DELPHI

http://www.ece.cmu.edu/calcm/delphi

A Framework for RTL-Based Architecture Design Evaluation Using DSENT Models

Michael K. Papamichael*, Cagla Cakir*, Chen Sun†, Chia-Hsin Owen Chen†, James C. Hoe*, Ken Mai*, Li-Shiuan Peh†, Vladimir Stojanovic‡

*Carnegie Mellon University †Massachusetts Institute of Technology ‡UC Berkeley

Computer Architecture Lab at Carnegie Mellon

ISPASS, March 30, 2015

Architecture Design Evaluation

Traditionally: Run cycle-level simulation

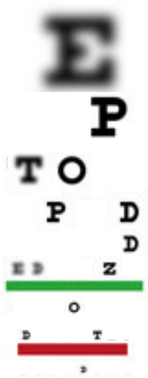
- Fast, flexible, easy to use & extend
- But what about **low-level HW implementation artifacts?**

Architecture Design Evaluation

Traditionally: Run cycle-level simulation

- Fast, flexible, easy to use & extend
- But what about **low-level HW implementation artifacts?**

❶ Recently: Use custom modeling tools, e.g.:



- Processors: McPAT, Wattch
- Memories: CACTI, DRAMSim
- Networks-on-Chip (NoCs): Orion, DSENT

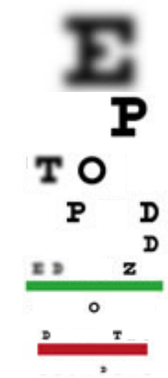
Fast and easy, but limited scope and/or fidelity

Architecture Design Evaluation

Traditionally: Run cycle-level simulation

- Fast, flexible, easy to use & extend
- But what about **low-level HW implementation artifacts?**

❶ Recently: Use custom modeling tools, e.g.:



- Processors: McPAT, Wattch
- Memories: CACTI, DRAMSim
- Networks-on-Chip (NoCs): Orion, DSENT

Fast and easy, but limited scope and/or fidelity

❷ Even more recently: Take RTL through ASIC flow

Increased Interest in RTL Evaluation

IP blocks whenever it is possible. We evaluate the power and area performance of our MGUARD by integra

ID. To calculate the latency overhead of the sub-block selection, we implemented the tag match and the sub-selection logic in Verilog, synthesized in 45nm and scaled to 32nm [20]. The sub-

To estimate the area, power con frequency of the hardware modu

We evaluate the area and power overheads of the proposed architecture by implementing it in Verilog HDL and synthesizing the design using Compiler [28] with FreePDK [29 then scaled to 22nm (relevant pa ble 3). Instruction and data men

Manufacturing Company (TSMC) and 2 GHz clock frequency. In the e

We synthesized[1] the cache lookup logic using Synopsys and quantify the area, latency, and energy penalty. *Amoeba-Cache* is compat-

[1] We do not have access to an industry-grade 32nm library, so we synthesized at a higher 180nm node size and scaled the results to 32 nm (latency and energy scaled proportional to Vdd (taken from [36]) and Vdd^2 respectively).

RTL Implementation We have implemented the SGMF all unit types and interconnect) to power, area and timing. The design Synopsys toolchain and a commercial 65*nm* cell library, and the results were then extrapolated for a 40*nm* process. Based on this process, we determine that

of the tile architecture and the routing

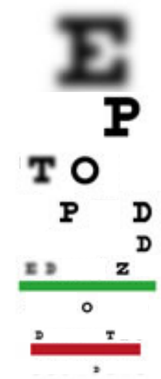
4.4.3. Controller Power & Area To analyze the impact of the controller on the area and power overheads, we synthesized the controller design in an industrial 65nm process. ... ... power estimates which we then scaled to the 32nm target technology node. The synthesized design includes

For the hardware evaluation part, we implemented the baseline ert mechanism (all 32 invari- ware Description Language nesized using Synopsys De- TSMC libraries at 1 V oper- cy. The results were used to alysis and evaluate the over-

Architecture Design Evaluation

Traditionally: Run cycle-level simulation

- Fast, flexible, easy to use & extend
- But what about **low-level HW implementation artifacts?**



❶ Recently: Use custom modeling tools, e.g.:

- Processors: McPAT, Wattch
- Memories: CACTI, DRAMSim
- Networks-on-Chip (NoCs): Orion, DSENT

Fast and easy, but limited scope and/or fidelity

❷ Even more recently: Take RTL through ASIC flow

Higher fidelity, accuracy, generality, but

- **Hard** – complex error-prone, requires expert knowledge
- **Tedious** – Costly, limited, hard-to-get tools and libraries
- **Slow** – tens of minutes to multiple hours per evaluation

Architecture Design Evaluation

Traditionally: Run cycle-level simulation

- Fast, flexible, easy to use & extend
- But what about **low-level HW implementation artifacts?**

1 **Recently: Use custom modeling tools, e.g.:**

- Processors: McPAT, Wattch
- Memories: CACTI, DRAMSim
- Networks-on-Chip (NoCs): Orion, DSENT

Fast and easy, but **limited scope and/or fidelity**

2 **Even more recently: Take RTL through ASIC flow**

Higher fidelity, accuracy, generality, but

- **Hard** – complex error-prone, requires expert knowledge
- **Tedious** – Costly, limited, hard-to-get tools and libraries
- **Slow** – tens of minutes to multiple hours per evaluation

Can we get the best of both worlds?

Architecture Design Evaluation

Traditionally: Run cycle-level simulation

- Fast, flexible, easy to use & extend
- But what about **low-level HW implementation artifacts?**

1 Recently: Use custom modeling tools, e.g.:

- Processors: McPAT, Wattch
- Memories: CACTI, DRAMSim
- Network-on-Chip (NoC): DSENT

