Standardized ML EDA Data Pipelines with Ontology-driven Data Mapping

Vidya A. Chhabria, Arizona State University

Arizona State University

Outline

- **Challenges and motivation for ML EDA data pipelines**
- Si2 open standards working group ML EDA data pipeline
  - Proposed flow overview
  - Components of the flow with an example usecase
- Conclusion

Challenges with Data Formats

Challenges with data formats:

- AI-unfriendly data structures and interfaces of EDA tools, hindering high volumes of data queries, transfer, and processing
- Steep learning curve of EDA knowledge and tools for AI experts
- Lack of shared intermediate representation, hindering data reuse across projects

Challenges with creating a dataflow pipelines

- Lack of an ontology to standardize naming convention
- Diverse tools with inconsistent data formats
- EDA tool servers are different from ML GPU servers and require the transfer of large amount of data quickly

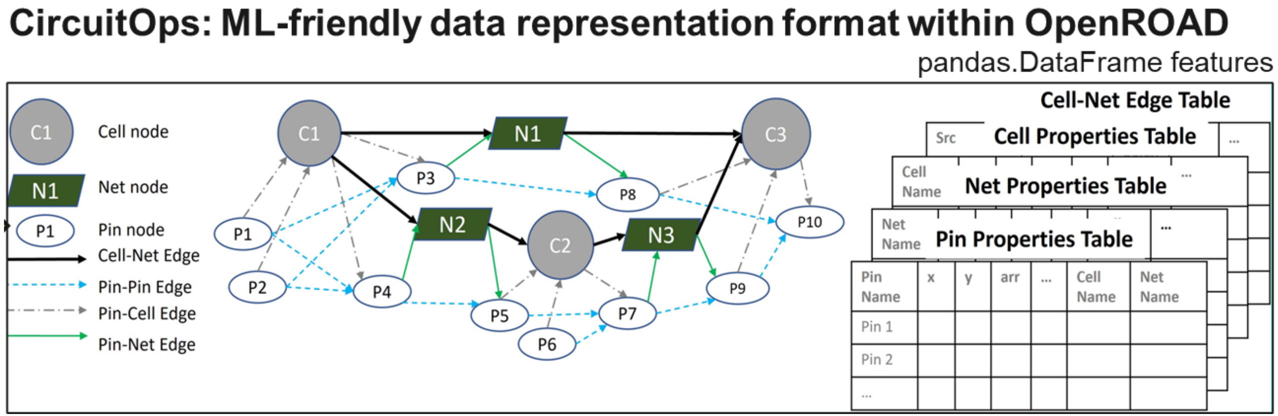
Arizona State University

CircuitOps: Existing Dataformat for ML EDA

Slide credits: Rongjian Liang, NVIDIA

https://github.com/NVlabs/CircuitOps

- Labeled-property graph-based data format backed by IR tables (csvs)
- Suitable for training data generation for different ML EDA applications by filtering out data from LPG



- Data communication between servers and lack of ontology still remains a challenge

Arizona State University

Outline

- Challenges and motivation for ML EDA data pipelines
- Si2 open standards working group ML EDA data pipeline
  - **Proposed flow overview**
  - Components of the flow with an example usecase
- Conclusion

