SPFDs - A new method for specifying flexibility

From the paper:

"A New Method to Express Functional Permissibilities for LUT Based FPGAs and Its Applications"

by S. Yamashita, H. Sawada, and A. Nagoya

NTT Communications Science Lab., Kyoto, Japan

ICCAD'96

BUT: this paper is more about a new kind of don't cares, rather than about application to FPGAs.

Methods for Expressing Sets of Functions

- Don't Cares
  - Satisfiability DCs
  - Observability DCs
- ATPG Redundancy Removal
- Boolean Relations
- Sets of Boolean Relations

And now, SPFDs. Actually, they are sets of incompletely specified functions (ISFs).

ODCs and SPFDs

ODCs:

- Different ways to compute them
  - Full method (difficult)
  - Subsets
    * Compatible Sets of Permissible functions (CSPFs) - (Easier to compute).
    * Compatible observability don't cares (CODCs - see Savoj thesis.) - generalization of CSPFs
- CSPFs (and CODCs)
    * have advantage that they can be computed fast - from outputs to inputs.
    * are independent

SPFDs:

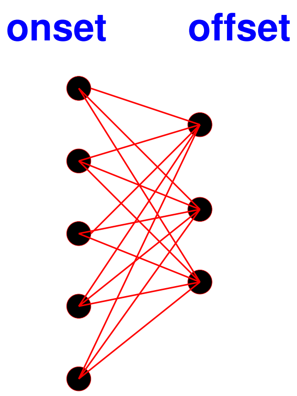
- are like CSPFs/CODCs in the above, but are more powerful and are not a subset of DCs (they are a set of ISFs)

ISFs

Usually we deal with binary incompletely specified functions (ISFs).

- They can be represented as complete bipartite graphs on minterms.

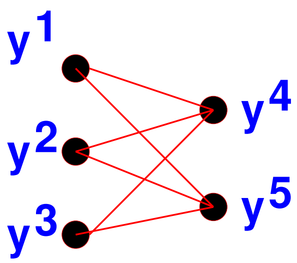
- An edge means that we need to build a function that distinguishes the two minterms on the edge.

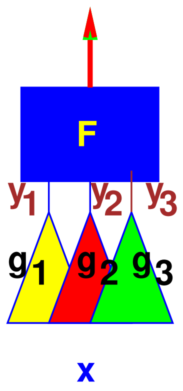


A connected bipartite graph has only two colorings, one the onset and the other the offset or vice versa.

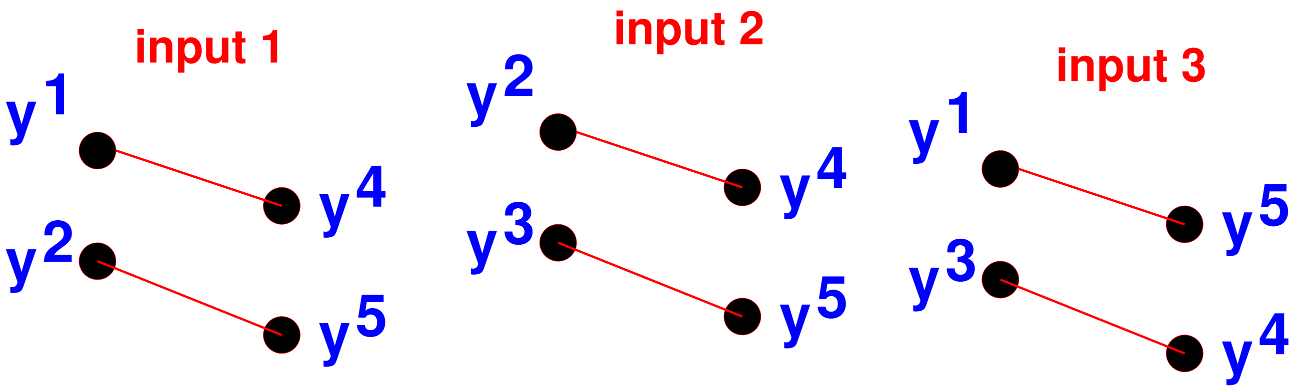
Building F

We start with a complete bipartite graph (ISF) for F. This gives a set of minterms that must be distinguished.



