Modelling Mixed 4phase Pipelines: Structures and Patterns

Graham M. Birtwistle
Department of Computer Science
The University of Sheffield

Kenneth S. Stevens
Electrical and Computer Engineering
University of Utah

***Abstract*—This paper presents an exploration of the design space for homogeneous and mixed 4phase asynchronous linear pipelines. We extend previously published results by uncovering their complete ordered design space, demonstrate relationships between the latter's governing lattice structures, tabulating the ways in which mixed linear pipelines compose, and establish the basic rules underpinning mixed pipeline behaviours. The inherent structures and patterns we describe give rise to a succinct and transparent method for calculating the specifications of mixed and homogeneous pipelines of arbitrary depths. Practical applications of this theory are demonstrated.**

I. Setting and approach

The computational core of many asynchronous designs is a pipelined datapath. It is standard practice in the first stage of design development to concentrate upon control signals and the ways in which they can interleave. This enables one to check that subsystems compose harmoniously (are safe, live, preserve essential properties, etc.) before extending the model towards data movements and calculations.

In this paper, we explore the design space for both homogeneous and mixed 4phase asynchronous linear pipelines, and show how the complete ordered design space can be described and tabulated in a succinct and transparent way. The lattice-based method we describe has both strictly formal underpinnings (it is based on a formal description of pipelines as processes expressible in the process calculus CCS [14]) and has clear industrial relevance: our previously reported work (summarised below) has already allowed the abstract analysis of all possible untimed 4phase controller interleavings, and synthesized an implementation of each protocol in silicon.

Our approach complements and extends that of McGee and Nowick [13], who were the first to present a lattice-based framework for modelling asynchronous pipelines, and also those of Furber *et al.* [8], [9], [11] and Lines [10], who present earlier explorations of various related protocols. Also of relevance are Nowick and Singh's insightful overviews of asynchronous pipeline design styles and techniques [17], and Blunno *et al.*'s analysis of handshake protocols [3].

In previous work [1], [21] we presented a family of abstracted untimed (delay-insensitive and speed-independent) 4phase latch controllers derived by state reduction from max_1, the most concurrent such abstraction. The family is obtained by systematically cutting away input states and/or output states from max_1. Applying each combination of output channel cut (Lcut) and input channel cut (Rcut) to max_1 generates the *shape* (a minimised state machine) of a new family member which, if live, represents a viable protocol. The cuts form separate Lcut and Rcut lattices, $\mathcal{L}$ and $\mathcal{R}$, whose $10{\times}25$ cartesian product, $\mathcal{L}{\times}\mathcal{R}$, defines the complete untimed design space. These abstracted controllers were synthesized from their specifications, placed and routed, and characterised via ModelSim and PrimeTime using post layout extraction based upon the static Artisan 12T library on IBM's 65nm 10sf process. We were thus able to compare such properties as area, cycle time, forward and backward latencies, and power per datum over a whole family of automatically generated circuits. This emphasises the point that state reduction via cuts is not aimed primarily at finding the smallest viable circuits, but at opening up the space of all possible circuit abstractions for exploration. See [16] for an illuminating application of selected shapes to asynchronous access control schemata.

In [2] we used this approach to explore mixed linear pipelines in the much smaller 2phase design space (containing just $3{\times}6$ cut combinations). We were able to relate the lattice $\mathcal{L}$ of output channel cuts to the lattice $\mathcal{R}$ of input channel cuts, and showed how this gives fruitful insights into calculating mixed pipeline behaviours. In particular, the independence of Lcut and Rcut behaviours exhibited for 4phase controllers when composed into homogeneous pipelines in [21] was found to hold for 2phase mixed pipelines as well. This independence is extremely important as it enables one to generate and utilise separate compact tables LTAB:$\mathcal{L}{\times}\mathcal{L}$ and RTAB:$\mathcal{R}{\times}\mathcal{R}$ to characterise pipeline behaviours. Without such independence, one would expect a much larger $(\mathcal{L}{\times}\mathcal{R})^2$ table to be needed. Finally we were able to show that each quadrant of the larger $6{\times}6$ RTAB is related to, and can be determined from, the smaller $3{\times}3$ LTAB.

As we shall see, these insights can be transferred *in toto* to the much larger 4phase design space, where $\mathcal{L}$ has 35 output cuts and $\mathcal{R}$ has 140 input cuts. The corresponding LTAB has dimensions $35{\times}35$ and RTAB may be partitioned into 16 like-sized sub-tables. We have developed succinct and transparent (one line) formulae for both LTAB and for pertinent partitions of RTAB. In short, we have a concise algorithm for calculating the specifications of mixed (and hence also homogeneous) 4phase pipelines of arbitrary depths.