Fast and easy, but limited scope and/or fidelity

2 Even more recently: Take RTL through ASIC flow

Higher fidelity, accuracy, generality, but

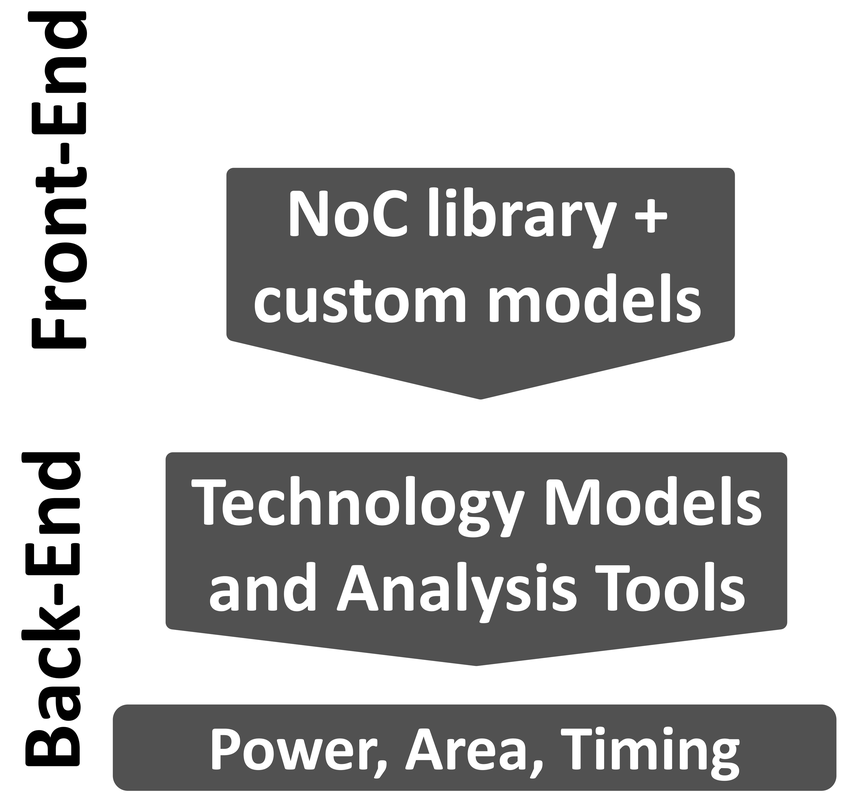
- **Hard** – complex error-prone, requires expert knowledge
- **Tedious** – Costly, limited, hard-to-get tools and libraries
- **Slow** – tens of minutes to multiple hours per evaluation

DELPHI combines DSENT + ASIC flow

Can we get the best of both worlds?

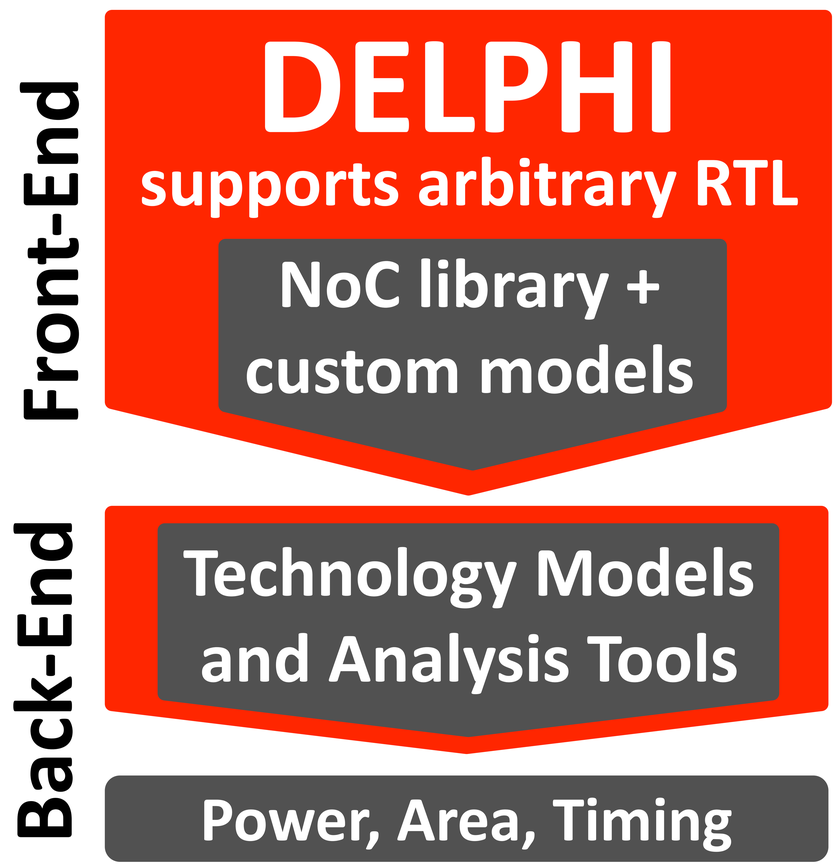
A Closer Look at DSENT

- **Originally developed for NoC evaluation. Has two parts:**
- **Front-end: <u>User-defined design models</u>**
  - Limited to NoC-specific components
  - Hard and tedious to build new models
    - Equivalent to "synthesis by hand"
- **Back-end: <u>Technology models and tools</u>**
  - Very powerful
    - **Fast** – characterize circuit in seconds
    - **High fidelity** – model at transistor level
    - **Flexible** – target many technology models



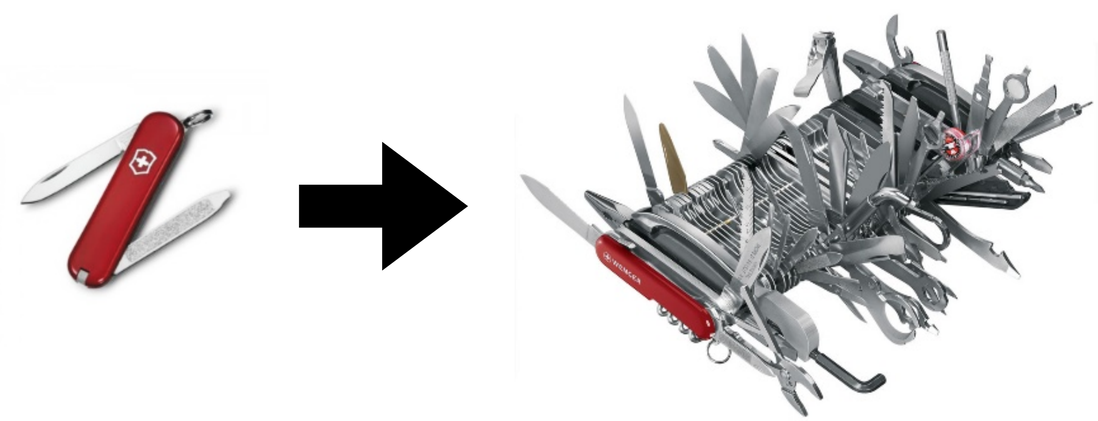
A Closer Look at DSENT

- **Originally developed for NoC evaluation. Has two parts:**
- **Front-end: <u>User-defined design models</u>**
  - Limited to NoC-specific components
  - Hard and tedious to build new models
    - Equivalent to "synthesis by hand"
- **Back-end: <u>Technology models and tools</u>**
  - Very powerful
    - **Fast** – characterize circuit in seconds
    - **High fidelity** – model at transistor level
    - **Flexible** – target many technology models