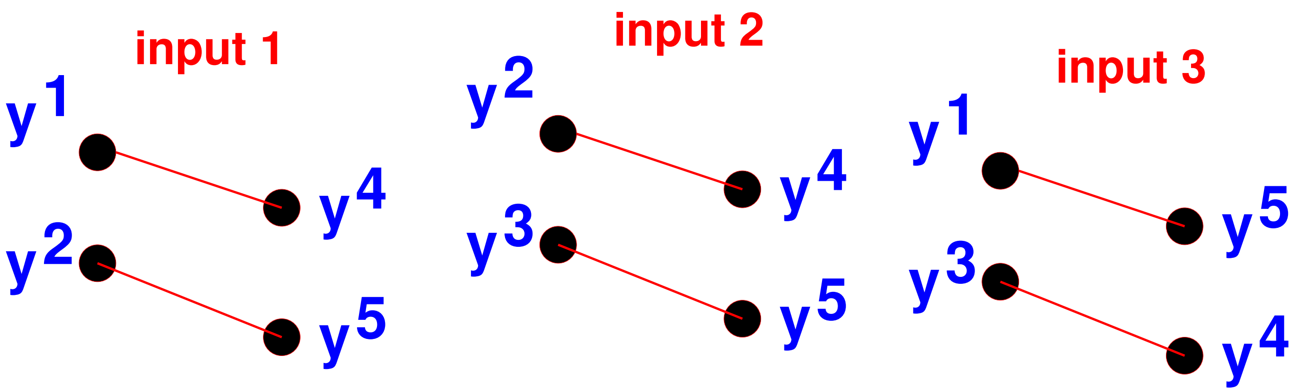


Suppose that the inputs to F are $y = (g_1(x), g_2(x), g_3(x))$. As long as $\{y^1, y^2, y^3\}$ are encoded differently from $\{y^4, y^5\}$ by $g(x)$, there exists a function F which implements the ISF. We can distribute the encoding job separately to the input wires.



Input Graphs

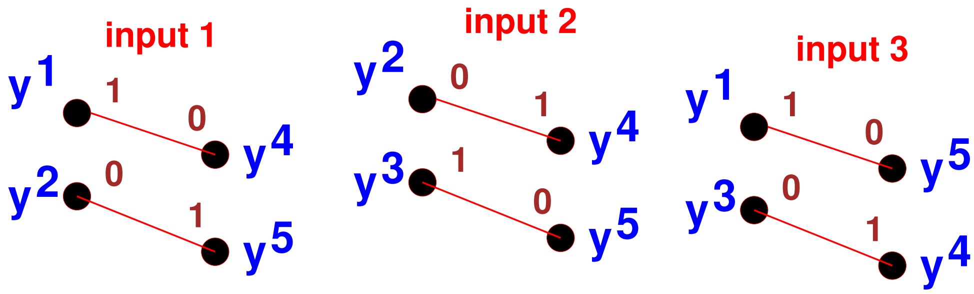
Each of the edges on the input graphs means that the input signal $g_i(x)$ has to have a different value for the two nodes.



Note the graphs are bipartite, but not complete and may be disconnected. In the above case, each graph has 4 different legal colorings.

Assigning Values

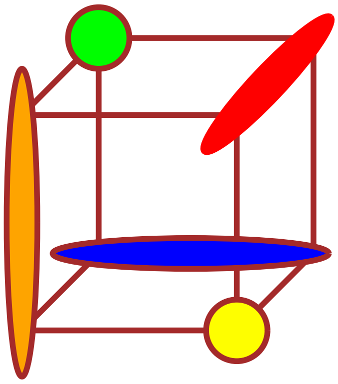
Now we can assign values to the graphs (color the graphs).



Thus,

$$y^1 = 1*1, y^2 = 00*, y^3 = *10, y^4 = 011, y^5 = 100$$

Note that y^1, y^2, y^3, y^4, y^5 are all uniquely encoded in this case but in general, that is not necessary.



The original ISF can be implemented with this encoding

$$F(y^1) = F(y^2) = F(y^3) = 1 \text{ and } F(y^4) = F(y^5) = 0$$

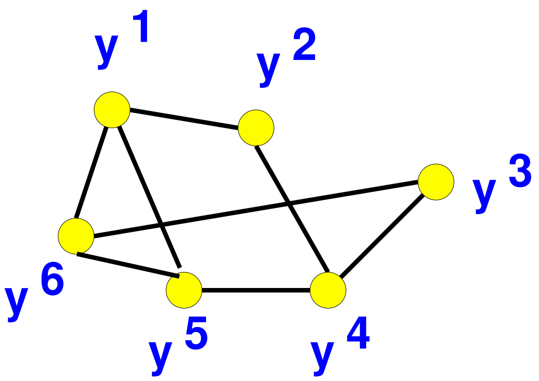
Now just call `ESPRESSO`.

