Power Conscious Design Methodology for Actel FPGA

Mir Sayed Ali
Actel Corporation
MAPLD 2008

Agenda

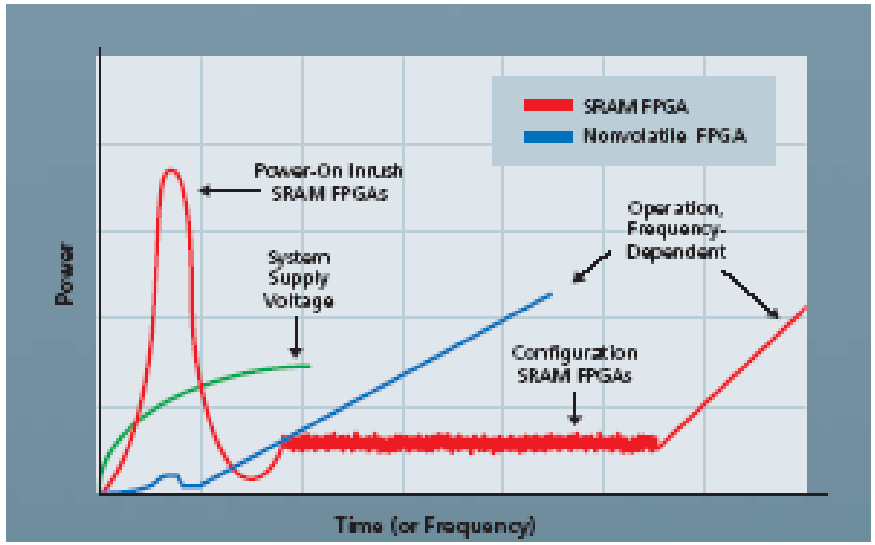
- The Law of Actel Power Aware Design
- FPGA Power Components
- Fighting Static Power
- Fighting Dynamic Power
- Power-Aware Tools
- Final Recommendation

The Law of Actel Power Aware Design

- Consider the big picture instead of the typical narrow interpretation of power numbers
- To reduce power, you have to
  - **Work as a team, and**
  - **Deploy efforts and time**
    - **System operation modes**
    - **Know your FPGA and associated enablers**
    - **Power analysis**
    - **Design changes**
    - **Relaxed Timing Constraint**
    - **etc.**

FPGA Power Components

- FPGA power components:
  - **Power-Up**
    - **In-rush Power**
    - **Configuration Power**
  - **Operation (over Temperature)**
    - **Static Power**
    - **Dynamic Power**
  - **Sleep mode power**
    - **Power during sleep mode**



- Actel FPGA:
  - **No power-up or configuration power in Actel FPGA**
  - **Very low power sleep mode**

Coping with Power: Proposed Methodology

- Understand end system operating modes, the power profile for each mode
- If Profile shows high sleep or shutdown modes
  - **Watch out for power up/down and programming currents**
  - **Tackle Static Power and board level design**
- If Not, then Check the Design Dynamic Power Profile
  - **Identify Major Bottlenecks for better RoI**
  - **Identify the best technique to fight Dynamic Power**

Static Power Dissipation

- Static Power is Dominated by *Leakage Current* in Various Forms:
  - **Sub-V_T leakage**
  - **Junction leakage (i.e., source/drain, well, and triple-well junctions)**
  - **Gate-Induced Drain Leakage (GIDL)**
  - **Gate leakage**
- Static Power Adversely Affected by Temperature
  - **Especially for Sub-100nm Technologies**
- Users Need to follow Simple Principles
  - **Smallest die**
  - **Master your FPGA architecture**
  - **Master Your Board Layout**
- Key is the selection of the FPGA with low Static Power

Fighting Static Power

- Smallest Die
  - **Smaller dies in a family have lower static power than larger ones**
- Know Your FPGA Architecture
  - **Examples:**
    - **Understand the various power-down modes of resources such as PLLs, RC oscillators, IO banks**
- Board Design Practices Include Thermal Management, Voltage Levels, and Resistive Loads
- Static Power is Adversely Affected With Increasing Temperature
  - **Keep the ambient temperature as low as possible**

Fighting Static Power

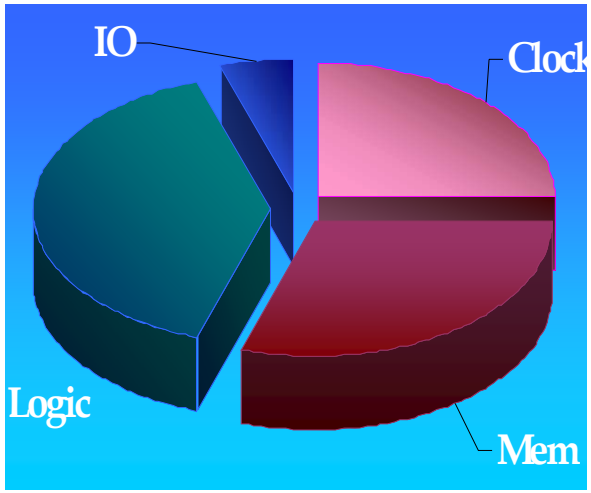
- General Recommendations:
  - **Driving inputs to the full voltage level so that all transistors are turned on or off completely**
  - **Tie off any unused power supplies (such as VCCPLL, VCCI, VMV, VJTAG, and VPUMP) to the ground plane**
  - **Use low-voltage CMOS I/O standard and the lowest drive strength**
  - **Avoid using pull-ups and pull-downs on I/Os because these resistors draw extra current**
  - **Avoid driving resistive loads or bipolar transistors, since these draw a continuous current, thereby adding to the static current. If they must be used, drive them to a level that consumes minimal power during static operation**
  - **When partitioning the design across multiple devices, minimize I/O usage between the devices**