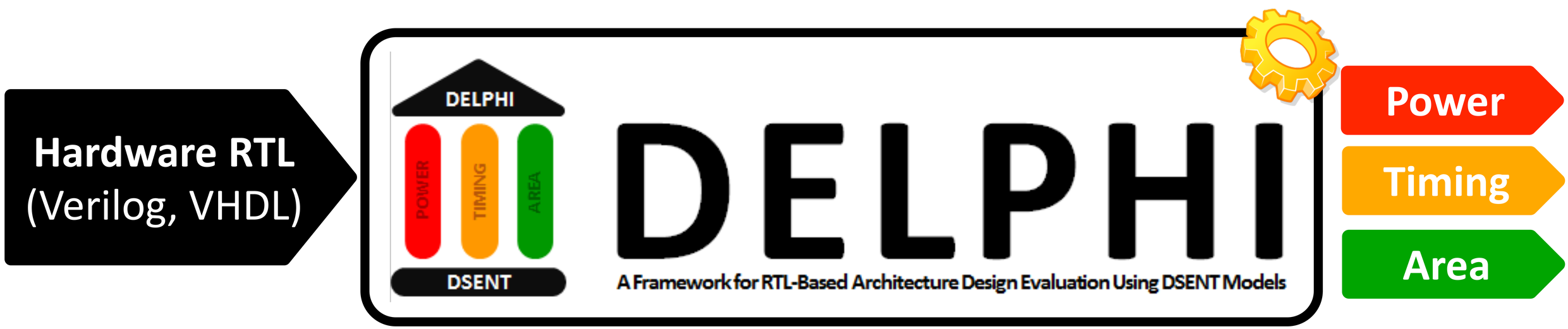
DELPHI extends front-end to support arbitrary RTL

- Generates DSENT design models
- Leverages modeling back-end
- Result is a powerful tool



The DELPHI RTL Characterization Framework

- **Fast RTL-Based Architecture Design Evaluation**



- **First-order power, area, frequency results**
  - Comparable to conventional RTL synthesis, but
  - **Faster** – sweep across multiple technology models in seconds
  - **Simpler** – provides tools and scripts for automated flow
  - **More flexible** – allows easy integration with SW
- **Leverages DSENT electrical modeling engine**
  - Use library of existing or custom technology models
- **http://www.ece.cmu.edu/calcm/delphi**

Outline

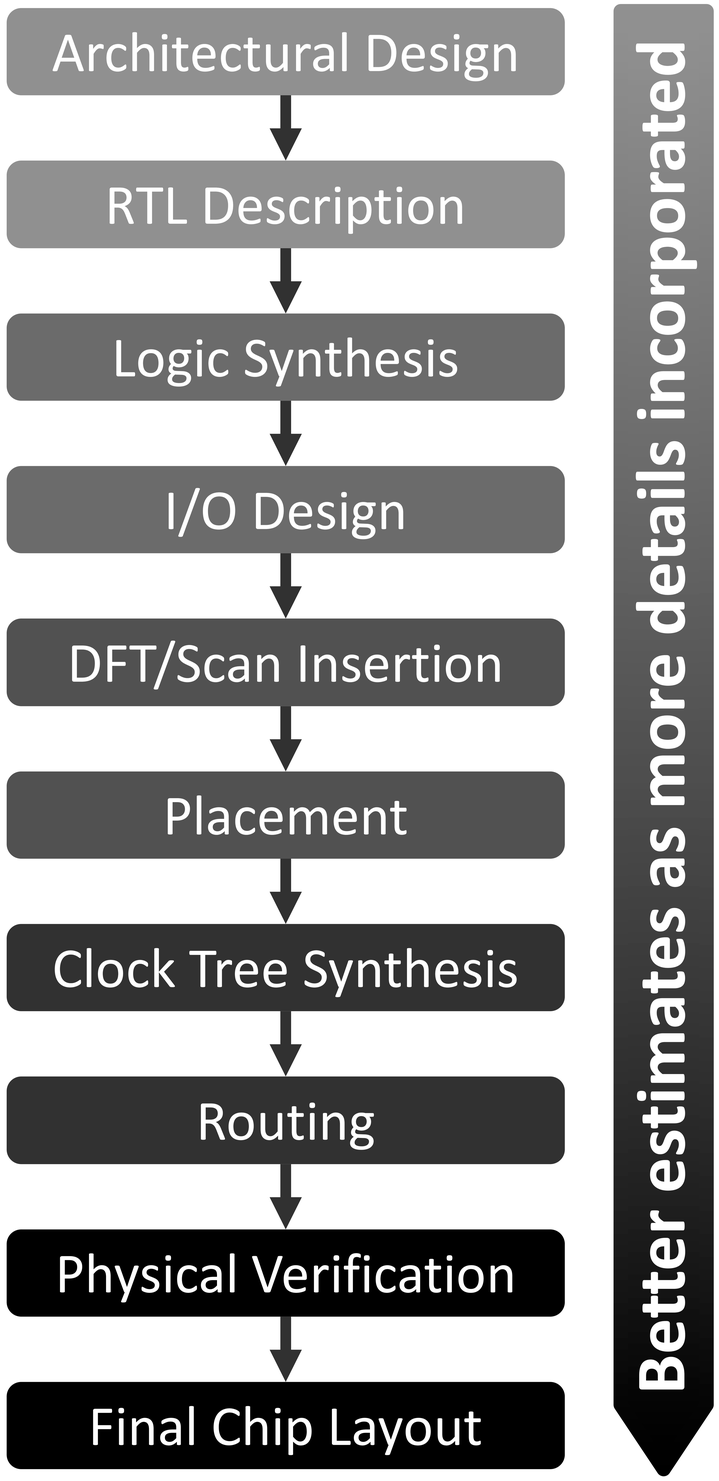
- Background: ASIC Flows and Synthesis
- The DELPHI Flow
- Evaluation
- Conclusion

Outline

- **Background: ASIC Flows and Synthesis**
- The DELPHI Flow
- Evaluation
- Conclusion