A. Structure of the paper

In the remaining sections in this paper: (II) gives an overview of the above approach and pertinent results of [21], and describes both max_1, the maximal 4phase controller, and its shape. (III) presents the full Lcut and Rcut lattices, how to relate them, the complete family of shapes cutaway from max_1, and patterns of growth in homogeneous pipelines. (IV) develops the structures and lattices arising from mixed pipelines. (V) and (VI) develop LTAB and RTAB and their index/extent formulae respectively which cover mixed pipeline behaviours. We conclude in (VII) by presenting several small case studies which hint that mixed pipelines containing timed stages can be coerced into pipelines with untimed interfaces.

II. max_1—The Maximal 4phase Controller

In this section we review key properties of max_1, the most concurrent 4phase controller, and its properties when composed into homogeneous pipelines.

A. The max_1 state machine

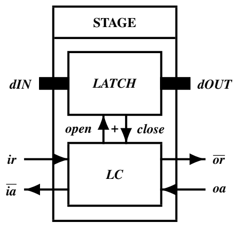


Fig. 1. Stage = Latch and Controller

Joined by Varanesi in [21], the authors give a model for a pipeline stage consisting of a *LATCH* which, when ready, captures a fresh data value from input bus *dIN* and shows it on output bus *dOUT*; together with controller *LC* which is responsible for the safety of such operations under the assumption that data is bundled and valid on the rising edge $ir\uparrow$. Since a stage operates the same way whatever the incoming data, it is usual to omit buses from the model. Informally, the interplay between *LATCH* and *LC* will result in added internal time spent but makes no observable (externally visible) difference to control signal interleaving possibilities. A formal explanation of this equivalence is given in [21].

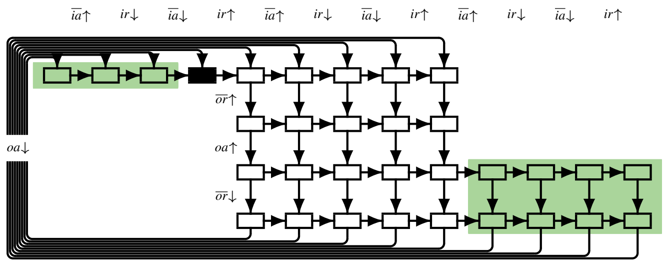


Fig. 2. Minimised equivalent state machine for max_1

In [21], the authors argue that the most concurrent 4phase latch controller abstracts to the state machine shown in Figure 2. Its specification is given entirely in terms of the interleaving possibilities of its four external handshake lines. By convention, outgoing signals are overbarred, whilst incoming signals are not. Input request/acknowledge signals flow horizontally, whilst output request/acknowledge signals flow vertically down and then wrap around to the top shifted left by 4 states.

The initial state in Figure 2 (and all shape diagrams) is marked ■ . When quiescent, the circuit will return to this state to await the next transaction. The upper left shaded three states show underrun when the output channel gets ahead of the input channel. The lower right shaded 4×2 block of states shows overrun when the input channel is getting well ahead of the output channel. It may only be entered after an $oa\uparrow$ has been accepted and a fresh data value is guaranteed to have been passed.

The *shape* abstraction was introduced in [1] as a sufficient and clear abstraction of max_1 and its cutaways. Figure 3 gives the shape of max_1. Again the initial state is indicated by ■, other live states by □. We save clutter by omitting the wrap around $oa\downarrow$ arrows.

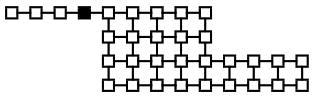


Fig. 3. *shape*: uncluttered shorthand for max_1

B. Growth of homogeneous max_1 pipelines

Let HP(S,d) represent the homogeneous pipeline built from *d* copies of the controller S. max_d abbreviates to HP(max_1,d). The shapes of pipelines $max_1..max_3$ are pictured in Figure 4.

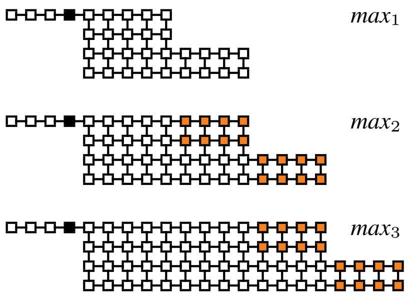
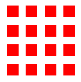


Fig. 4. HP(max_1, 1..3): pipelines built with max_1

Since max_d is the maximal pipe of depth d, all linear homogeneous (and mixed) pipelines from core shapes may be expressed as Lcuts and Rcuts from max_d. The pipelines HP(max_1, d) grow in regular fashion with state sizes 32, 48, 64, ... This growth is emphasised in Figure 4 by shading the extra states per iteration. Notice how max_d *keeps its shape* as d grows. We may indicate this growth informally by

$$max_{d+1} = max_d +$$



III. Families of Related Shapes

In this section, we extend the design space from the untimed $10{\times}25$ already explored in [1], [21] to the complete space of 35 Lcuts by 140 Rcuts, relate these larger Lcut and

Rcut lattices, show how to characterise timed and untimed controllers by their cuts, and present experimental results over the complete design space.