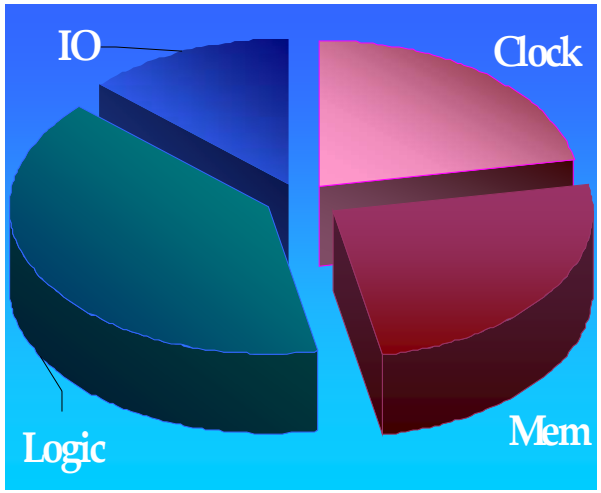
Fighting Dynamic Power

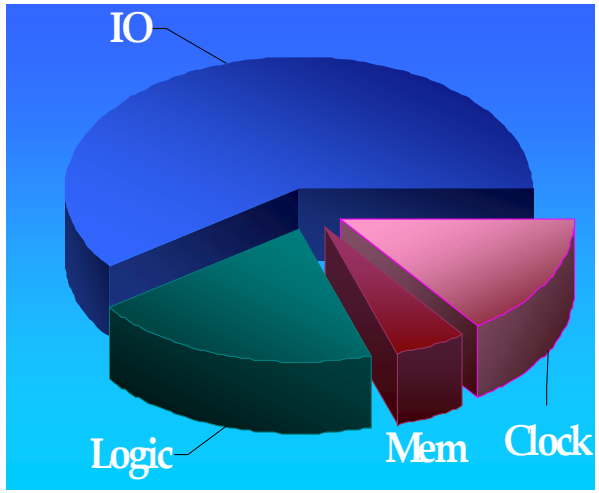
- Contributors to Dynamic Power
  - **Voltage**
    - **Voltage setting has quadratic effect on dynamic power**
  - **Toggling frequency**
    - **Dynamic power increases linearly with frequency**
  - **Loading**
    - **Dynamic power increases with capacitive loading**
  - **Switching**
    - **CMOS circuits dissipate power during switching**
    - **The more logic levels used, the more switching activity needed**
- First step to Fight Dynamic Power is to Design Dynamic Power Profile

Design Dynamic Power Profile

- Examples







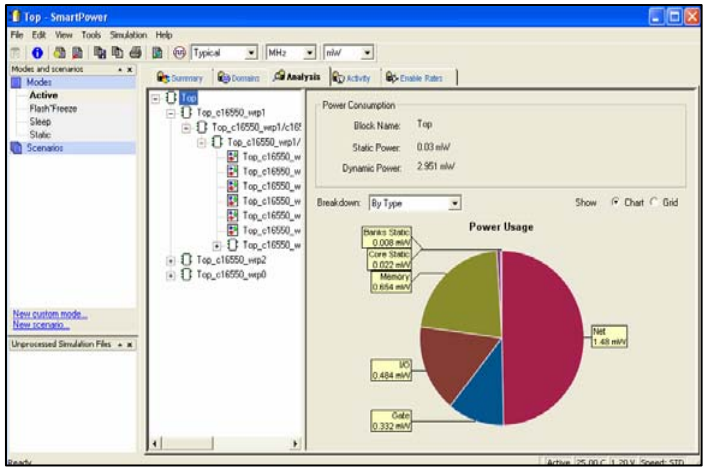
- Helps Evaluate ROI and channels the focus
  - **Tackling IO Power in case 1 is meaningless when compared to tackling Logic and RAM Power**
  - **Tackling IO Power in Case 3 is a MUST**

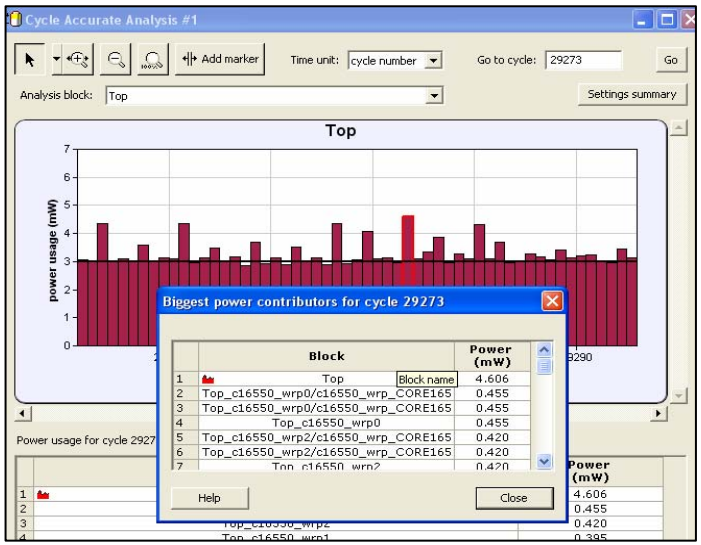
Actel Power Analysis Options

- 3 Levels of analysis
  - **Spreadsheet Calculator**
    - **Manually enter # registers, frequencies, clock domains etc.**
    - **Excel format, nothing hidden, save results for comparison later**
  - **SmartPower in Libero IDE**
    - **Import HDL/ Design and report on nets, logic, clock domains**
    - **Import Simulation for cycle accurate analysis**
  - **Using Dev kit or Pro Kit**
    - **Run your design and program to the board**
    - **Careful about passive element and the floating I/Os**

