香港中文大學
The Chinese University of Hong Kong

CENG3430 Rapid Prototyping of Digital Systems

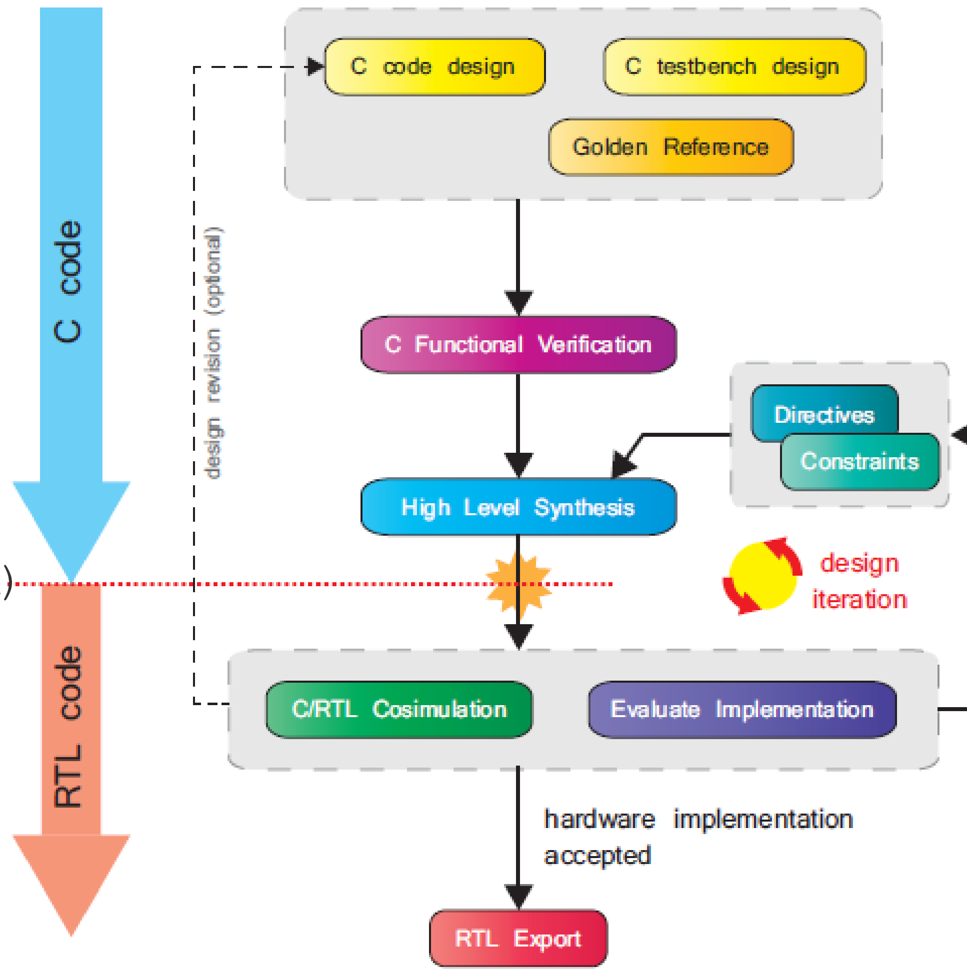
Lecture 09: Rapid Prototyping (III) – High Level Synthesis