A. *Left cuts on output signals*

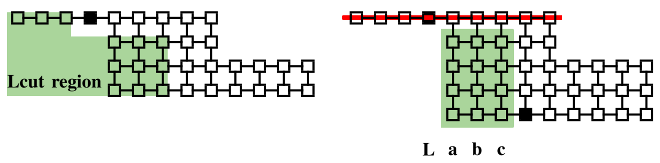


Fig. 5. Two views of the Lcut region: L000..L444

On the left of Figure 5 we shade the region for valid Lcuts (output channel cuts) over the standard shape abstraction for max_1. On the right, we have moved the top line of the shape down 4 rows with an appropriate alignment to the right. The potential candidates for a left cut now lie in the shaded 4 by 3 block of states. This view makes it easier to introduce the Lcut notation. Lcut Labc denotes the removal from max_1 of a states from the column marked a, b states from column b, and c states from column c of max_1, working vertically upwards. This is a notational change from [21] which made horizontal cuts across input signals. The change makes cuts along the flow of output signals $\overline{or}\uparrow.oa\uparrow.\overline{or}\downarrow.oa\downarrow$ (here, from right to left) and assists the search for relationships between Lcuts and Rcuts.

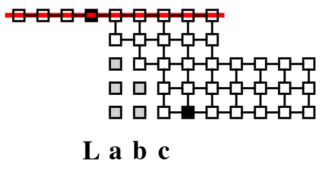


Fig. 6. Lcut L320

Figure 6 depicts the specific cut L320 in which the 5 cutaway states are shaded and their incoming and outgoing arcs removed.

Labc constraints: $(0 \leq a,b,c \leq 4) \wedge (a \geq b \geq c)$

The constraints ensure that all states remaining after an Lcut can be both entered and exited (there are no dead end states). If we try to cut away more states on the left than L444, we lose the ability to return to the initial state.

B. *$\mathcal{L}$: the Lcut lattice*

Figure 7 displays LL, the lattice of 35 Lcuts, arranged in a wedged pattern of triangles T5..T1.

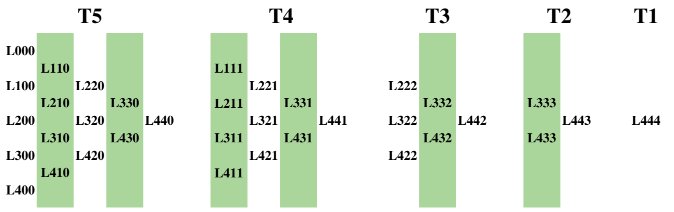


Fig. 7. LL: wedge representation of the 35 Lcuts

The *corner point* notation **L000..L400..L444** is a convenient shorthand for representing the Lcuts in the full lattice T5..T1. It may be extended to cover selected sub-lattices, e.g. T4..T2 is represented by **L111..L411..L443**. The shading of columns of cuts in Figure 7 and Figure 11 is explained in subsection III-E.

C. *Right cuts on input signals*

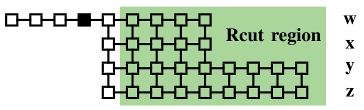


Fig. 8. Rcut region: R0000..R4488

The Rcut region is shown shaded in Figure 8. When making Rcuts, we remove states row by row working horizontally from right to left. Rwxyz denotes the removal from max_1 of w states from the row marked w, x states from the row marked x, etc. The maximal cutaway per row is 4 for rows w and x; and 8 for rows y and z.

Rwxyz constraints: $(0 \leq w,x \leq 4) \wedge (0 \leq y,z \leq 8) \wedge (x+4 \geq y \geq z \geq w \geq x)$

The constraints ensure that all states remaining after an Rcut can be both entered and exited (there are no dead end states). If we try to cut away more states on the right than R4488, we lose the ability to return to the initial state.

Figure 9 depicts the specific cut R1032 from max_1. The 6 cutaway states are shaded and their incoming and outgoing arcs deleted. No deleted state is reachable from the initial state.

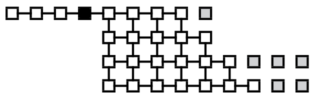


Fig. 9. Rcut R1032

Figure 10 shows the result of taking both Lcut L420 and Rcut R2042 from max_1. Our standard notation for this shape is L420[1]R2042, indicating its Lcut, Rcut and that these are taken from max_1. The use of [1] here opens the way to extend this style of specification to pipelines of arbitrary depth. A circuit implementation of this shape is published as BAF1 in [9]. The 14 cutaway states are shaded and their incoming and outgoing arcs deleted. No deleted state is reachable from the initial state.