Identifying Bottleneck using SmartPower

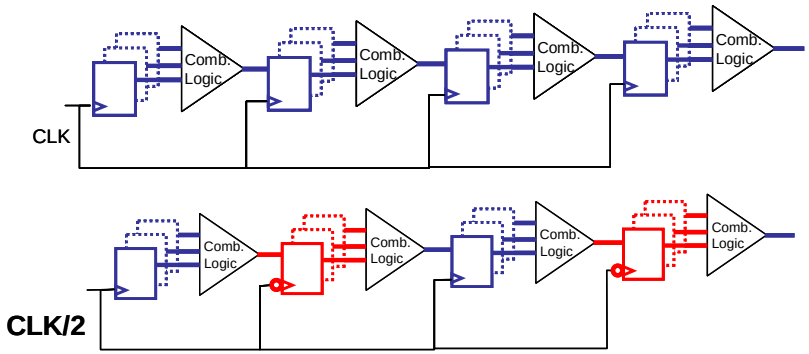
- SmartPower allows to analyze the hierarchy and specific instances within a hierarchy
- Custom modes for testing "what if" scenario
- Cycle-Accurate analysis view
  - **Obtain the cycle number, the start and end times, and the power usage for the selected cycle**
  - **View the most power-consuming blocks for a cycle**





Fighting Clock Tree Power

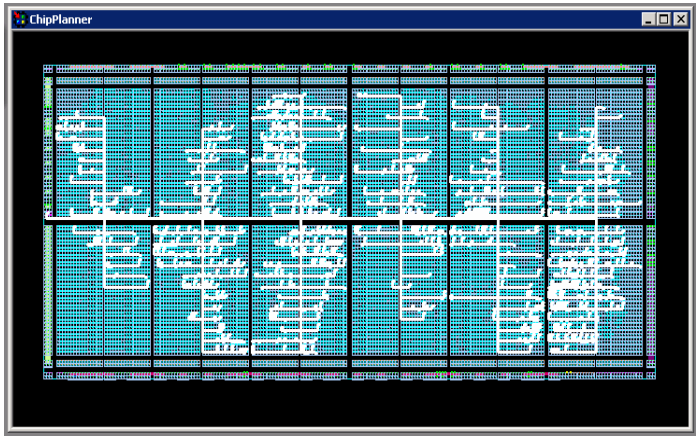
- To divide the clock by 2 or 4, use divider rather than PLL
- RTL Level Changes for Regular Pipelined Structure
  - **If timing allows, you could save half clock domain power**

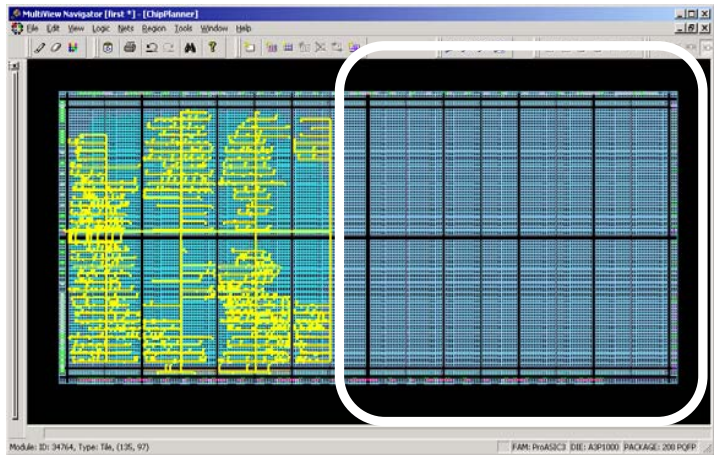


- RTL Level Changes for non Regular Pipelined Structure (loops back)
  - **Change possible, but tedious**

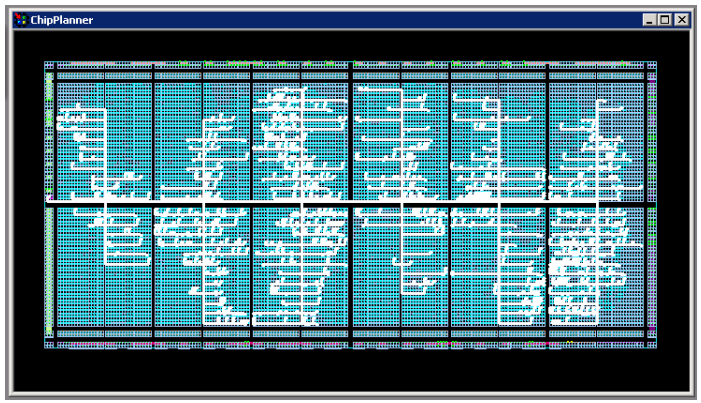
Fighting Clock Network Power

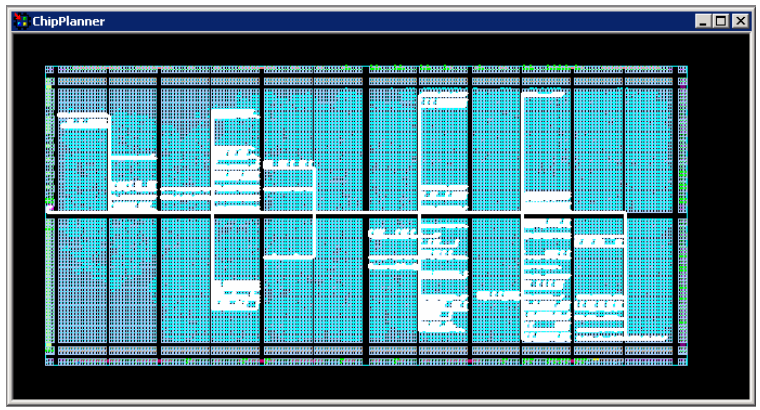
- Manual: Clock Domains Floorplan





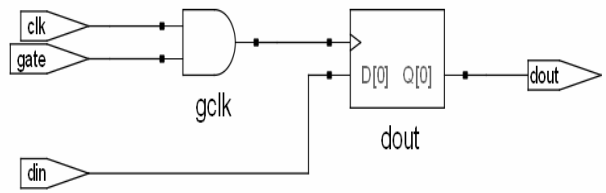
- Automatic: Power-Driven Place and Route