Ming-Chang YANG

mcyang@cse.cuhk.edu.hk

Recall: What we have done in Lab10

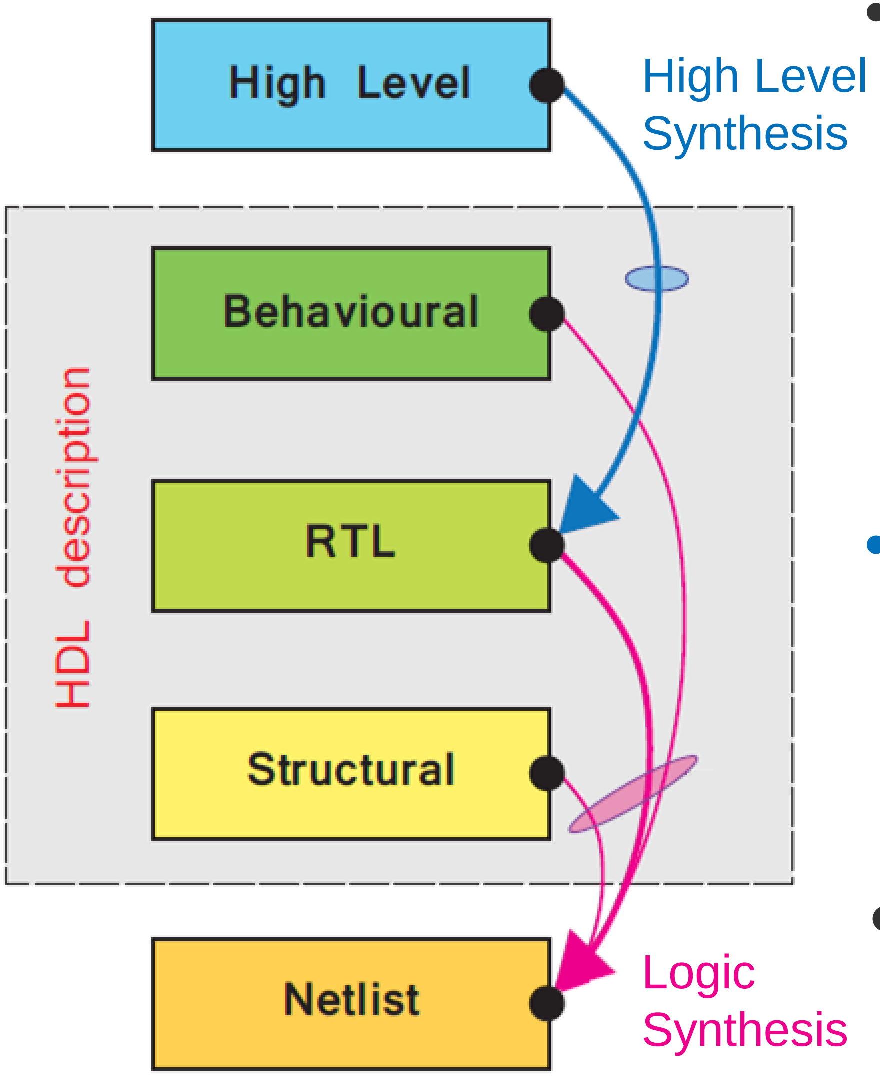
```
// function to be accelerated in HW
template <typename T, int DIM>
void mmult_hw(T a[DIM][DIM], T b[DIM][DIM], T out[DIM][DIM])
{
  int const FACTOR = DIM/2;

  // matrix multiplication
  L1: for(int ia=0; ia<DIM; ++ia)
    L2: for(int ib=0; ib<DIM; ++ib)
    {
      T sum = 0;
      L3: for(int id=0; id<DIM; ++id)
        sum += a[ia][id]*b[id][ib];
      out[ia][ib] = sum;
    }
  return;
}
```



Outline

- What is High Level Synthesis (HLS)?
- Why High Level Synthesis (HLS)?
- Design Metrics in HLS
- High-Level Synthesis Process
  - Scheduling
  - Binding
- Vivado High Level Synthesis
  - Algorithm Synthesis
  - Interface Synthesis
  - Inputs and Outputs

What is High Level Synthesis (HLS)?



- In FPGA design, “synthesis” usually refers to logic synthesis.
  - The process of analyzing, interpreting HDL code, and forming the netlist.
- High-level synthesis means synthesizing the high-level C, C++ or SystemC code into an HDL description.
- *Note: Both syntheses will be applied (one after the other) when designing a Vivado HLS.*

Why High Level Synthesis (HLS)?

- With a high-level representation abstracting low-level detail, the description of the circuit becomes simpler.
  - The result is that designs can be generated much more rapidly than using more traditional methods.

- HLS separates the functionality and implementation.
  - The source code *does not fix* the architecture.
    - Variations on the architecture can be created quickly by applying appropriate directives to the HLS process, rather than having to fundamentally rework the source code.