SPFDs

Now we generalize this because,

1. we can
2. it is more powerful (*can be applied to multiple outputs and multiple valued variables*)
3. it improves understanding.

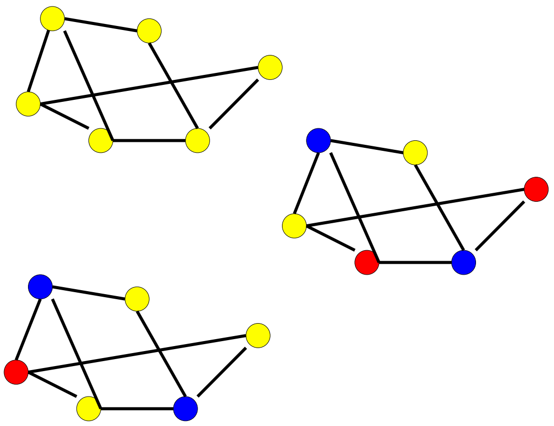
Definition 1 *A* **SPFD** $\mathcal{F}(y)$ *on domain* Y *is an undirected graph* (V, E) *where each* $v \in V$ *is encoded as a minterm* $v = (y_1, \ldots, y_k) \in Y$*.*



Implements

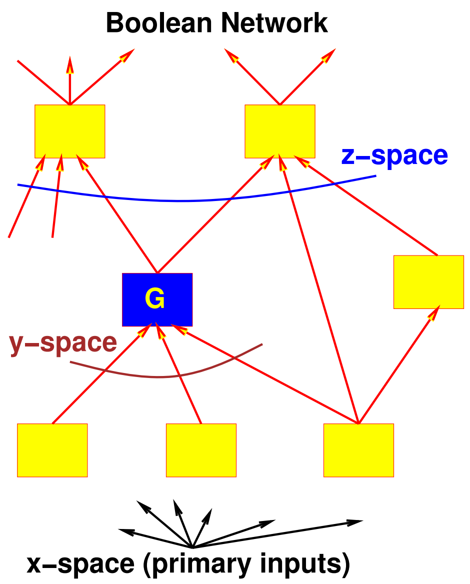
Definition 2 *A function* $f(y)$ **implements** $\mathcal{F}(y) = (V, E)$ *if* $f(y), y \in V$ *is a valid coloring of* $\mathcal{F}$*, i.e.*

$$f(y^1) \neq f(y^2), (y^1, y^2) \in E$$



Two Colorings
Must have at least 3 colors since
the chromatic number of graph is 3

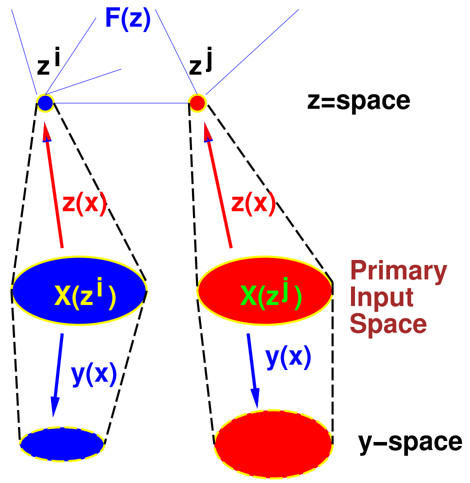
Compatible



Definition 3 *A mapping* $y(x) = (y_1(x), \ldots, y_k(x))$ *is* $\mathcal{F}(z)$**-compatible** *if*

$$\forall(z^i, z^j) \in \mathcal{F}(z), y(X(z^i)) \cap y(X(z^j)) = \emptyset$$

where $X(z^i) = \{x | z(x) = z^i\}$.