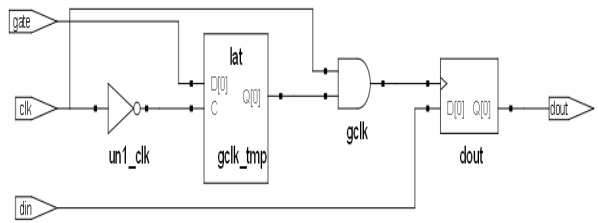


Clock Gating

- Classic Clock Gating Technique
  - **Latch-free clock gating:**
    - **Imposes a requirement on the circuit that all enable signals be held constant from the active (rising) edge of the clock until the inactive (falling) edge of the clock**
  - **Latch-based clock gating:**
    - **Adds a level-sensitive latch to the design to hold the enable signal**
- Identify groups of flip-flops which share a common enable and implement the clock gating
- Run STA



Latch-free clock gating style



Latch-based clock gating style

Summary Tables

- Clock Tree Power

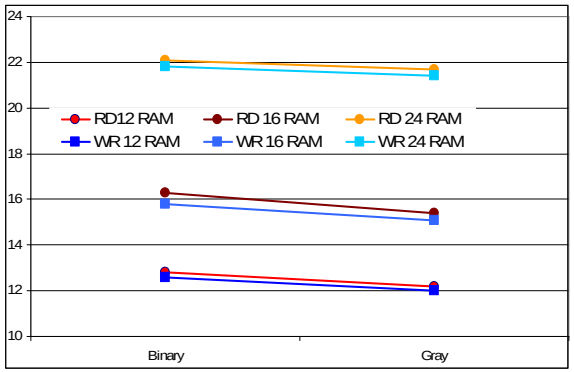
Design Option	Premises and Alternatives	Estimate Savings
Clock gating	Static timing analysis (STA) and functional validation required	Could be substantial, but design dependent
RTL changes to clock groups of registers with opposite clock edge	Timing must allow for the change	Peak power reduction depends on the size and routing in the logic driven by registers
RTL changes for Pipelined Logic	Easy: No feedback loops	Upto 50%
	Challenging: If feedback loops exist between stages. Require detail STA	10%+
Area Oriented Synthesis	Enough positive slack margin and sizeable blocks	Depends on the size of timing relaxed blocks
Power driven Place and Route	Need to combine it with Timing constraints and TDPR	Up to 10%
Clock tree Floorplan	Analysis of implicit placement constraints and potential artificial congestion	Depends on Die size and span of clock tree

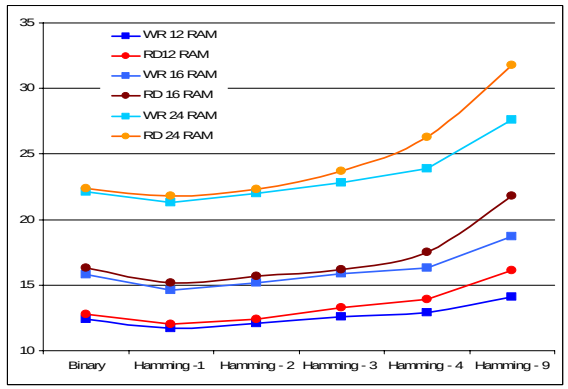
RAM Power

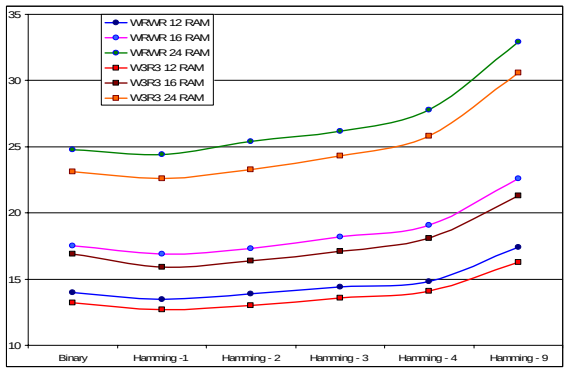
- READ (Involved Circuits)
  - **Address and control latches, Row pre-decoder, Read column decoder, Row final decoder, Read column decoder control, Sense amplifier, Data output muxes and latches, Sense enable logic, Read control logic, Bit-line precharge, etc.**
- WRITE (Involved Circuits)
  - **Address and control latches, Row pre-decoder, Write column decoder, Row final decoder, Write column decoder control, Write driver, Bit Line pre-charge, etc.**
- Characterization of Read vs Write
  - **Impact of Address Change**
  - **Impact of RE/WE Sequence**
  - **Impact of RAM Cascading Schemes**
  - **Gating of RAM Control and Clock Signal**

RAM Power (Flash FPGA)

- Read has slightly higher power dissipation than Write
- Reduce the Hamming_distances between successive addresses
- Use consecutive write or read rather than switching between write and read more frequently.

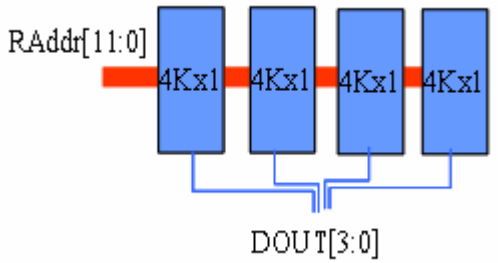




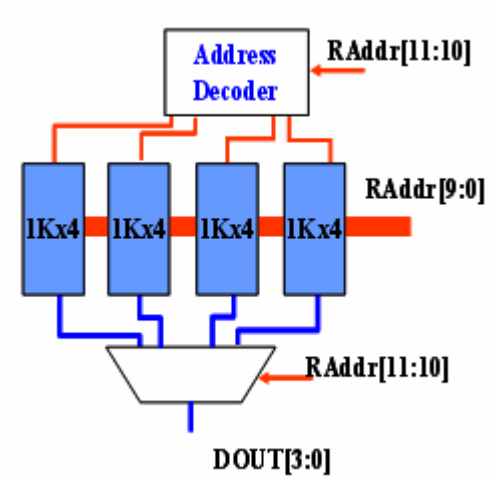


RAM Cascading

- Structural Alternatives
  - **Adopt cascading with all blocks active (Case1) when timing is critical and address are not local else use the proposed alternative (Case2)**
- Other Options
  - **Root vs. Leaf Clock and Enable Gating**
  - **Consider Read and Write Operations on opposite clock edges!**
  - **Consider Shutting down the RAM Block in between**