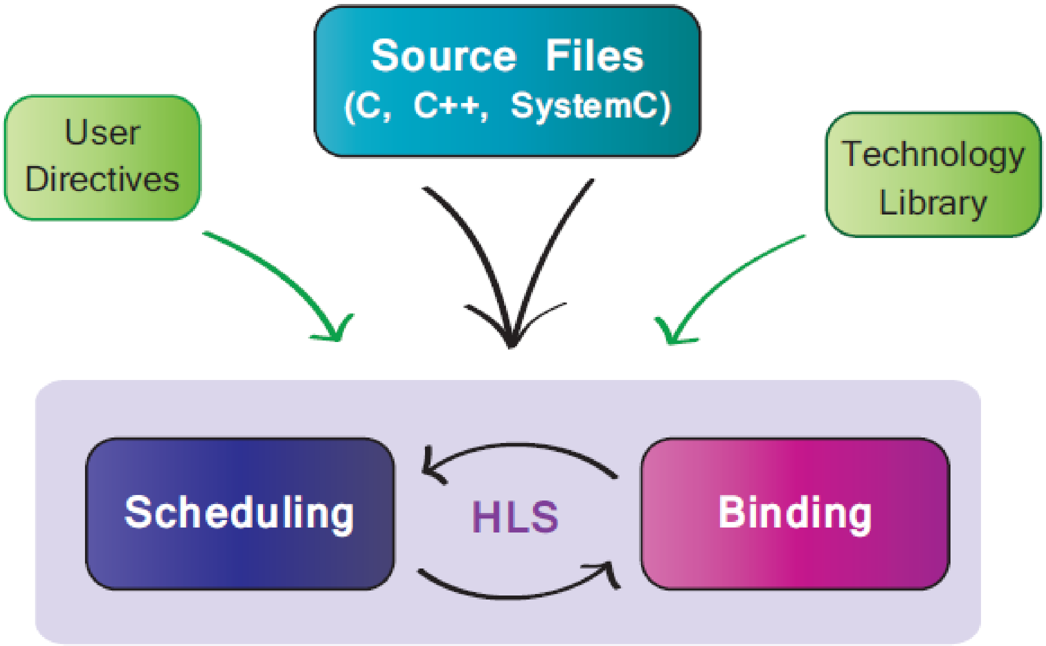
- HLS is also beneficial in system development and software/hardware partitioning.
  - There is a common language for targeting both software and hardware elements of the system.

Design Metrics in HLS

- As mentioned, the functional description and the implementation are separate in HLS.
  - As a result, the designer has the opportunity to evaluate possible architectures generated by the HLS process, and optimize according to different requirements.
- Hardware design has two fundamental metrics:
  1) **Area**, or **Resource Cost** — the amount of hardware required to realise the desired functionality;
  2) **Speed**, or specifically **throughput** or **latency** — the rate at which the circuit can process data.
- The designer is always faced with a trade-off between resource cost and speed.
  - This is the key aspect evaluated and optimized during HLS.

High-Level Synthesis Process (1/4)

- In general, HLS is comprised of two main processes:
  - **Scheduling** is the translation of the RTL statements interpreted from the C code into a set of operations, each with an associated duration in terms of clock cycles.
  - **Binding** is the process of associating the scheduled operations with the physical resources of the target device.