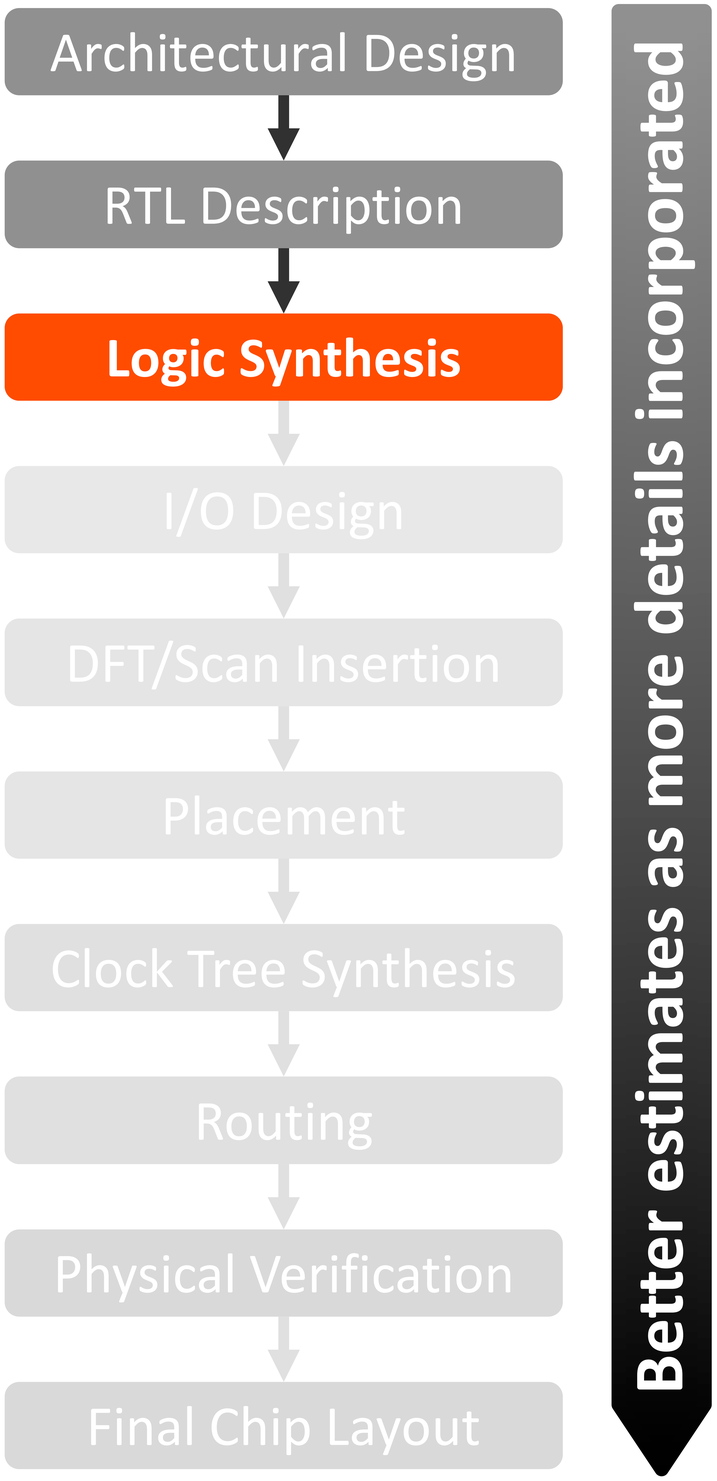
ASIC Flows and Synthesis

- **ASIC flow is multi-step process**
  - Starts with design Register-Transfer Level (RTL) description (e.g., Verilog or VHDL)
  - Ends with HW implementation that corresponds to detailed chip layout
  - Better power, area, frequency estimates as more details are incorporated



ASIC Flows and Synthesis

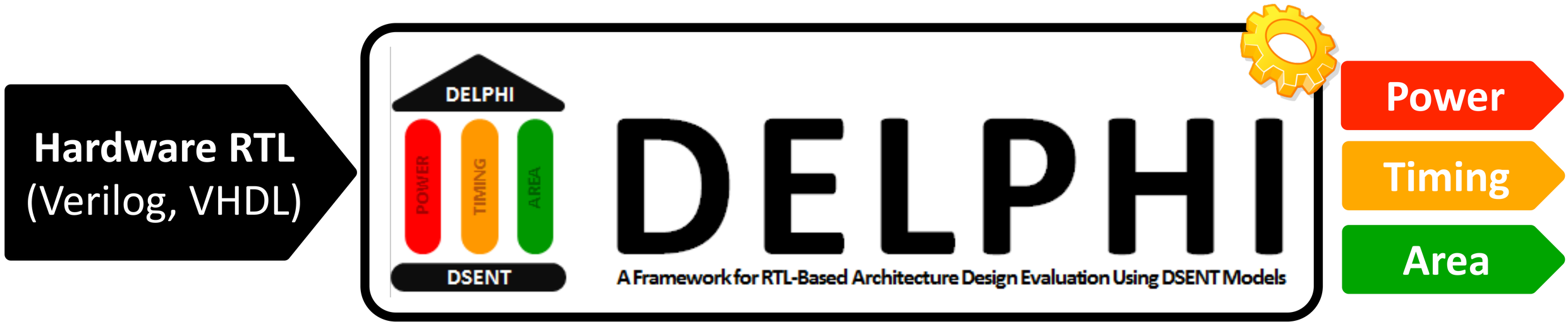
- **ASIC flow is multi-step process**
  - Starts with design Register-Transfer Level (RTL) description (e.g., Verilog or VHDL)
  - Ends with HW implementation that corresponds to detailed chip layout
  - Better power, area, frequency estimates as more details are incorporated
- **Architects often focus on synthesis**
  - Turns RTL description into fundamental building blocks (e.g., gates, registers), called **standard cells** in an ASIC flow
  - Resulting interconnected set of standard cells is known as (gate-level) **netlist**
  - Does not capture physical layout details



Outline

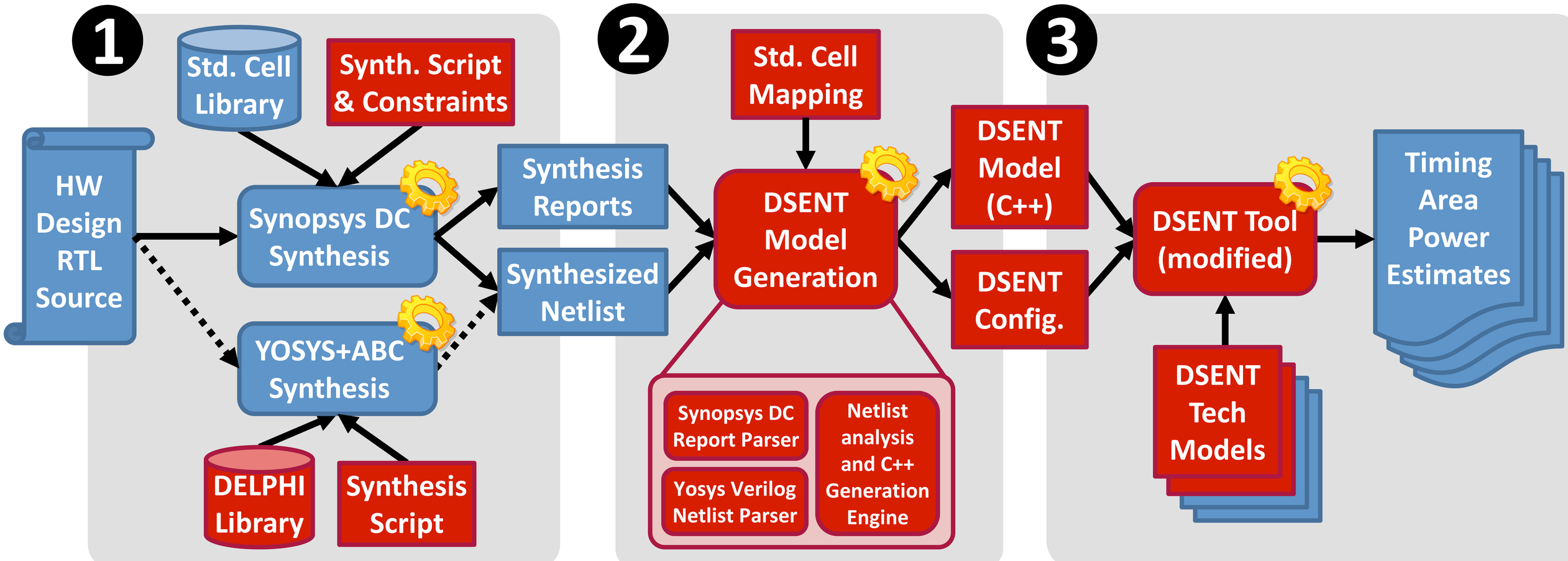
- Background: ASIC Flows and Synthesis
- **The DELPHI Flow**
- Evaluation
- Conclusion