Case1



Case2

Summary Tables

- RAM Dynamic Power

Design Option	Premises and Alternatives	Estimate Savings
Cascading scheme	If overhead logic (muxing and decode) is timing critical and if address are not local, adopt cascading with all blocks active, elsewhere use the proposed alternative	"-2%" to "5%"
Root and leaf gating of the enable	If not possible adopt root gating of the clock	"5%" to "10%"
Write/read access sequencing	As many write operations before as many reads possible. If read/write are simultaneous, spread them over 2 edges of the clock (if timing allows)	"2%" to "5%"
Successive address changes	Least hamming distance	"1%" to "4%"

I/O Standards and Power Dissipation

- Use low Vcci for I/O
  - **Changing Vcci from 3.3 V to 1.5 V can save up to 80% of your I/O power**
- Determine the speed/waveform requirements in order to use the lowest possible drive strength
- Change the load to reduce the I/O power consumption
- Use Differential and resistively-terminated I/Os for highest toggling frequencies and single-ended I/Os such as LVCMOS for low frequencies
  - **I/Os that spend most of the time active will benefit from differential I/Os, but I/Os that are static most of the time may suffer from the higher static component of differential I/Os**
- Reduce the number of I/O by time multiplexing
- Reduce the activity or toggling rates of I/Os and eliminate unnecessary glitches
  - **Use tri-state output buffer instead of a simple output and monitor the logic that generates the enable signal of the tri-state**
- Divide the active outputs into two groups, positive clock edge and opposite edge whenever possible
- And, finally work closely with the board layout team for the defining of the pin assignment to I/O banks

Summary Tables

- I/O Dynamic Power

Design Option	Premises and Alternatives	Estimate Savings
Reduce the number of I/Os	Partitioning and/or time multiplexing	The number of I/Os contribution per I/O (I/O standard, voltage swing, and toggle rate dependent)
Use tri-state out buffer instead of plain output buffer	Drive the enable signal for tri-sate buffer cleverly	Proportional to enable toggle rate
I/O toggle rate	Investigate bus encoding	Depends on bus encoding quality
I/O toggle timing	Stagger I/O toggling in time if STA allows	No gain, but reduction in peak power
I/O standards selection and I/O bank assignment	For high toggling frequencies use differential or resistively-terminated I/Os standard For low toggling frequencies use single-ended with least voltage provided timing waveform etc requirement	The higher the frequency and the lower the voltage swing and the higher the saving

Power Profiling of Arithmetic

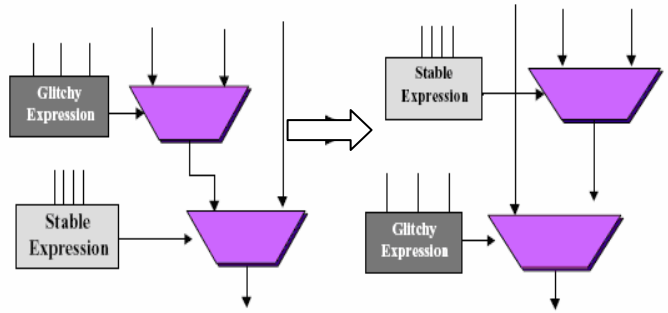
- Evaluate Synthesis mapping
- Adder:
  - **DesignWare BK Adder is the most power-friendly**
  - **A Paper covering Power-Aware Adders Architecture – Coming Soon**
- Multiplier:
  - **Must consider not only power but also area/speed**
- Counter:
  - **Use Model**
    - **Used to account for a sequence of events, an elapsed time,**
    - **Used to drive a load such as RAM address or data busses, a state machine's next-state or output logic, etc.**
    - **Outputs are used to perform certain processing when they reach various decoded values**
  - **Recommendation**
    - **If counter is only used for final count flag,**
      - ***Binary is most power friendly***
    - **If counter is driving a large load bus**
      - ***Gray offer best power saving***
    - **If several counter values are decoded, Ring counters are the choice (decode logic is least)**

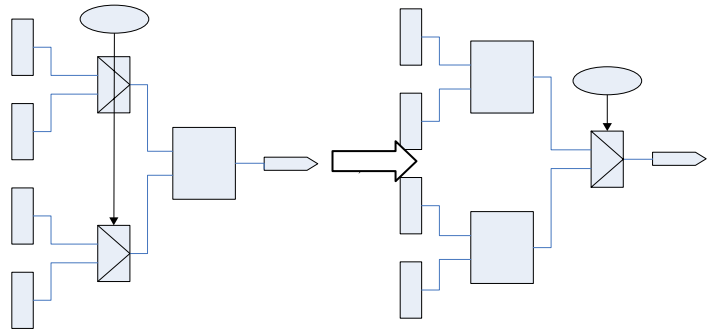
Logic Toggling Activity Reduction

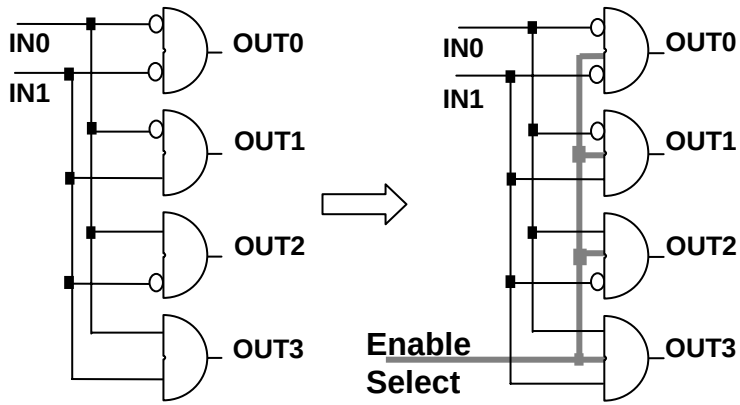
- Goals
  - **Prevention**
    - **Avoid unnecessary logic toggling**
  - **Cure**
    - **Filtering unnecessary logic toggling**
    - **Limit the propagation of unnecessary signals**
    - **Spread the toggling over time**
- Helps Peak Power Reduction