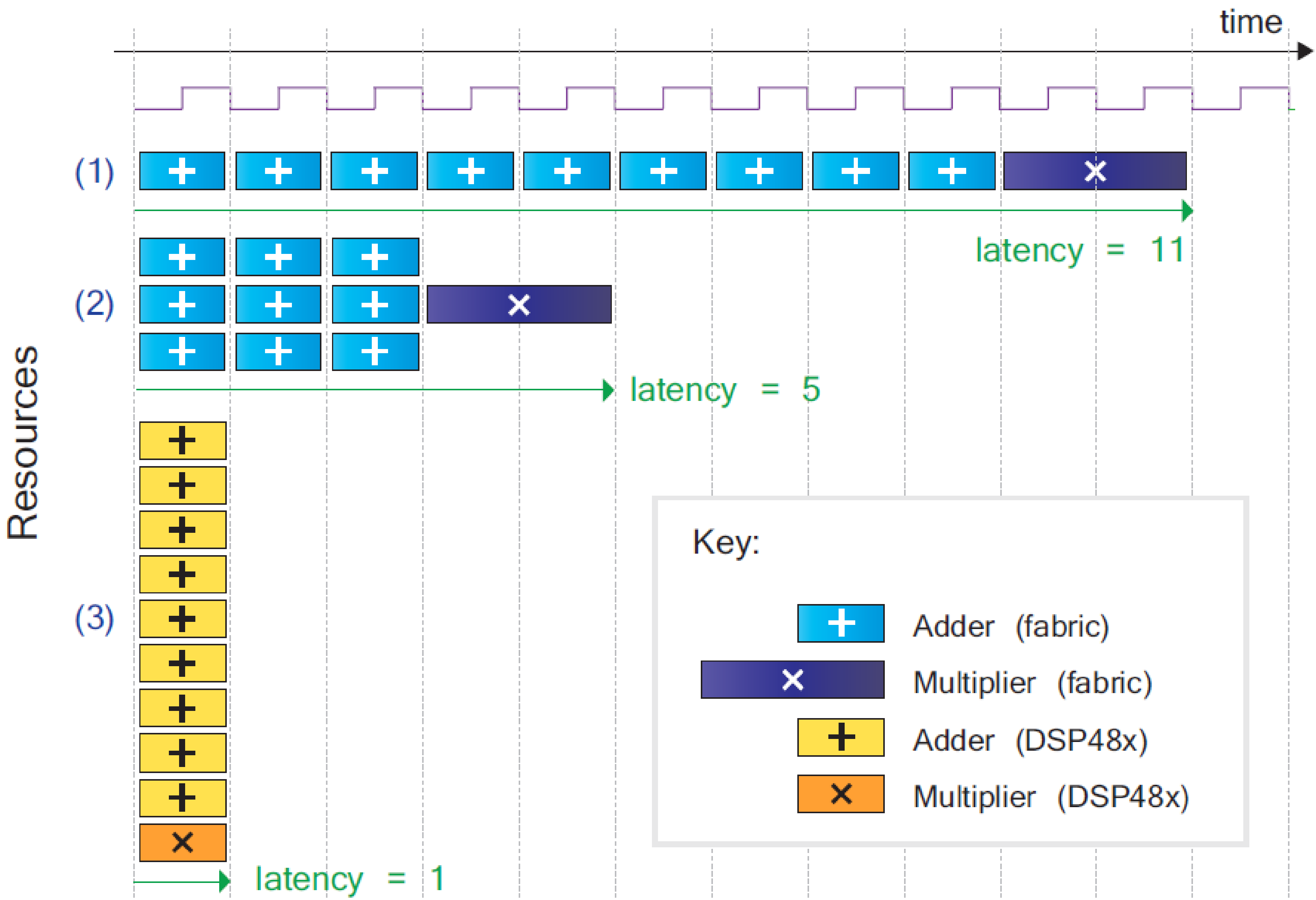


High-Level Synthesis Process (2/4)

- That is, the HLS process must decide
  1) How to schedule the operations (how many clock cycles to allocate to their completion), and
  2) How to bind the operations (i.e. how to map them to the computational resources on the PL).

- The resulting implementation has a set of metrics, principally in terms of (i) latency, (ii) throughput, and (iii) the amount of resources used.
  - For example, consider a C program involves calculating the average of an array input, consisting of ten numbers. The implied operations are:
    - 9 addition to find the total; followed by
    - A multiplication by 0.1 to calculate the average.

High-Level Synthesis Process (3/4)



High-Level Synthesis Process (4/4)

- By default, the HLS process will optimize area.
  - That is, it will adopt the first strategy, which consumes the fewest resources with long latency and low throughput.
- There are two methods available to drive the high-level synthesis process towards different goals.
  - **Constraints** — The designer places a limit on some aspect of the design.
    - For instance, the minimum clock period may be specified.
  - **Directives** — The designer can exert more specific influence over the RTL implementation via directives.
    - There are various types of directive available which map to certain features of the code, enabling the designer to dictate.
    - For example, how the HLS engine treats loops or arrays identified in the C code, or the latency of particular operations.