The DELPHI Flow



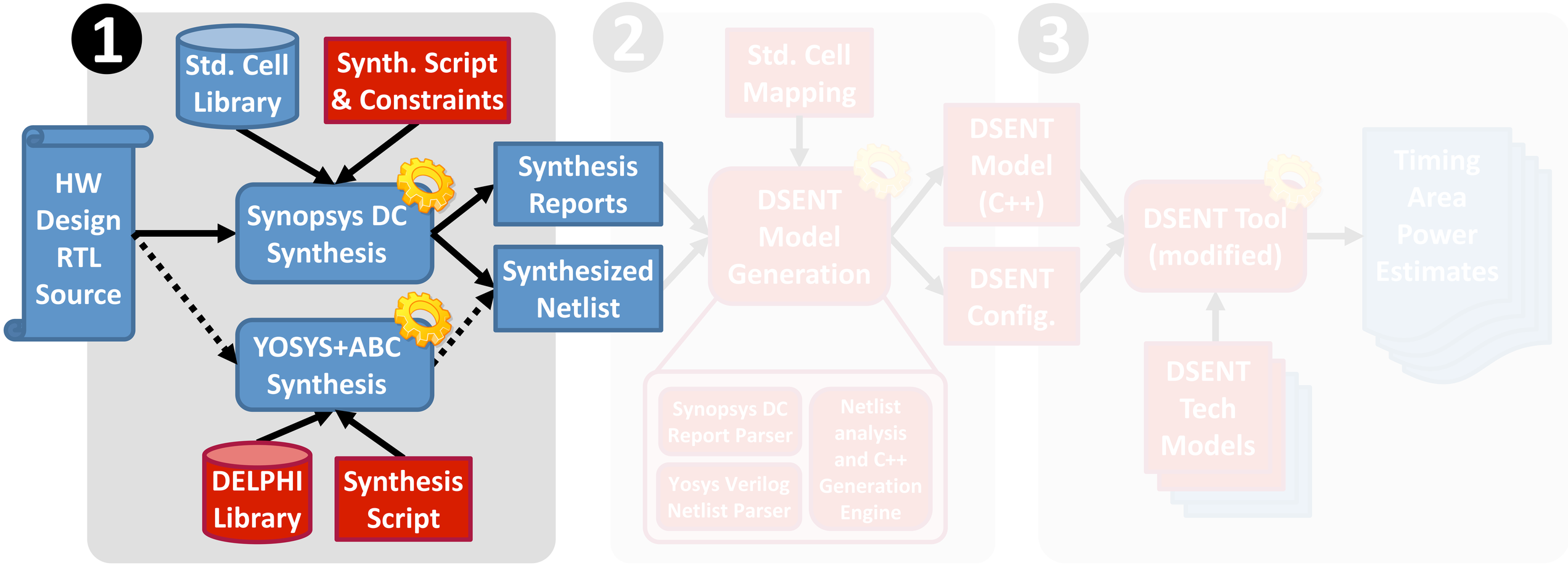
Red parts were developed or modified in support of the DELPHI flow