Arizona State University

Goal for Si2 Open Standards Working Group

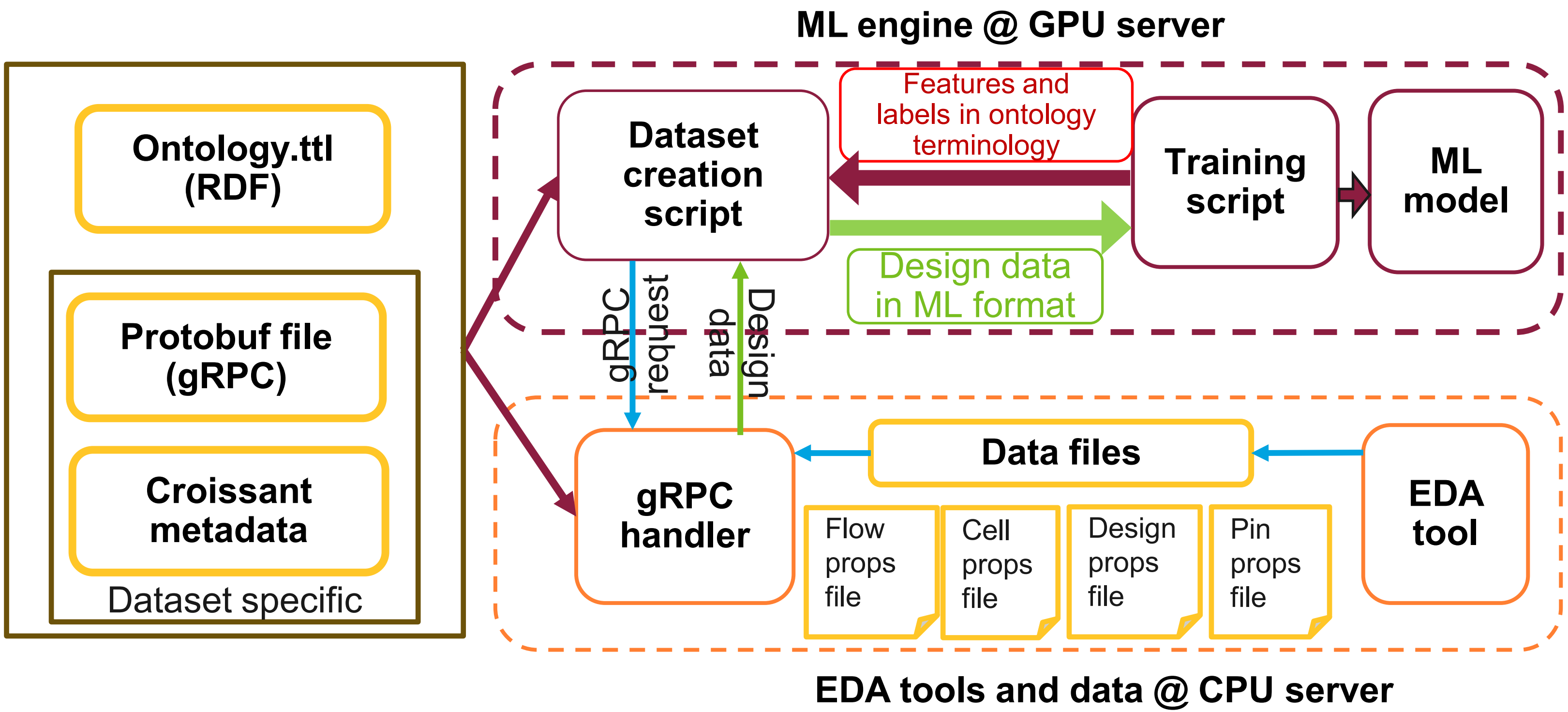
Enable AI-based EDA via:

- Standardized schema flows
- Ontology-based data relationships
- Tool-agnostic and dataset-agnostic data pipelines
- Seamless integration with EDA tools on CPU servers and ML on GPUs for training and inference

Si2 Open Standards Working Group

Proposed Flow Overview

Slide credits: Vikram Gopalakrishnan, ASU



Outline

- Challenges and motivation for ML EDA data pipelines
- Si2 open standards working group ML EDA data pipeline
    - Proposed flow overview
    - **Components of the flow with an example usecase**
- Conclusion

Arizona State University

Cell Arc Delay Example Usecase with CircuitOps as a Data Source

A model that predicts the cell delay with input slew and load capacitance.

- Features: Input pin slew, output load capacitance, cell type
- Labels: Arc delay

Required features and labels in ontology-compliant language

- Timing Graph with:
    - TimingEdge properties: edgeType, delayValue
    - sourceNode properties: worstSlew, libCellName
    - sinkNode properties: loadCapacitance

Parts of the RDF Ontology for Usecase

Highlighting those classes and properties required for our usecase

Owl Class Definitions

```
eda:Net a owl:Class ;
    rdfs:label "Net" ;
    eda:primaryId eda:netName ;
    rdfs:comment "A collection of electrically connected pins and wires." .

eda:SignalNet a owl:Class ;
    rdfs:subClassOf eda:Net ;
    rdfs:label "Signal Net" .

eda:PowerNet a owl:Class ;
    rdfs:subClassOf eda:Net ;
    rdfs:label "Power Net" ;
    rdfs:comment "Net that distributes power (e.g., VDD, VSS, or analog supplies)" .

eda:Wire a owl:Class ;
    rdfs:label "Wire" ;
    rdfs:comment "A physical segment of metal interconnect belonging to a net." .

eda:Via a owl:Class ;
    rdfs:label "Via" ;
    rdfs:comment "A vertical connection between different metal layers, belonging to a net." .

eda:Pin a owl:Class ;
    rdfs:label "Pin" ;
    eda:primaryId eda:pinName ;
    rdfs:comment "A connection point on an instance or a port." .

eda:SignalPin a owl:Class ;
    rdfs:subClassOf eda:Pin , [
            a owl:Restriction ;
            owl:onProperty eda:connectsTo ;
            owl:allValuesFrom eda:SignalNet
    ] ;
    rdfs:comment "Pin that transmits logical or timing-related information." .
```