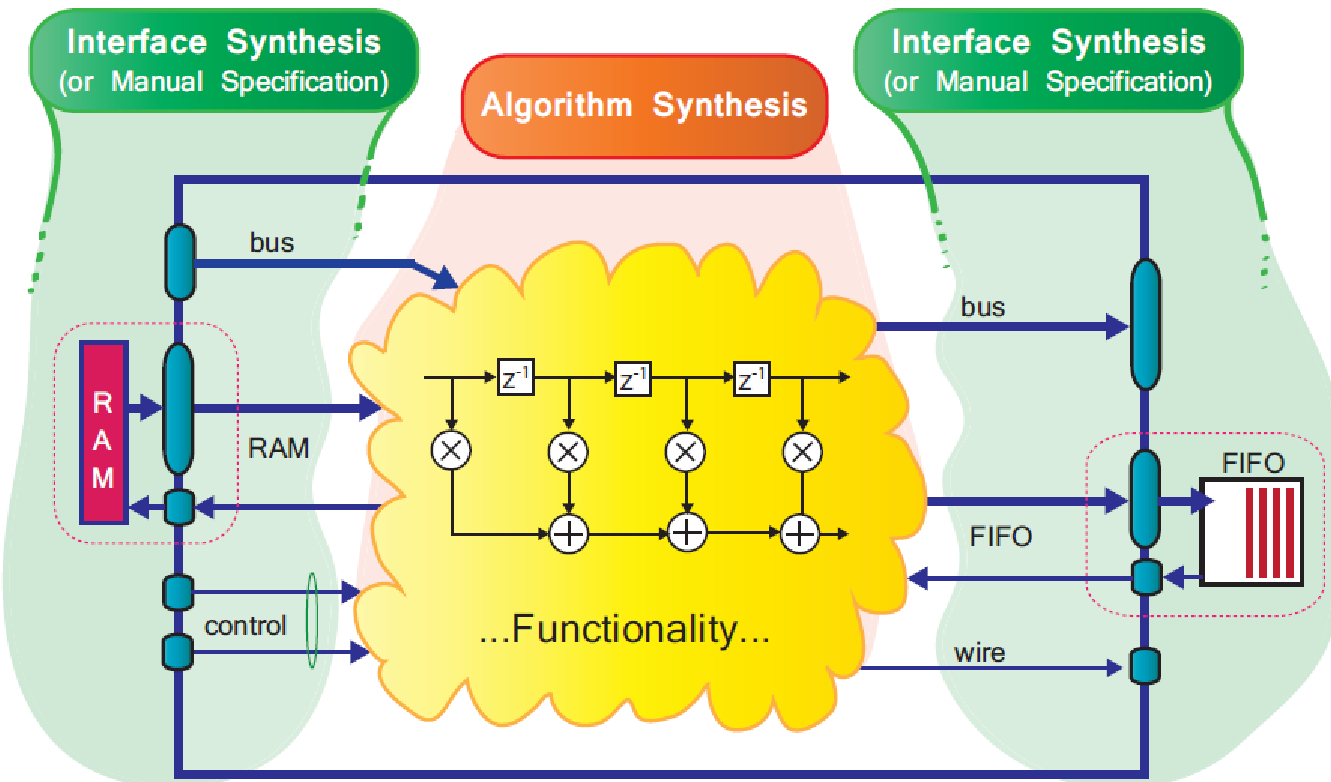
Vivado High Level Synthesis (1/2)

- Vivado High Level Synthesis
  1) First transforms (i.e., high-level synthesis) a C, C++ or SystemC design into an RTL implementation
  2) Then synthesizes and implements (i.e., logic synthesis) the RTL implementation onto the programmable logic of a Xilinx FPGA or Zynq device.

- In Vivado HLS design, the two primary aspects of the design are analyzed:
  – The ***interface*** of the design: its top-level connections;
  – The ***functionality*** of the design: the algorithm(s) that it implements.

