/\alpha^2$
Delay, $1/f_{clk}$	$1/\alpha^2$	$1/\alpha$
Energy per operation	$1/\alpha$	$1/\alpha^3$
Power density	α^3	1

Table 1 – Transistors scaling

4.4.2 Interconnect limitation

Limit to constant field scaling : the resistance of interconnect per length unit increases. Reduce the interconnect delay :

- New inteconnect (technology)
  Change material (lower resistance)
  Change dieletric (void)
- New technique (Design)
  Keep high metal level for global interconnect and not scaled
  Extensive use of repeater (buffering)

4.4.3 Leakage limitation

When the gate length becomes too small, the leakage power dominate. To limit the static power dissipation :

- Multi V_t techniques (Design) but high complexity
  Use low V_t on critical path
  Use high V_t on non-critical path (leakage reduction)
- Stop V_t scaling (Technology) but low speed
  Limitation V_t and V_{dd} (Velocity saturationd and mobility reduction)
  Strained-Silicon has Enhanced Mobility

4.4.4 Gate leakage current

The gate leakage exponentially increases with T_{ox} scaling. Short-channel effects

- Degradation of the slope,
- Shift of V_t (roll-off)
- S-D leakage current

$\Rightarrow$ Process diversifacation

- Low standby power (LSTP) CMOS
- Low operation power (LOP) CMOS
- High performance (HP) CMOS

$\Rightarrow$ Increase C_{ox} to limit short-channel effects $\Rightarrow$ Change oxide material

4.4.5 Variability and scalability

Speed binning : Regroup IC by Normalized leakage and frequency (due to variability - Random dopant fluctuations **RDF**).

variability on V_t limit maximum speed and increase total leakage on large chip. It also reduce the SNM for SRAM (failures). To mitigate the variability :

— New devices (technology)

Finfet : 3D CMOS - Pricer wafer but simpler process (No significant increase)

Full-depleted (FD) SOI : thin undoped channel (no RDF) in a buried oxide

— New techniques (Design)

4.4.6 Lithography

Since 180 nm generation, we use sub wavelength lithography

— Rounding effects

Gate length variation induces high leakage

Higher capacitance

Short circuit

— Line edge roughness (**LER**)

Varialbe line resistance/capacitance

Further V_t fluctuations

Solutions

— Optical proximity correction (OPC) -> interference

— Resolution enhancement techniques (**RETs**)

From 45/40nm : highe numerical aperture and phase shift masks

From 32/28nm : Immersion lithography

From 22 nm : multiple patterning

From 5 nm : Extreme Ultra-Violet (**EUV**) with 13nm wavelength

⇒ Wafer cost is exploding

5 Module D : HW/SW co-design

6 Homework

6.1 A1

6.1.1 Code

— Code : pure code

— RO : Read Only (eg. image input)

— RW : Read and write (data)

— ZI : zero init data (stack and heap)

6.1.2 Memory

the iROM is a non volatile memory and iRAM is a volatile memory (for stack + heap + ...). We need ram, because flash memory (ROM) has a limited number of writing and RAM is also low power.

6.1.3 Encoding error for 64*128

The heap is built in a ascending way and the stack is built in a descending way. If we go out of boundary we will have corruption