Pin Properties

```
### Pin properties
eda:pinName a owl:DatatypeProperty ;
    rdfs:domain eda:Pin ;
    rdfs:range xsd:string ;
    rdfs:label "pin name" .

eda:pinX a owl:DatatypeProperty ;
    rdfs:domain eda:Pin ;
    rdfs:range xsd:float ;
    rdfs:label "pin X" ;
    qudt:unit unit:MicroM .

eda:pinY a owl:DatatypeProperty ;
    rdfs:domain eda:Pin ;
    rdfs:range xsd:float ;
    rdfs:label "pin Y" ;
    qudt:unit unit:MicroM .

eda:worstSlew a owl:DatatypeProperty ;
    rdfs:domain eda:Pin ;
    rdfs:range xsd:float ;
    rdfs:label "worst slew" ;
    qudt:unit unit:NanoSecond .

eda:loadCapacitance a owl:DatatypeProperty ;
    rdfs:domain eda:Pin ;
    rdfs:range xsd:float ;
    rdfs:label "load capacitance" ;
    qudt:unit unit:FemtoFarad .

eda:isClockPin a owl:DatatypeProperty ;
    rdfs:domain eda:Pin ;
    rdfs:range xsd:boolean ;
    rdfs:label "is clock pin" .
```

Slide credits: Ivan Kissiov

Parts of the RDF Ontology for Usecase

Highlighting those classes and properties required for the usecase

Timing graph

```
### Timing graph properties
eda:hasNode a owl:ObjectProperty ;
    rdfs:domain eda:TimingGraph ;
    rdfs:range eda:TimingNode ;
    rdfs:label "has node" ;
    rdfs:comment "Relates a timing graph to a timing node contained within it." .

eda:hasEdge a owl:ObjectProperty ;
    rdfs:domain eda:TimingGraph ;
    rdfs:range eda:TimingEdge ;
    rdfs:label "has edge" ;
    rdfs:comment "Relates a timing graph to a timing edge contained within it." .

eda:sourceNode a owl:ObjectProperty ;
    rdfs:domain eda:TimingEdge ;
    rdfs:range eda:TimingNode ;
    eda:primaryId eda:sourcePinName ;
    rdfs:label "source node" ;
    rdfs:comment "Relates a timing edge to its source node." .

eda:sinkNode a owl:ObjectProperty ;
    rdfs:domain eda:TimingEdge ;
    rdfs:range eda:TimingNode ;
    eda:primaryId eda:sinkPinName ;
    rdfs:label "sink node" ;
    rdfs:comment "Relates a timing edge to its sink node." .

eda:represents a owl:ObjectProperty ;
    rdfs:domain eda:TimingNode ;
    rdfs:range [ a owl:Class ; owl:unionOf(eda:SignalPin eda:SignalNet eda:SignalPort) ] ;
    rdfs:label "represents" ;
    rdfs:comment "Links a timing node to the design element (pin, port, or net) it represents." .
```

```
eda:delayValue a owl:DatatypeProperty ;
    rdfs:domain eda:TimingEdge ;
    rdfs:range xsd:float ;
    rdfs:label "delay value" ;
    qudt:unit unit:NanoSecond .

eda:edgeType a owl:DatatypeProperty ;
    rdfs:domain eda:TimingEdge ;
    rdfs:range xsd:bool ;
    rdfs:label "edge type" .

eda:riseDelay a owl:DatatypeProperty ;
    rdfs:domain eda:TimingEdge ;
    rdfs:range xsd:float ;
    rdfs:label "rise delay" ;
    qudt:unit unit:NanoSecond .

eda:fallDelay a owl:DatatypeProperty ;
    rdfs:domain eda:TimingEdge ;
    rdfs:range xsd:float ;
    rdfs:label "fall delay" ;
    qudt:unit unit:NanoSecond .
```

Timing edge properties

Slide credits: Ivan Kissiov

Croissant Metadata for CircuitOps

- Croissant is a metadata format for structured datasets, enabling interoperability and schema definition.
- Key elements:
    - *Dataset* – Represents the entire collection of files and metadata.
    - *FileObject* – Defines individual files in the dataset (e.g., CSVs, images, JSON).
    - *RecordSet* – Describes the structure of tabular data (e.g., design_properties.csv).
    - *Field* – Specifies individual columns in a RecordSet, including data types and references.
- Reference: https://docs.mlcommons.org/croissant/docs/croissant-spec.html#resources