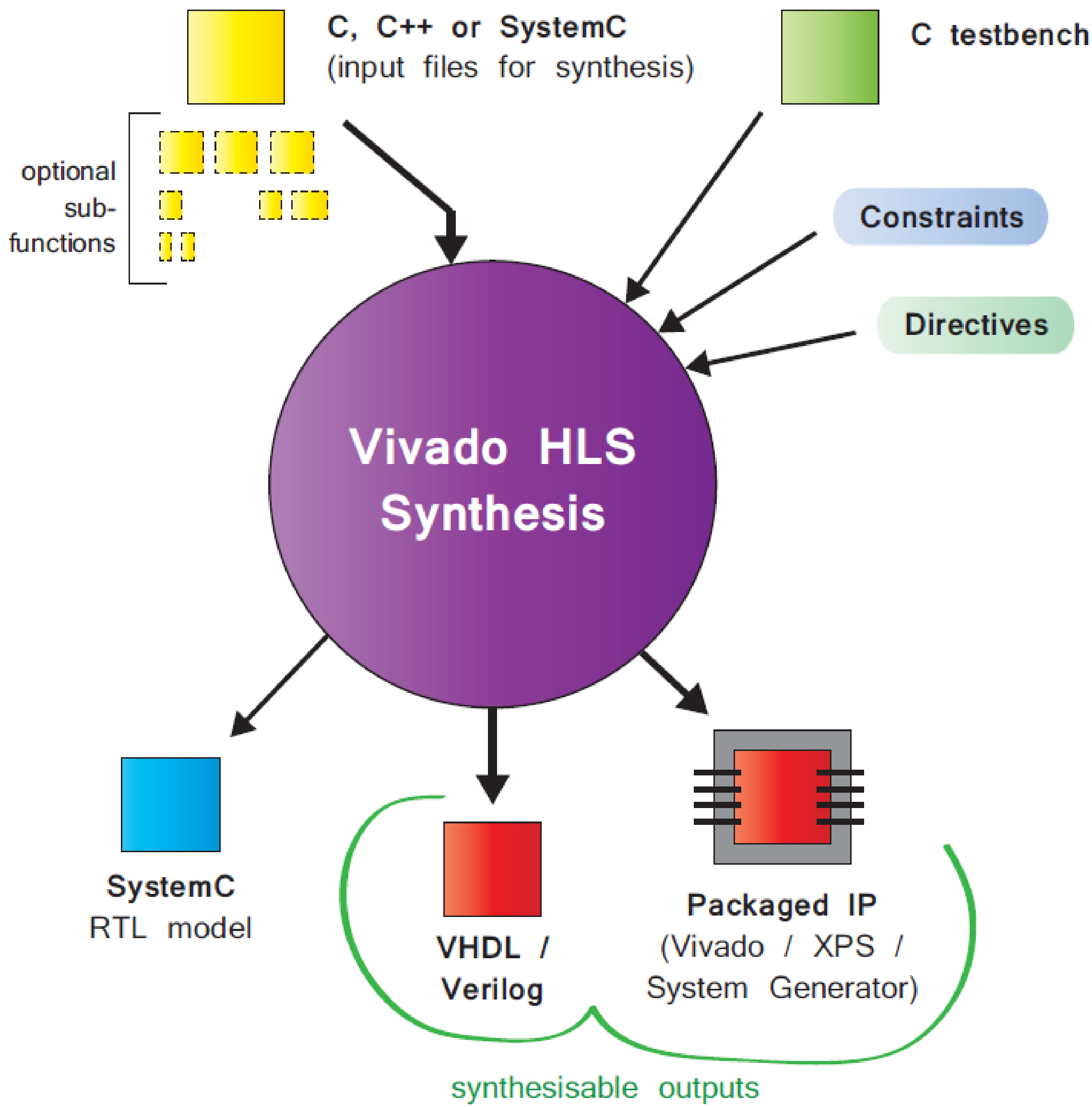
Vivado High Level Synthesis (2/2)

- The **functionality** is synthesized from the input code via the process of Algorithm Synthesis.
- The **interface** is created either be manually specified, or inferred from the code (Interface Synthesis).



Inputs/Outputs of Vivado HLS (1/3)