Glitches re-arranging the logic

- Logic can be re-arranged to move the glitch downstream
- By moving the glitch upstream helps to consume much less power
- Decoder outputs are generally heavily loaded
  - **Enable/Select signal prevents the propagation of their switching activity when the decoder is not in use**

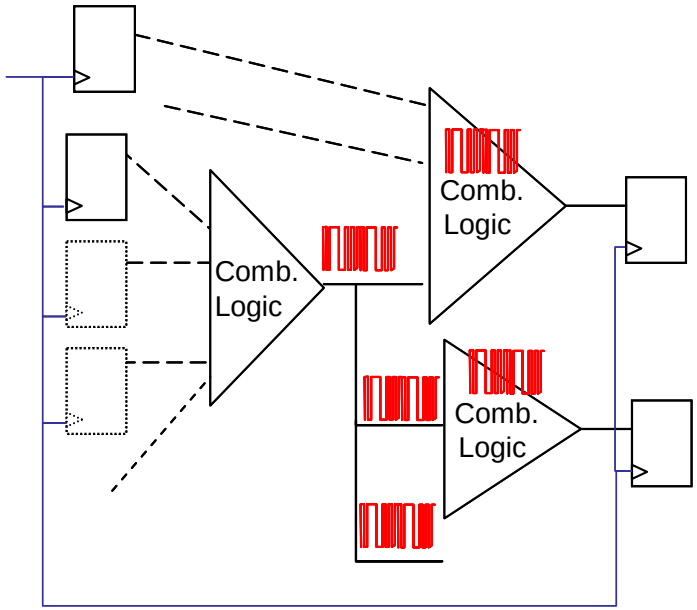


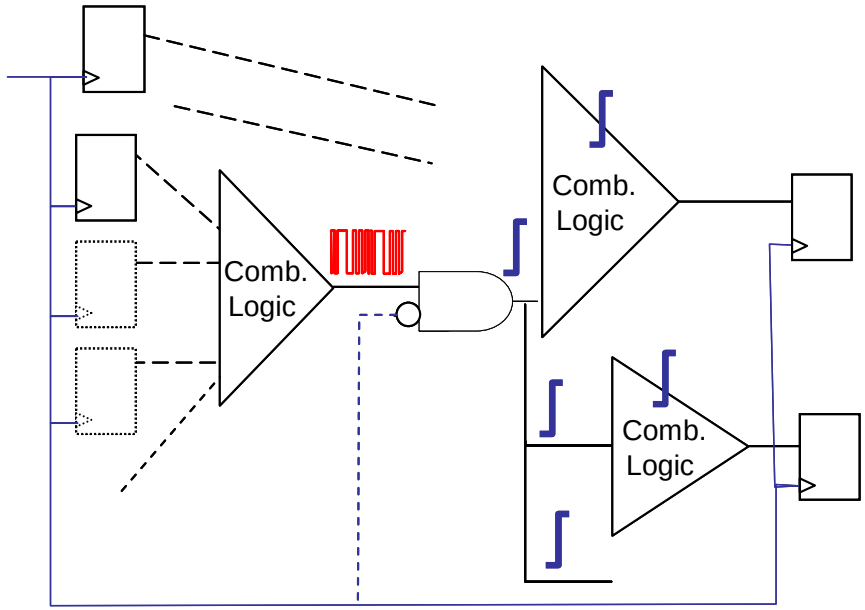




Easy Way if ....

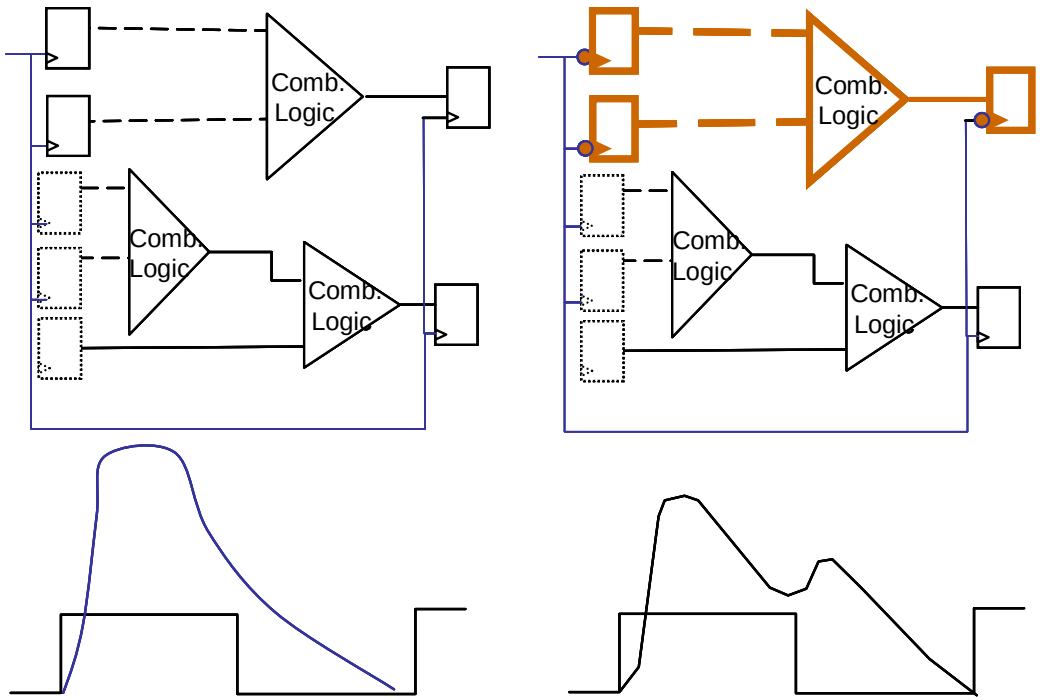
- Enough Slack to add one gate (and its routing) even for single cycle paths
- Notice opposite edge clock input of the gate