The DELPHI Flow



Red parts were developed or modified in support of the DELPHI flow

The DELPHI Flow

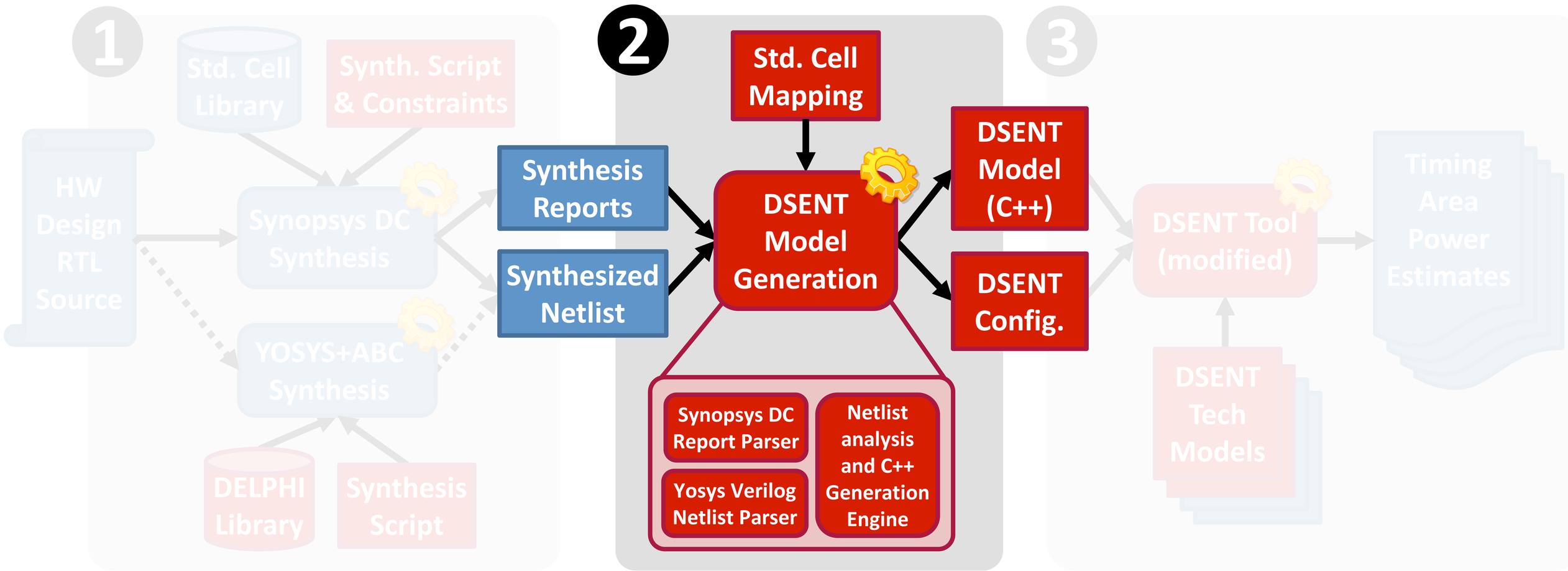


tens of minutes

❶ DELPHI Synthesis

- Supports commercial (Synopsys DC) and open-source flows

DELPHI Flow



under a minute

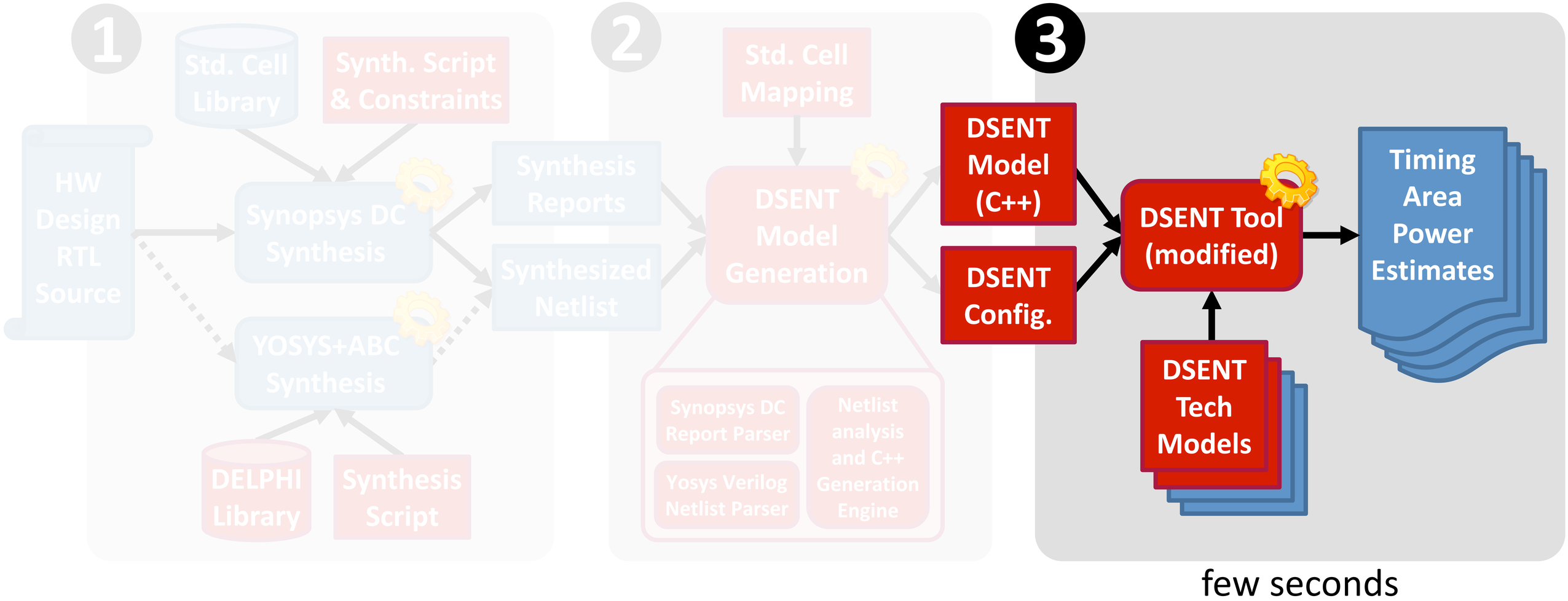
❶ DELPHI Synthesis

- Supports commercial (Synopsys DC) and open-source flow

❷ DSENT Model Generation

- Analyze netlist to produce DSENT C++ model and configuration

DELPHI Flow



few seconds

❶ DELPHI Synthesis

- Supports commercial (Synopsys DC) and open-source flow

❷ DSENT Model Generation

- Analyze netlist to produce DSENT C++ model and configuration

❸ Get power, area, timing estimates

- Target multiple technology models in seconds

Outline

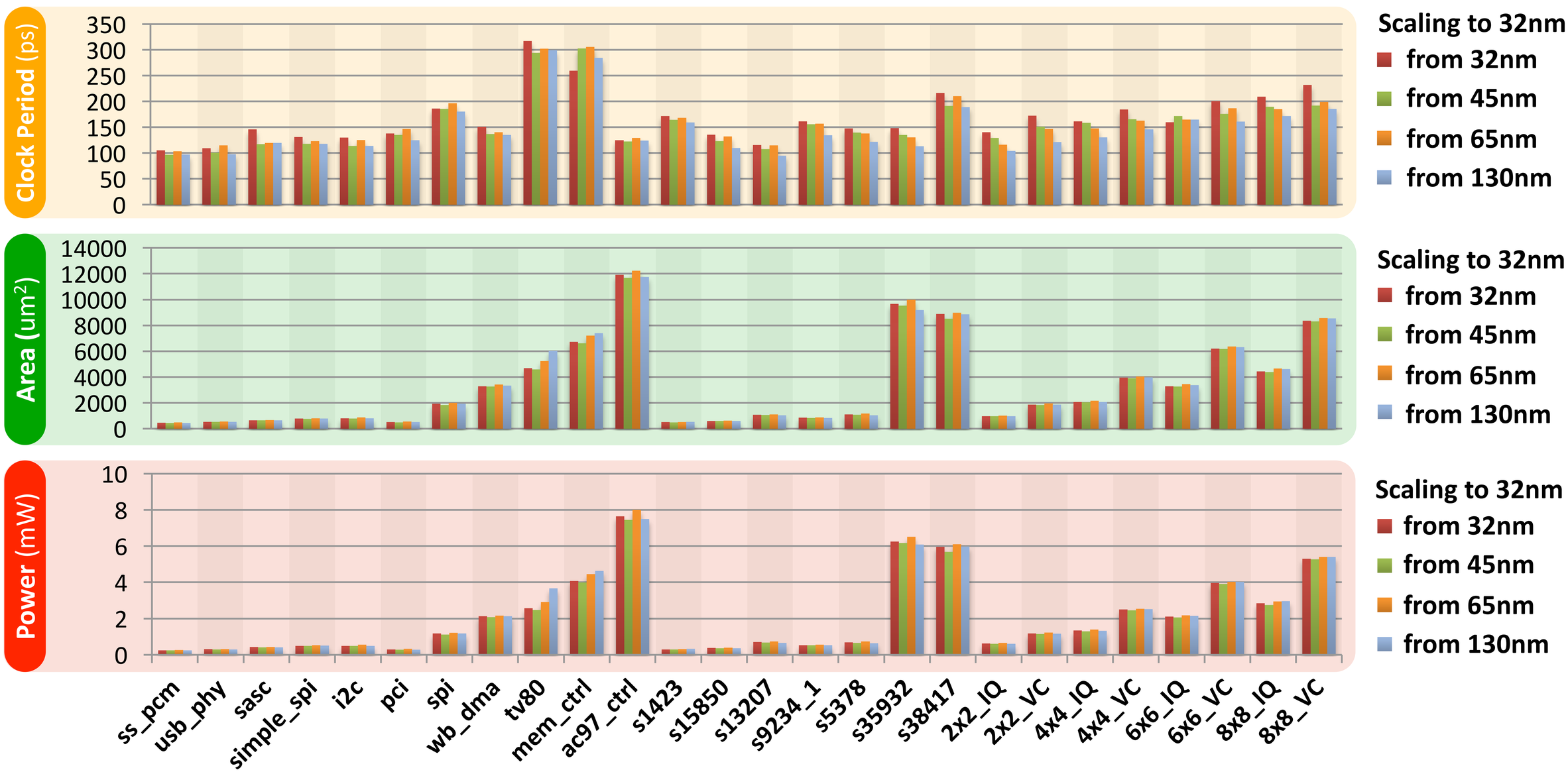
- Background: ASIC Flows and Synthesis
- The DELPHI Flow
- Evaluation
- Conclusion