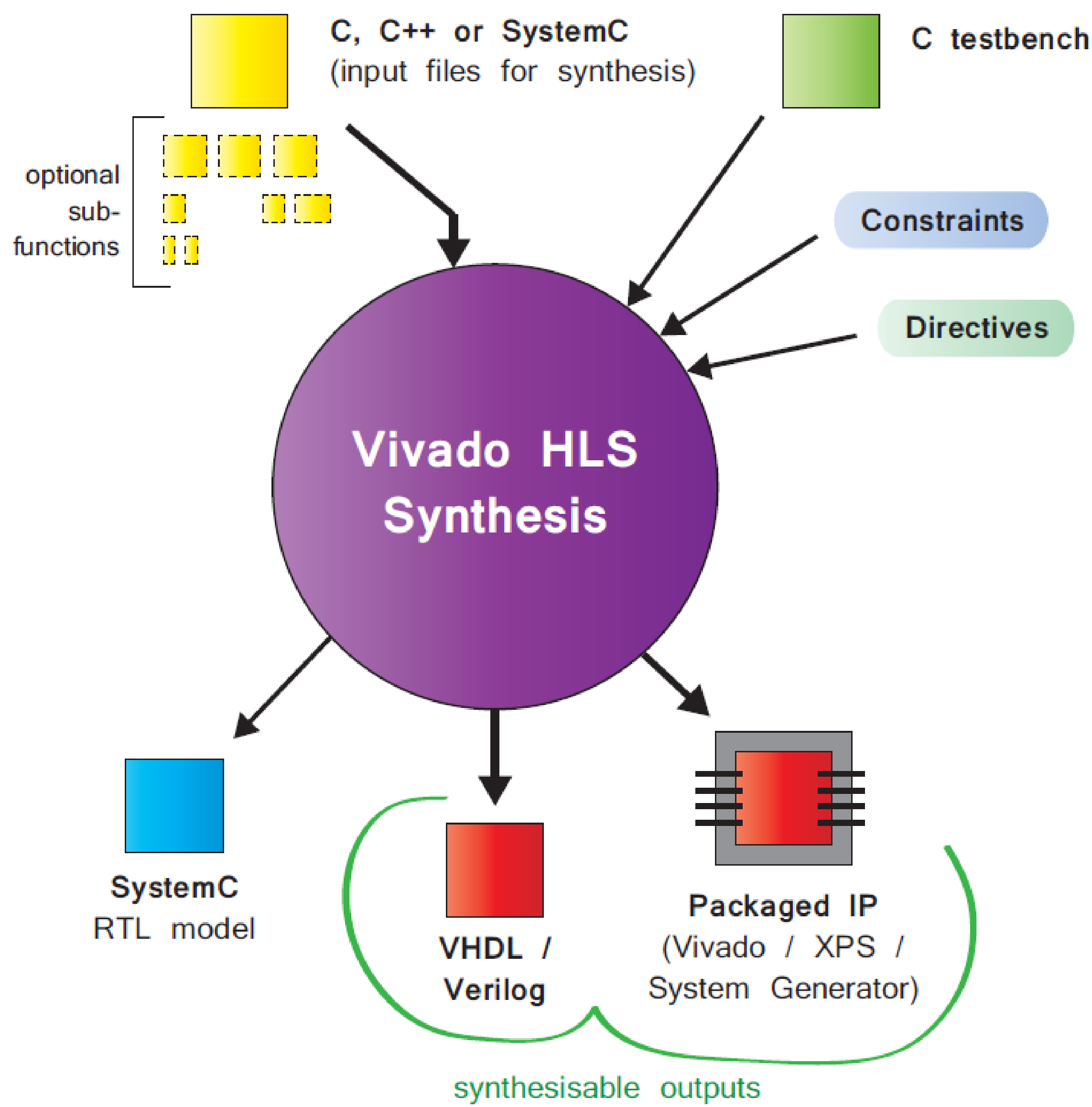
- **Inputs** of Vivado HLS process:
  1) **C, C++ or SystemC files**
     - These contain the functions to be synthesized.
  2) **C testbench files**
     - These form the basis for verifying both the C code and the RTL code generated by the HLS process.