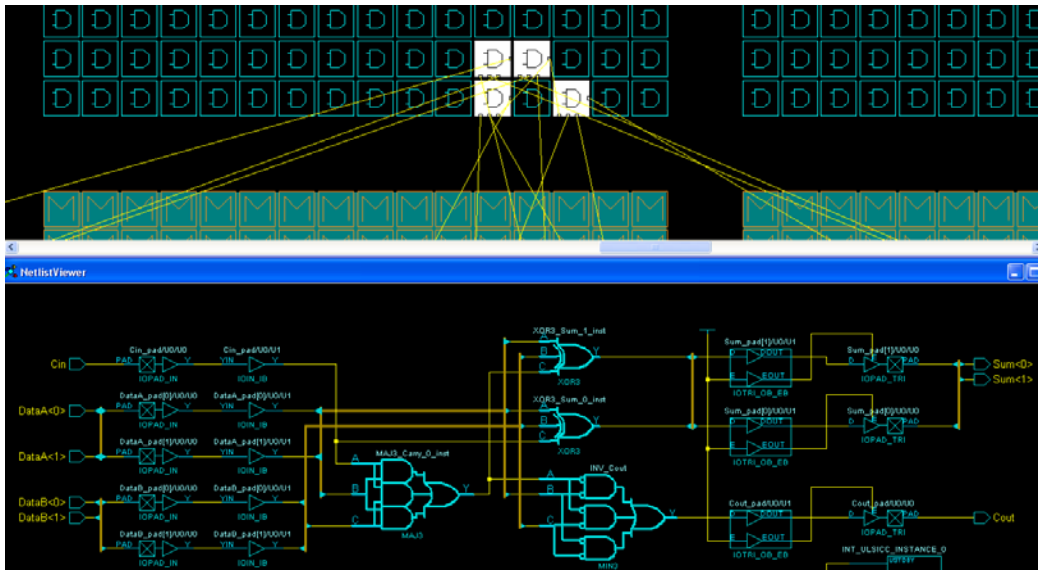
Glitches and Peak Power

- In some cases, it is possible to spread the toggling activity over time
- Clocking change (if possible) could lead to lower peak power



Use logic Floorplanning

- Use minimum wire length for local nets
  - **Place the logic adjacent to each other, the layout tool is able to use the ultra-fast local routing**
- Should be only after power driven layout for additional power reduction



Glitch Reduction Methodology

- Steps to follow
  - **Identify highly-toggling nets and their drivers**
  - **Estimate worst-case timing for the inputs of the driver cells**
  - **Push the source of the glitching up or down in the logic**
  - **Re-estimate the worst-case timing for the inputs of the driver cells**
  - **Insertion, after the driver, of a register clocked with opposite or direct edge of the clock depending on the previous timing analysis**
- Easier to implement for multi-cycle and false paths
- Elsewhere, requires STA scripts change