Methodology

- **Evaluated across 24 HW RTL designs**
  - 18 designs from IWLS 2005 benchmark suite
  - 6 router designs using public CONNECT NoC generator
- **Synthesis results**
  - Synopsys Design Compiler (Version I-2013.12-SP3)
  - Targeting 4 commercial standard cell libraries from 3 vendors
    - 32nm, 45nm, 65nm, 130nm
  - For timing we try to maximize frequency
  - For power and area we fix frequency to 500MHz
- **DELPHI results**
  - Modified version of DSENT 0.91
  - Targeting 4 existing DSENT technology models
    - 11nm, 22nm, 32nm, 45nm

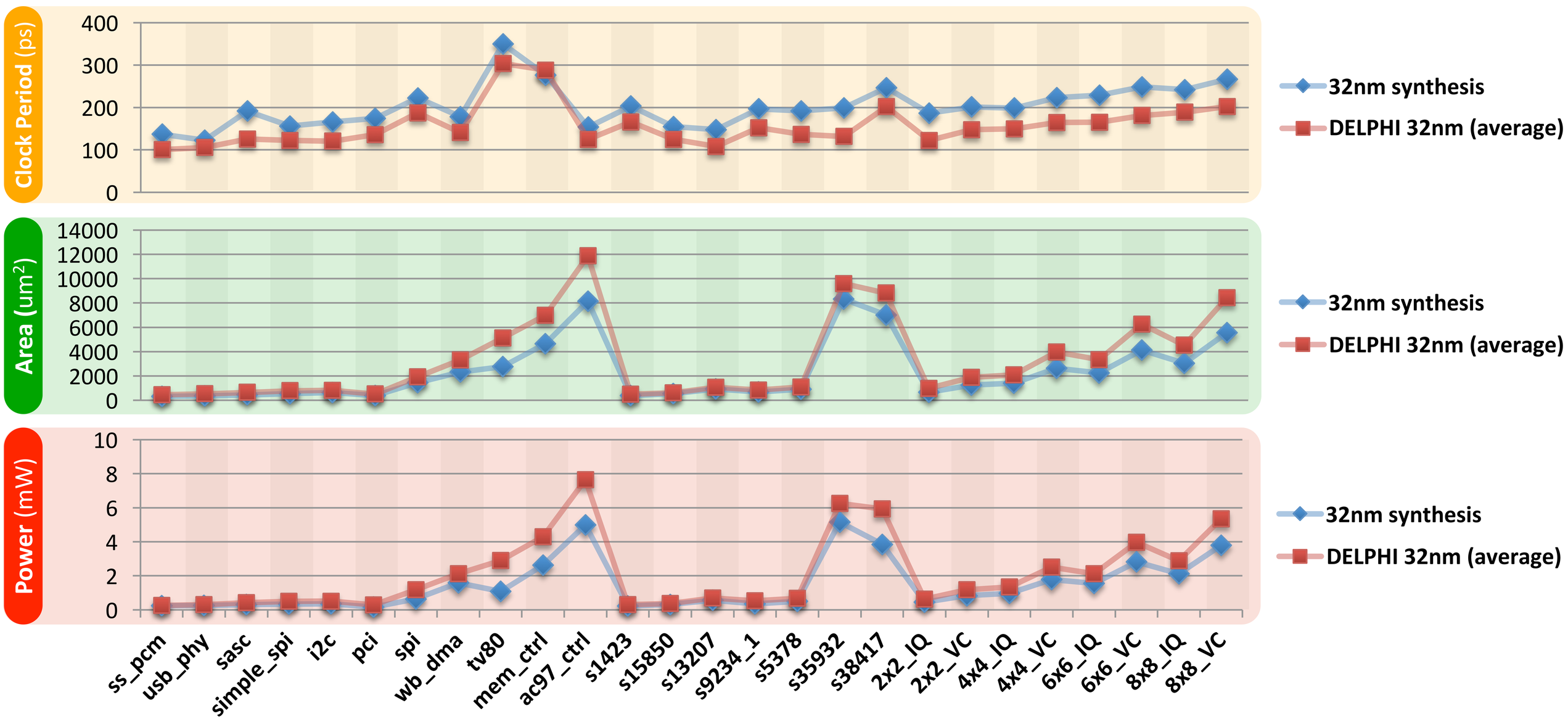
Scaling Robustness

- **Use DELPHI to scale to 32nm using synthesis results**
  - from 4 commercial std. cell libs (32nm, 45nm, 65nm, 130nm)
- **Results show low variance and behave consistently**



Capturing Trends

- **Juxtaposing power, timing, area trends of**
  - conventional synthesis results using 32nm commercial library
  - against average of DELPHI projections using a 32nm model

