Synthesis ELEC2570

Louis Devillez

26 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
NRE : Non-recursive Engineering (cost)

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 GOPS/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

CPI : Clock per instruction.
GOPS : Giga operation per second.
GFLOPS : Giga floating operation per second.

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
- Software developers
- Operators
- Retailers
- Consumer

IP : Intellectual property.
EDA : Electronic design automation
OEMs : Original equipment manufacturer
ODMs : Original Design manufacturer

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

RTL : Register transfer level
High-level synthesis : Synthesize RTL code from C/SystemC/Matlab. Improved productivity 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-level 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 testbench 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 its specification. It is defined by

— 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 sequential 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

HDL : Hardware description language

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 constraint :

$$T_{cycle} > T_{clk2Q} + T_{setup} + T_{hold}$$

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.

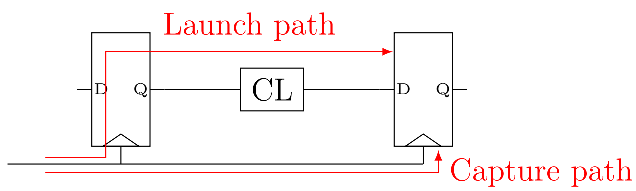

FIGURE 1 – Timing constraints

Timing exceptions : Everything whose timing constraint is not defined by $T_{cycle} > T_{C2Q} + T_{delay} + T_{setup}$ (e.g. asynchronous I/O, 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) $\Rightarrow$ restable register
- Delay statements $\Rightarrow$ constraints in the .sdc OR delay the signal by clock cycles

Undesired logic :
- Unwanted latch $\Rightarrow$ add default case in *if/case* statements
- Combinatorial feedback loops $\Rightarrow$ 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 $\Rightarrow$ 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.

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

MBTF : Mean-time between Failure $\dfrac{e^{S/\tau}}{T_W F_C F_D}$
- F_C : synchronizing clock frequency

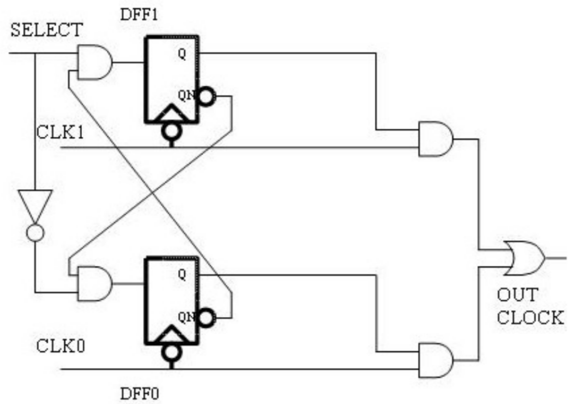

FIGURE 2 – Glitch-free clock MUX

— 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 IO 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)

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

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 V_t

— α_f activity factor
— β_{SC} short circuit factor

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 $\Rightarrow$ 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.
— Reducing V_{dd} reduce the power but increase the delay
— Reducing f_{clk} reduce the power but increase the delay
MEP : Minimum energy point

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
— Implement sleep mode and wake 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 : SNM (mV)
SNM : Static noise margin
Synthesized RAMs can be done with registers (for small size $<$ 1kB). The best implementation is to register the input address and not the output (to prevent power loss due to glitches).

The bigger mux will reduce the RAT but increase the power and the area.
RAT : Read access time

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 → 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

	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

Limit to constant-voltage scaling : 5V and power
— Area overhead and speed penalty
— High power density
— Total power : battery lifetime concern

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
⇒ Process diversification
— Low standby power (LSTP) CMOS
— Low operation power (LOP) CMOS
— High performance (HP) CMOS
⇒ Increase C_{ox} to limit short-channel effects ⇒ 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

5.1 Hardware accelerator

5.1.1 General-purpose processor

— Control-oriented applications
— Optimized for
  Conditional branching
  Flexible data moves (block, words,GPIO,...)
— Performance metric : MIPS
— Instruction set architecture (**ISA**) : RISC vs CISC
— Instruction execution : in-order vs out-of-order
— Memory architecture : Harvard vs Von Neumann
— Gpp clock speed increase due to optimizations (pipeline)
— Energy per instruction improved but not dramatically

5.1.2 DSP cores

— For signal (1D) or image (2D)
— Optimized for
  arithmetic functions
  regular memory accesses
— Performance metric : GOPS

code/DSP–pseudocode.s

```
LOAD R1, R4, R0
LOAD R2, R5, R0
MULT R1, R1, R2
ADD  R3, R1, R3
SUB  R0, R0, #1
BNE  R0, #0
```

MAC : Multiply accumulate, fusion of a multiplication and a addition ($R_3 = R_3 + R_1 * R_2$)

Hardware looping : improve the repetition of insructions sequentially on large data blocks (vectors/arrays) by getting rid of control instructions. It requires logic resources :
— Loop counter
— Instruction loop buffer
— Hardware data address generation

Triple data bus : Fetch data for 2 mac at once

Delay Register : add a delay block between the input of the two mac

Key ideas :
— Datapath / ALU with rich snigle-cycle arithmetic operation (MAC ...)