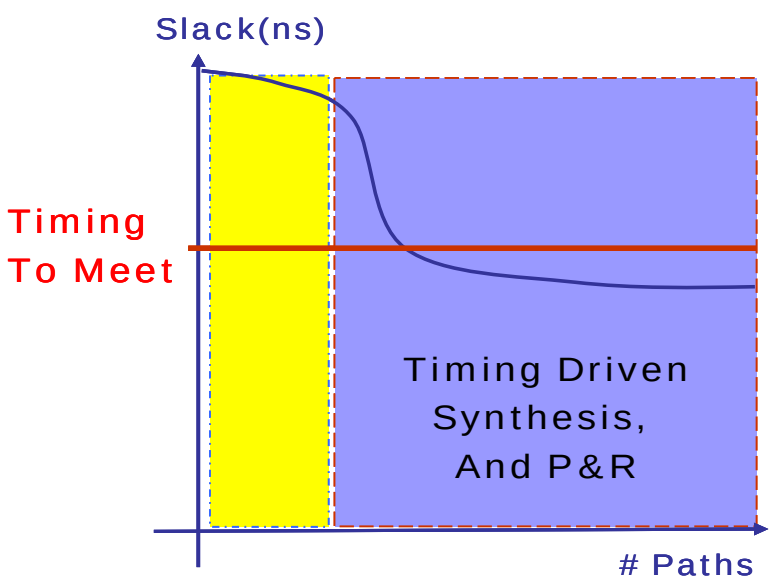
Summary Tables

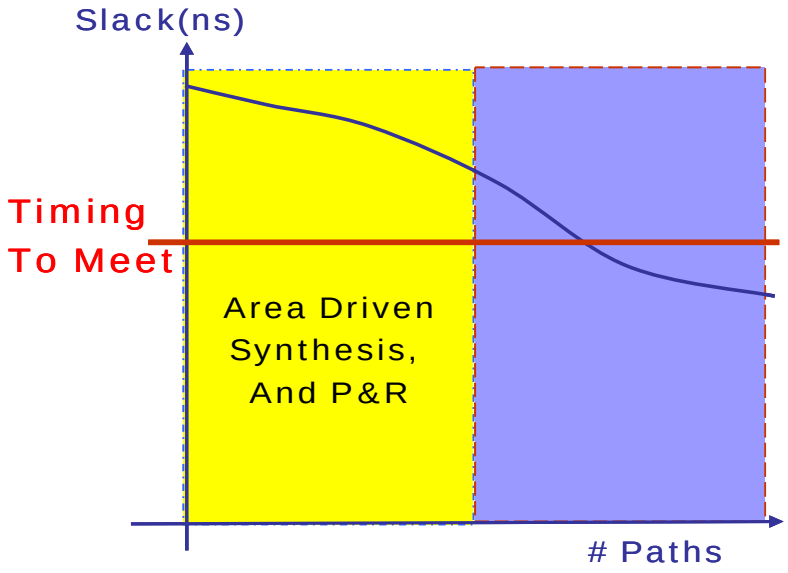
- Logic Power

Design Option	Premises and Alternatives	Estimate Savings
Area oriented synthesis	STA shows enough positive slack	Could be substantial-design dependent
Power friendly arithmetic blocks	Timing and area attributes of blocks required	“5%” to “15%”
Power friendly counters	If only final count is used, Binary counters are best. If counters used to drive bus, Gray are best. If many counter value need decoding, Ring counters are best	Depends on number and size of counters as well as use model
State machine Encoding	Gray uses less transition, but it is always possible to encode all states	Depends on the number of states and output requires
RTL changes for glitch reduction	Using pipeline increase latency	“5%”
	Insert AND gate driven by opposite clock edge. Require minor positive slack in glitch paths	Depends on size of down stream logic and glitch activity
	Insertion of Latches driven by opposite clock edge. Requires STA	Depends on size of downstream logic and glitch
RTL changes for switching activity	Need to validate	Depends on the switching signals
PDPR	Combine with TDPR	“5%” to “12%”

Power-Aware Tools

- Pay attention to the synthesis process and don't use stringent global timing constraints
- Study the so-called "Slack Distribution" for each clock domain
- Use careful area-oriented synthesis for the blocks that have relaxed timing or have all internal paths with enough margins





Synthesis Options and Switches

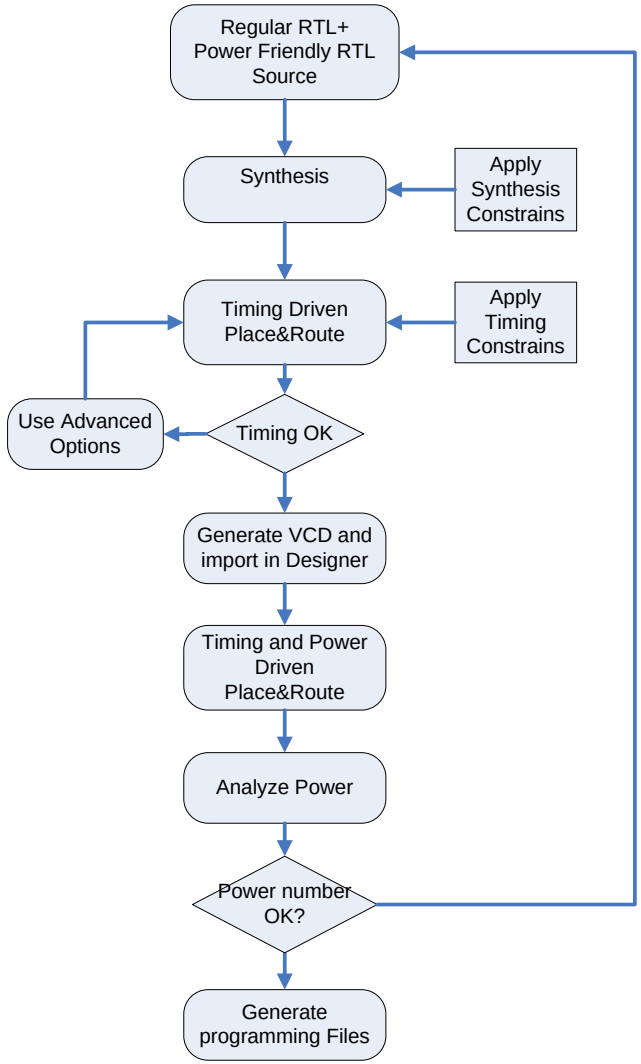
- Synthesis options and switches
  - **Watch out for global timing settings**
  - **Watch out for set_max_fanout**
  - **Area vs. Timing Driven**
    - **Analyze Design Slack Distribution**
    - **Use Area Oriented Synthesis for Paths with Significant Timing Margin**
  - **Check the Resource Sharing option**
    - **Synthesis will try to shares hardware resources like adders, multipliers, and counters wherever possible, and minimizes area**
  - **For designs with large state machine, follow the guideline from the above dynamic power reduction techniques and write the RTL directly into the intended encoding**
  - **For small designs, you might improve area by using the 'syn_hier' attribute with a value of 'flatten'**

Designer Options and Switches

- Run Power Driven Layout (PDPR) in addition to Timing Driven Layout
  - **Clock tree reduction by using fewer spines and rows**
  - **Net Capacitance reduction based on estimated activities**
  - **Power-driven placement also work with activities estimation from timing constraints and average toggle rates**
- To get the most out of Power Driven Layout:
  - **Use the VCD files from post-layout simulation**
  - **Enter maximum delay, minimum delay, setup, and hold constraints in SmartTime's constraint editor or in SDC**
  - **Set false paths on any paths that have a constraint, but do not need one (this will help layout meet the constraints that are needed)**
- SmartPower
  - **Create Power profiles based on functional modes**
  - **Cycle-accurate analysis**
  - **Spurious transitions analysis**
  - **Battery life estimation tool**
  - **Enable Variable Voltage use modes**

Default Low Power Design Flow

- The RTL changes as far as timing allows
- Perform Functional and Timing Validation of Changes
- Apply the Synthesis Hints
- Use Power-Driven Place and Route
- Apply the Floorplan and Place and Route Hints
- Validate ... Validate ... Validate!!!



Conclusion

- Power is Limiting Factor Affecting Performance and Features in Most Important Products
- Work as a team, and Deploy efforts and time
- Focus on the early stage

"Power Design Techniques Presented Along with Actel's Low Power FPGAs Produces the Lowest Power Designs"