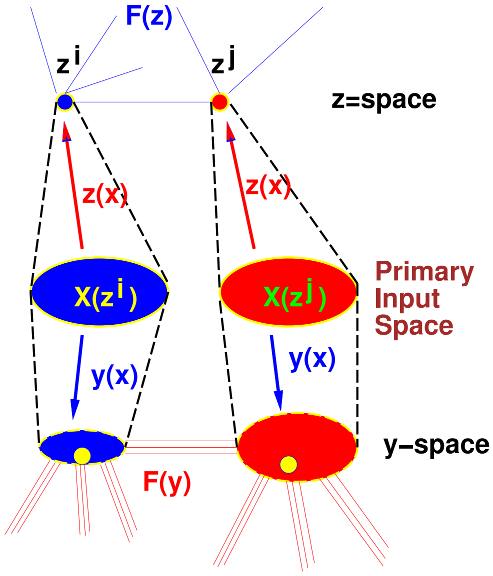


Re-implementing a Function

Theorem 1 *If $y(x)$ is $\mathcal{F}(z)$-compatible, then there exists a function $\widehat{f}(y)$ which implements $\mathcal{F}(z)$.*

Proof.

- Translate SPFD $\mathcal{F}(z)$ into $\mathcal{F}(y)$.



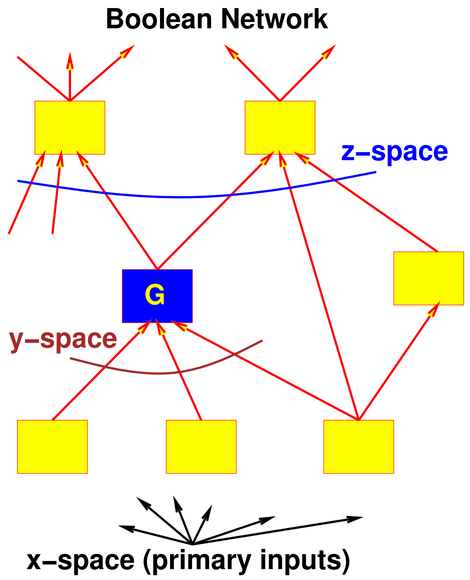
- Color $\mathcal{F}(y)$ and implement the corresponding ISF by some function $\widehat{f}(y)$.

□

Each $y(X(z^i))$ creates an equivalence class which can all be colored with the same color, but different colors can be used.

Re-implementing the Inputs to a Function

We want to build new functions $\widehat{y}(x)$ at the inputs to G.



For new function $\widehat{y}(x)$, just need for each edge in $\mathcal{F}(y)$, at least one input that separates:

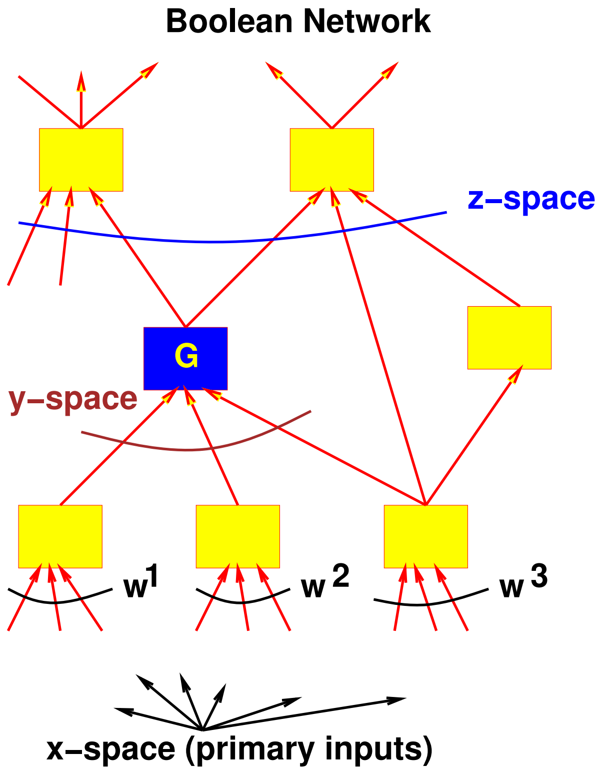
Theorem 2 *$\widehat{y}(x)$ is $\mathcal{F}(z)$-compatible if*

$$\forall (z^i, z^j) \in \mathcal{F}, \exists l, \forall (x^i, x^j) \in (X(z^i), X(z^j)), \widehat{y}_l(x^i) \neq \widehat{y}_l(x^j)$$

Proof. Since $\widehat{y}_l(x^i) \neq \widehat{y}_l(x^j)$, then $\widehat{y}(x^i) \neq \widehat{y}(x^j)$. Hence $\widehat{y}(X(z^i)) \cap \widehat{y}(X(z^j)) = \emptyset$. $\square$

Re-using Old Topology

A efficient way of building a new mapping $\hat{y}(x)$ is to use part of the old implementation.