Architecture	#cycles	#PMEM accesses	#DMEM accesses
Von Neumann	8N	6N	2N
MAC instruction	7N	5N	2N
Harvard/Thumb ISA	5N	5N	2N
Hardware loop	3N	5N	2N
C(i) $\rightarrow$ PMEM	2N	N	N
Dual mac/3bus	2N/2	N/2	2N/2
Delay reg	N	N/2	N/2

TABLE 2 – Effect of DSP features (FIR loop)

- Memory based architecture with single-cycle multiple data accesses
- Specific addressing modes for loops
- Reduced control overhead for loops (instruction decoding, jump instructions

SIMD : Single-Instruction Multiple-Data is a technique of performing the same operation on multiple pieces of data simultaneously (GPUs)

GPU : Graphic processing units - massively parallel processor to maximize performance for graphirc workload charcterized by many fragments

Superscalar : a CPU implementing a form parallelism called instruction-level parallelism (**ILP**). It execute more than one instruction during a clock cycle (instruction dispatch done at execution).

VLIW : Very-long instruction word is a packing of single word of instruction (instruction dispatch done at compilation).

MPSoCs : Multi-processors Socs - When higher fully parallel performance are required

5.1.3 Hardware accelerators

HW accelerators are highly specialized to the functions which allows reaching very high energy/performance efficiency. The drawback is the lack of flexibility : in a generic DPS we need numerous HW accelerators (avantage on dark Silicon issues).

Dark Silicon : We cannot activate all transistors at once on a chip some need to stay off.

Typical application

- Generic : digital filters, min/max search
- Signal analysis : FFT
- Communications : convolutional encoding, error correcting codes
- Crypto functions : encryption/decryption, hash function
- Audio applications : compressions codecs
- Video applications : compressions codecs, image enhancement
- Sensor applications : Bloom filter, classifier with ML (SVM)

5.1.4 MPSoc architectures

Instruction fetch and datpath control has high energy cost. Memory access have also a higher energy cost. Data locality is key to have energy-efficient computing.

Single-core memory organization

— Register file (2 read port + 1 write port)
Two bottlenecks
Separate L1 cache
Single-bus L2 main memory
— The bypass bottleneck
Bypass circuit for pipeline execution stage to get rid of stall
Tag comparison on register index
PPA hungry in long pipeliens
— The register file (**RF**) bottleneck
VLIW share a common register file bypass network
RF require 2N read port and N write port (not an SRAM)
In practice, 5 issues in parallel is the maximum

Interconnect fabrics
— I/O-based transfers
Specific routing network connect the I/Os of several cores together
Low latency and power
Lack of flexibility
Difficult synchronization
— FIFO-based transfers
Fifo as buffer between cores
Allow asynchronicity betweeen PEs with 2 independent clocks
Lack of flexibility
— Memory-based transfer
Easier programming
Requires cache coherency management
Higher latency and power
— DMA-based transfers
DMA allow offload the PEs of their block memory transfers
PE sends a requets to the DMA and goes back to work/sleep
Several bus architecture can support DMA transfer (bus becomes a bottleneck when many PEs are interconnected).
— Network on chipts (**NoCs**)
Ultimate solution for many-core GALS architecture
Each PE/SE is a node connectd by a router (including FIFO)
PE/SEs can have independent embedded V_{dd} and clock generator
Can re-use all the techniques from the off-chip communication networks (TDMA, CDMA...)

5.2 Anthropocene

Absolute resource footprint : KPI unit x Intensity
KPI : Key performance indicator

$$CO_2e = Pop * \frac{A}{Pop} * \frac{E}{A} * \frac{CO_2e}{E} \tag{6}$$

Observations

— The followup of efficiency-improvement laws did not lead carbon footprint reduction
— Affluence increases mor than the efficiency

Hypothesis

1. The impossibility of green growth

   Increasing KPi is the way to generate economic growth

   Decoupling between economic growth and energy footprint

2. Escalating NRE costs

   All KPI are bounded by a physical limit

   Getting closer to the limit requires increasing innovation investetments (NREs)

   Return on investment requires increasing the affluence

   Companies compete to attrack capitals by promising growth of the stock value (speculative capitalism instead of accumulative)

3. ICT innovation pitfalls

   KPI innovation (e.g. 5G)

   Buzzword-driven innovation (ML for energy-efficiency optimization).