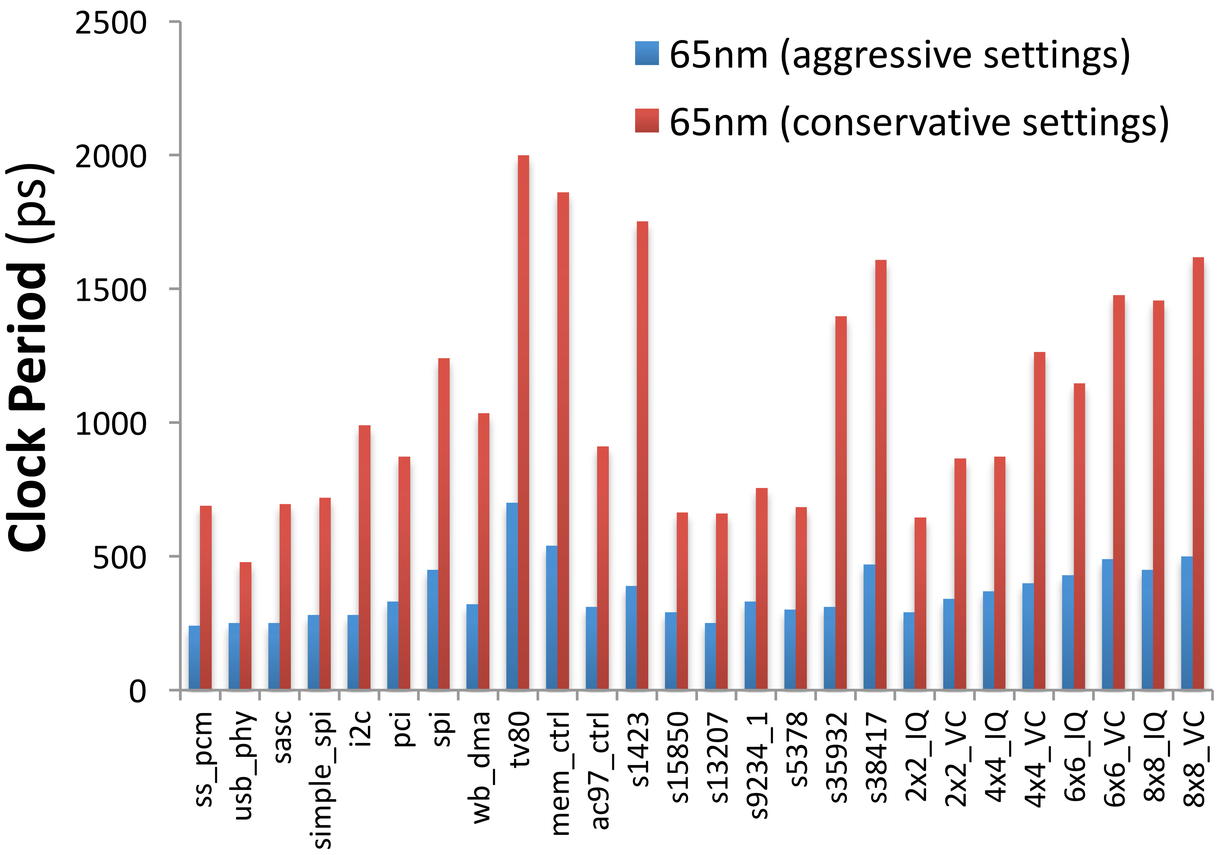


How much should we trust synthesis?

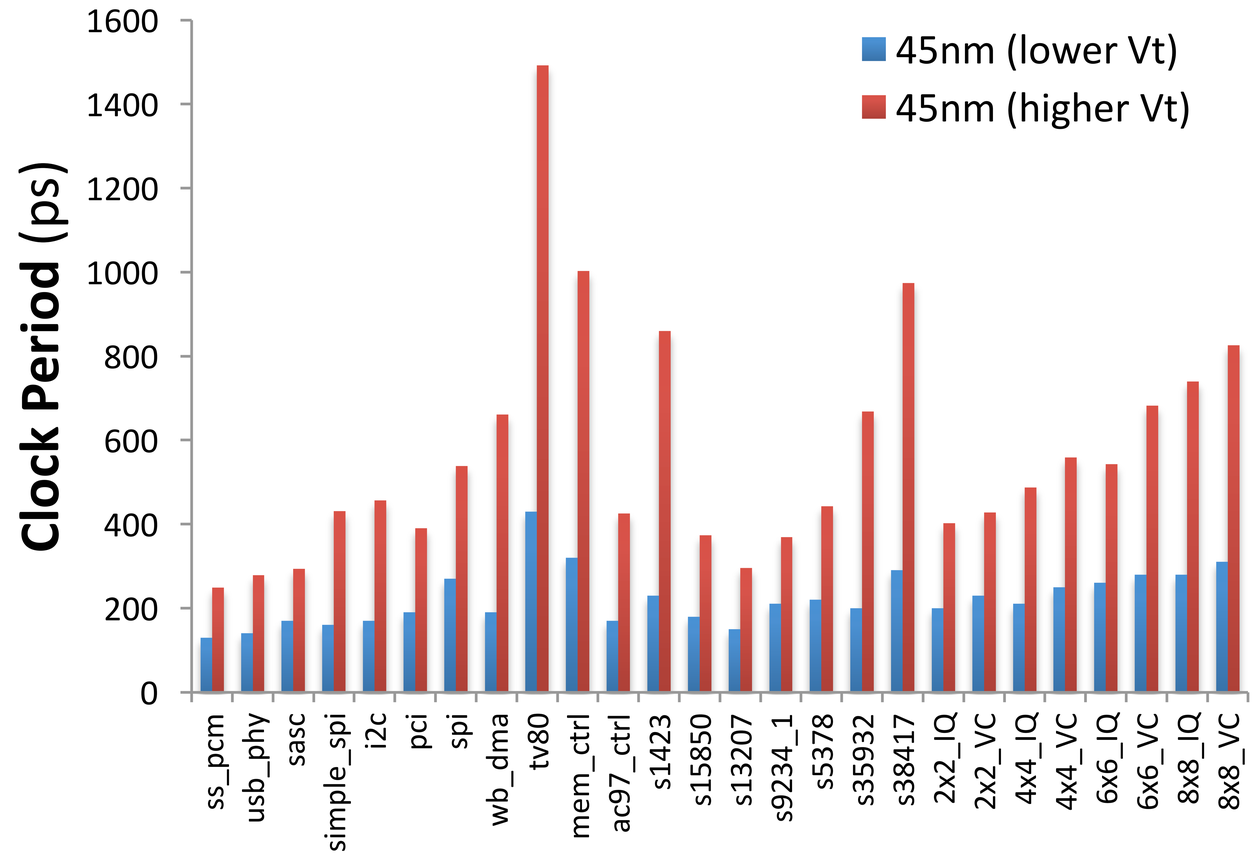
- **Take synthesis results with a grain of salt**
  - Can often deviate significantly from final product of ASIC flow
- **Modern synthesis tools are complex**
  - Easy to misuse or misconfigure
- **Factors that can affect synthesis results**
  - Lack of physical layout information
  - Optimization goals, constraints, and settings
  - Quality and features of synthesis tool
  - Variance across standard cell libraries

How much should we trust synthesis?

- Some examples of variance in synthesis results



Different synthesis settings



Different standard cell variants

How much should we trust synthesis?

- **Take synthesis results with a grain of salt**
  - Can often deviate significantly from final product of ASIC flow
- **Modern synthesis tools are complex**
  - Easy to misuse or misconfigure
- **Factors that can affect synthesis results**
  - Lack of physical layout information
  - Optimization goals, constraints, and settings
  - Quality and features of synthesis tool
  - Variance across standard cell libraries

1. **Synthesis only good for first-order characterization**
2. **You have to know what you are doing**

DELPHI Limitations

- **Only as accurate as synthesis**
  - Like synthesis, ignores or makes simplifying assumptions
- **Garbage in, garbage out**
  - Only as good as the technology models
- **Minimal set of standard cells**
  - Standard cell library not as rich as commercial libraries
- **Only supports a single clock domain**
  - Designs with multiple clocks need to be modeled piecewise
- **Large memories built out of flip-flops by default**
  - No “memory compiler” support. Incorporate CACTI models?
- **Limited to static power analysis**
  - Estimates based on probabilistic switching activity models

Outline

- Background: ASIC Flows and Synthesis
- The DELPHI Flow
- Evaluation
- **Conclusion**

Conclusions and Public Release

Conclusions

- **DELPHI: fast, simple, flexible RTL characterization flow**
  - Allows rapidly sweeping across multiple technology models
  - Compared to other models, DELPHI is a "meta-model"
- **With great power comes great responsibility**
  - Good to expose trends and get first-order results
  - Make sure to use the right tool for the right job

Public release

- **Full DELPHI flow (including modified DSENT)**
  - Supports commercial and open-source synthesis tools
- **http://www.ece.cmu.edu/calcm/delphi**
  - Contact us for pre-release version

Thanks!

Questions?