Construct sets $\{\mathcal{F}_1(y), \ldots, \mathcal{F}_k(y)\}$ such that

1. $\mathcal{F}_l(y) = \{(y^i, y^j) \in \mathcal{F}(y)$ only if $y_l^i(x) \neq y_l^j(x)\}$

2. $\forall (y^i, y^j) \in \mathcal{F}(y), \exists l (y^i, y^j) \in \mathcal{F}_l(y)$

Efficient Re-implementaion

Theorem 3 *Suppose the current implementation of the global functions at* y, $y(x)$, *are implemented with sets of intermediate variables* $w = (w^1, \ldots, w^k)$, *i.e.* $y = (f_1(w^1(x)), \ldots, f_k(w^k(x)))$. *Then for each* l, *any coloring of* $\mathcal{F}_l(y)$ *can be implemented with some* $\widehat{f}_l(w^l(x))$.

Thus a new encoding $\widehat{y}(x)$ can be obtained as

$$(\widehat{f}_1(w^1(x)), \ldots, \widehat{f}_k(w^k(x)))$$

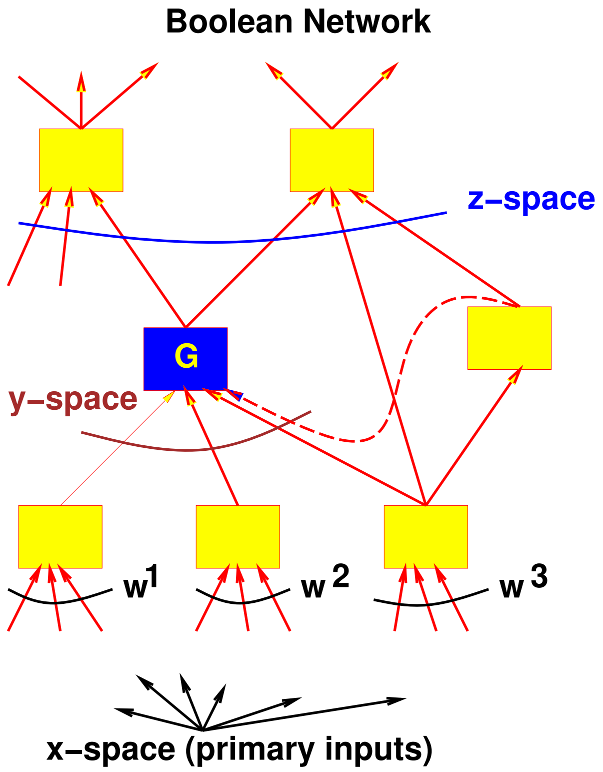
i.e. a set of functions of the current intermediate inputs to $f_1, \ldots, f_k$.

Proof. Consider $\mathcal{F}_l(y)$ for some l. The current f_l implements $\mathcal{F}_l(y)$ since for $(y^i, y^j) \in \mathcal{F}_l, f_l(y^i) \neq f_l(y^j)$, i.e.

$$f_l(X(y^i)) \cap f_l(X(y^j)) = \emptyset$$

Thus for all $x^i \in X(y^i), x^j \in X(y^j)$, we must have $w^l(x^i) \neq w^l(x^j)$. Otherwise, $f_l(w^l(x^i)) = f_l(w^l(x^j))$ contradicting that $f_l(y^i) \neq f^l(y^j)$. Since w^l distinguishes x^i from x^j, functions $\widehat{f}_l(w^l)$ exist which separate x^i and x^j. $\square$

How Do We Use This



Note that since $y(x)$ is $\mathcal{F}(z)$-compatible, then so is $y^{+} = (y(x), y_{k+1}(x))$.

Theorem 4 *Wire y_1 can be removed if and only if $(y_2, \ldots, y_{k+1})$ is $\mathcal{F}(z)$-compatible.*

Conclusions

- SPFDs offer greater choice in implementing a circuit
- New way to obtain Boolean divisors
- Implementation using MDDs in progress
- May be useful for DSM for rewiring or re-coding for noise avoidance