Croissant Metadata for CircuitOps

Dataset definition

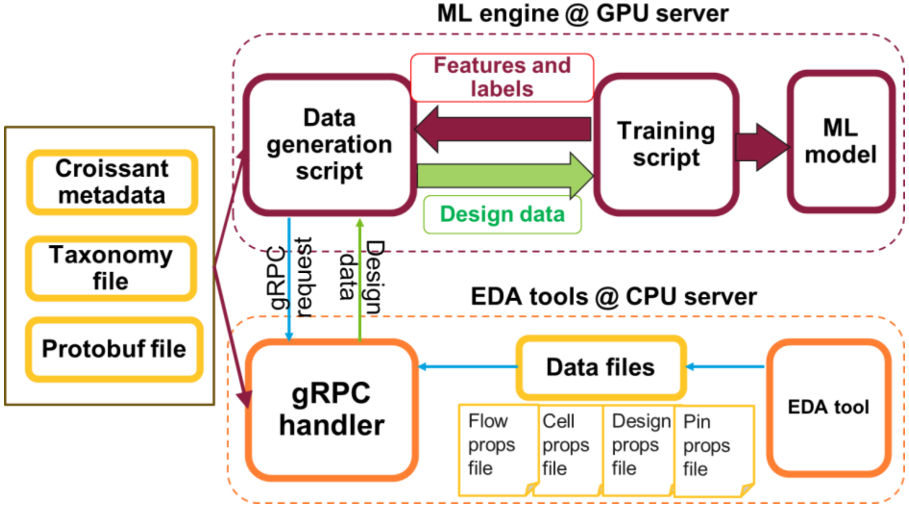
```json
{
  "@type": "Dataset",
  "@id": "source_dataset",
  "name": "Source Dataset",
  "description": "A dataset containing structured circuit design data.",
  "distribution": [
    {
      "@type": "cr:FileObject",
      "@id": "design_properties",
      "name": "design_properties.csv",
      "contentUrl": "data/design_properties.csv",
      "encodingFormat": "text/csv"
    },
    {
      "@type": "cr:FileObject",
      "@id": "net_properties",
      "name": "net_properties.csv",
      "contentUrl": "data/net_properties.csv",
      "encodingFormat": "text/csv"
    },
    {
      "@type": "cr:FileObject",
      "@id": "inst_properties",
      "name": "inst_properties.csv",
      "contentUrl": "data/inst_properties.csv",
      "encodingFormat": "text/csv"
    },
    {
      "@type": "cr:FileObject",
      "@id": "pin_properties",
      "name": "pin_properties.csv",
      "contentUrl": "data/pin_properties.csv",
      "encodingFormat": "text/csv"
    },
    {
      "@type": "cr:FileObject",
      "@id": "power_density",
      "name": "power_density.png",
```

RecordSet definition

```json
    {
      "@type": "cr:RecordSet",
      "@id": "pin_properties",
      "field": [
        {
          "@type": "cr:Field",
          "@id": "pin_properties/pinName",
          "name": "pinName",
          "dataType": "sc:Text",
          "source": {
            "fileObject": { "@id": "pin_properties" },
            "extract": { "column": "pin_name" }
          }
        },
        {
          "@type": "cr:Field",
          "@id": "pin_properties/worstSlew",
          "name": "worstSlew",
          "dataType": "sc:Float",
          "source": {
            "fileObject": { "@id": "pin_properties" },
            "extract": { "column": "pin_max_slew" }
          }
        },
        {
          "@type": "cr:Field",
          "@id": "pin_properties/loadCapacitance",
          "name": "loadCapacitance",
          "dataType": "sc:Float",
          "source": {
            "fileObject": { "@id": "pin_properties" },
            "extract": { "column": "load_capacitance" }
          }
        },
        {
          "@type": "cr:Field",
          "@id": "pin_properties/inputCapacitance",
          "name": "inputCapacitance",
          "dataType": "sc:Float",
          "source": {
            "fileObject": { "@id": "pin_properties" },
            "extract": { "column": "input_capacitance" }
```

Conclusion and acknowledgment

- Using those components described above, we have developed automated scripts for the ML EDA data pipeline
- Utilizing existing ML and ontology infrastructure is the least resistant path to build data pipelines (Croissant, RDF, protobuf)
- **Standards require community input and commitment to adoption**



Acknowledgments:

- Si2 open standard working group
- ASU VDA lab students

Arizona State University

Open-source ML EDA Research @ ASU

Synthetic data generation techniques using AI

- Timing cones (MIMIC): https://github.com/ASU-VDA-Lab/MIMIC
- Layout heatmaps using diffusion models (DALI-PD): https://github.com/ASU-VDA-Lab/DALI-PD

LLM chatbots and agents for OpenROAD

- EDA Corpus: Dataset and benchmarks to train chatbots for OpenROAD https://github.com/OpenROAD-Assistant/EDA-Corpus
- OpenROAD-Assistant: Chatbots QA and script generation https://github.com/OpenROAD-Assistant/OpenROAD-Assistant
- OpenROAD-Agent: Automated script generation and correction https://github.com/OpenROAD-Assistant/OpenROAD-Agent