Inputs/Outputs of Vivado HLS (2/3)

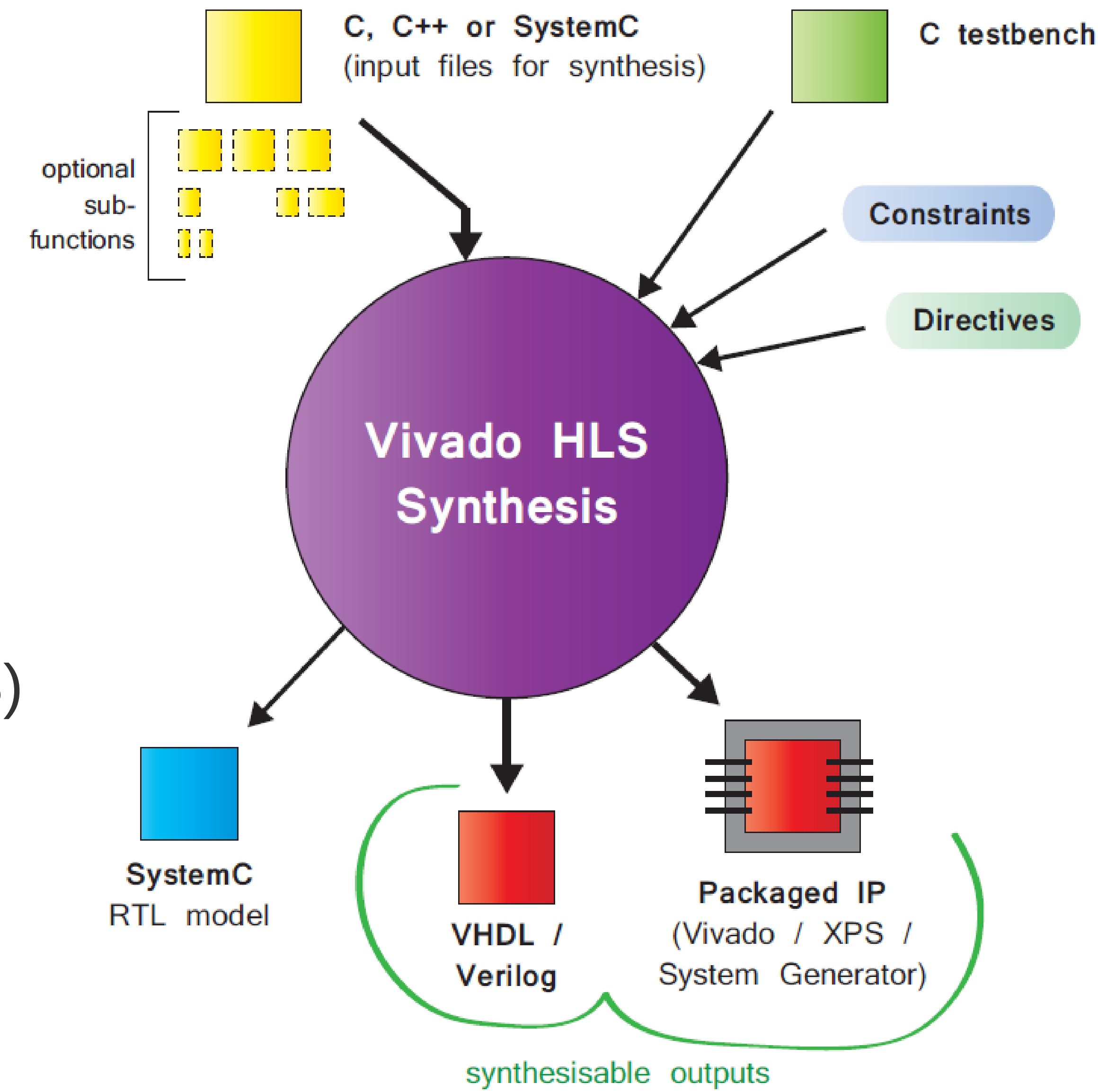
- **Inputs** of Vivado HLS process:
  3) **Constraints***
     - The timing constraint (desired clock period), along with a clock uncertainty figure and details of the target device.
  4) **Directives***
     - The style of implementation generated from the high-level.

**: optional*



Inputs/Outputs of Vivado HLS (3/3)

- The **outputs** produced are as listed below.
  1) **SystemC model**
  2) **VHDL or Verilog files**
  3) **Packaged IP** (for Vivado, System Generator, or XPS)
- The designer is able to choose based on different prototyping styles.



Summary

- What is High Level Synthesis (HLS)?
- Why High Level Synthesis (HLS)?
- Design Metrics in HLS
- High-Level Synthesis Process
  - Scheduling
  - Binding
- Vivado High Level Synthesis
  - Algorithm Synthesis
  - Interface Synthesis
  - Inputs and Outputs