Consequences
— Creation of artificial needs

   High emission form onlive video

— Obsolescence generation

5.3 Ecological transition in ICT

1. Focus innovation on human needs

   Don't create needs

   Needs human interaction with the rest of the world

2. Replace KPI drive by reduction in carbon / resource footprint

3. Limit rebound effects

   How will this innovation be used

   Can we prevent abusive/unlimited use

   Needs for multidisciplinarity

6 Homework

6.1 A1

We can find those name at output of the compiler :
— Code : pure code
— RO : Read Only (eg. image input)
— RW : Read and write (data)
— ZI : zero init data (stack and heap)

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.

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 (for example with the 64*128 image)

The main is not the first thing executed in a program. We need initialization whchi will perpare the system to run the main. It can be observed after a reset

6.2 A2

The compression rate is data dependent.

Improvement of cortex M4 :
- Thumb I and II
- Floating point hardware
- Harvard architecture

Avoid int64 (2 registers) and float (no hardware). Instead of dividing or multiplying try to use shift. Also try to simplify mathematical expressions.

6.3 A3

There is 4 phase of simulations
- JTAG : Programmation of the SoC
- DCMI : Fetch of the data from the sensor
- Compression : using the JPEG-xs algorithm
- SPI : sending the compressed image

The `pcb_module` module here is to model in the simulation the pcb (physical connection - we have some capacitance pF to charge ns).

The `ahb_slave_mux` selects the slave to be accessed by the master nad multiplexes the *HRDATA* signals of the different slaves. In the project it also select the current master of the AHB bus (Cortex-M0/JTAG) based on TMS signal.

6.4 A4

See Section 3.5 about Robust HDL coding.

6.5 A5

A path is between a startpoint and a endPoint. Both for the launch path and the capture path. Relevant ? ⇒ Synchronous or not.

We need to clean signal from I/O (e.g. reset) so ween the latch the signal. The clock must have a proper SDC declaration (to allow metastability) but it's also true for the I/O signals (e.g. reset). For GPOUT, we should forward the clock (Hardware or software (ex : using Gpout[9]))

6.6 A6

The tool is under-estimating the activity factor (internal cell power) of the different net and of the memories. Golden rule : keep 10% of each phase (DCMI,encoding,SPI).

Clock gating reduce the power but add the gate on the capture/launch path so it can reduce or increase the slack.

6.7 A7

We have 3 choice to reduce memory power usage

- Use smaller memory (we have oversized memory)
- Use small memory to build a bigger one and disabled those memory when we are not using it (memory banking)
- Create a low power cache to reduce the number of access to PMEM

6.8 A8

$$\Delta t \simeq \frac{V_{dd} * C_l}{\frac{1}{2} * \mu * C_{ox} * \frac{W}{L} (V_{dd} - V_{th})^2} P_{SW} \simeq C_L * V_{dd}^2 * f_{CLK} * \alpha_F * N_{nodes} \quad (7)$$

$$P_{leak} \simeq V_{dd} * \frac{W}{L} * \mu * C_{DEP} * U_{th}^2 * 10^{\frac{-V_{th}}{S}} \quad (8)$$

The process will influence

- C_l
- C_{ox}
- W
- L
- V_{th}

The temperature will influence

- μ
- V_{th}

T_{delay}, T_{setup} and T_{hold} are affected by PVT corner.

To reach the timing closure

- Decrease row utilization
- Decrease the frequency
- Define false paths
- Run multiple optimization
- Overconstraint the logic synthesis

6.9 A9

Latency : Delay between the source and the sink.

Skew : Difference of delay between two sink.

Clock uncertainty : Model the skew compared to an ideal clock. The tool will take this data into account when doing the optimization.

During the the *place* stage, we have a better estimation of the capacitance (physical distance between cells is known) and the power estimation will increase.

6.10 A10

Starting point
— Throughput limited by the power budget
— Mainly dynamic power consumption

Possible choice :
— Lower V_{dd}
    Lower P_{dyn}
    Longer delay
    (Higher static power)
— Lower V_t
    Shorter delay
    (Higher static power)

6.11 A11

Execution profile
— Bit packing : 30%
— Image pre-processing and DWT : 36%
— Data packing : 16%
Slave HW accelerator
— data is in the memory
— Cortex-M0 must read/write it to the accelerato buffers
— SW loop for read/write
— +33.8% Throughput
— +13.3% Power
Slave and Master HW accelerator
— Data is in the memory
— HW accelerator can read/write directly the memory
— +46.8% Throughput
— +0.5% Power

HW accelerator with DCMI
— Data is already in the peripheral
— No accesses overhead
— Data needs to be stored
— +48.8% Throughput
— +0.8% Power
— +8.6% Area (Combinational logic +17%)

6.12 A12