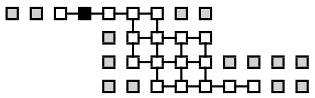


Fig. 10. shape L420[1]R2042

D. *$\mathcal{R}$: the Rcut lattice*

The lattice of Rcuts has 140 elements Rwxyz. It splits naturally into 5 levels, named RR0, RR1, ..., RR4, with all the cuts at level RRx possessing the same x index. Lattice level RR0, shown in Figure 11, is characterised by triangles T5..T1, RR1 by triangles T5..T2, ..., and RR4 by just T5.

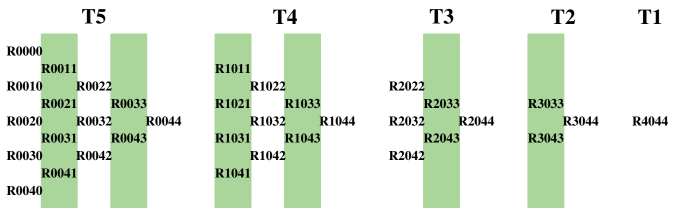


Fig. 11. RR0: wedge representation of a portion of the 140 Rcuts

Table I characterises Lcut wedge LL and the 5 Rcut wedges in turn with using separate columns to specify their extent first in triangle notation, then corner notation, and finally notes their respective sizes (number of cuts in this wedge).

TABLE I
HOMOGENEOUS RCUT CHARACTERISTICS

Wedge	Extent	Corner points			Size
LL	T5..T1	L000	L400	L444	35
RR0	T5..T1	R0000	R0040	R4044	35
RR1	T5..T2	R1111	R1151	R4155	34
RR2	T5..T3	R2222	R2262	R4266	31
RR3	T5..T4	R3333	R3373	R4377	25
RR4	T5	R4444	R4484	R4488	15

There are 35 Lcuts and 140 Rcuts. The complete design space has 4900 shapes including max_1, not all of which are live. Shapes Labc[1]Rwxyz deadlock when their Lcuts and Rcuts abut or overlap. For example, L440[1]R4444 is not live since all the states in row 2 of its shape have been deleted. A shape's liveness may be calculated directly from its cuts.

E. Design spaces

We can generate related design families by choosing structured subsets from Lcuts and Rcuts. These subsets relate to protocol classes that are characterised by various timing assumptions. The most robust protocols are the *untimed* which assume unbounded circuit delays [7], [12]. *DI—delay-insensitive* also assumes unbounded wire delays; *SI—speed-independent* assumes negligible wire delays. The remaining cuts are *timed*. Timed cuts are partitioned into two classes: "locally timed" (LT for short) and "architecturally timed" (AT). Fundamental mode, burst-mode [4], [6], and pulse based systems [19] are examples of LT cuts. In such systems the internal (local) delays of the circuit implementation must be faster or have less skew than the external environmental response time. Timed LT shapes can be designed and validated using relative timing [18], [22]. AT cuts require timing relationships between external architectural elements. For example, an AT cut will require that in some state the input channel must always respond before the output channel. The discipline of AT design is unexplored.

The timing discipline of a cut Labc is syntactically transparent. It is untimed (UT for short) iff *a,b,c* are all even (10 cases); it is locally timed (LT for short) iff just *a* and/or *c* are odd (9 cases). The remaining 16 cuts are architecturally timed. They are shaded in Figure 7 and Figure 11.

F. Experiments conducted over homogeneous pipelines

We have conducted experiments composing all 4900 shapes S into linear homogeneous pipelines HP(S, d) for pipeline depths d of 1..12.

Let S = L[1]R and HP(S, d) = L'[d']R'.

Key properties noted in [21] for the untimed design space extend to the complete 4900 member design space.

1) **Liveness**: HP(S, d) is live iff S is live. Input and output signals preserve the invariants:
$$4\text{cycle}\ (\ ir\uparrow\ ,\ \overline{ia}\uparrow\ ,\ ir\downarrow\ ,\ \overline{ia}\downarrow\ )$$
$$4\text{cycle}\ (\ \overline{or}\uparrow\ ,\ oa\uparrow\ ,\ \overline{or}\downarrow\ ,\ oa\downarrow\ )$$
2) **Cut independence**: L' depends only on L; R' on R.
3) **Shape capacity**: depends only on its input cut R.
4) **Closure**: L' $\in$ Lcuts and R' $\in$ Rcuts. The Lcuts and Rcuts are closed over operation HP. Accordingly, when minimised, all live linear homogeneous (and mixed) pipelines may be expressed as Lcuts and Rcuts from some $max_{d'}$ by extending the shape notation to Labc[d']Rwxyz.

Modulo some minor regularity wobbles noted in [21], 7 distinct shape capacity indicators emerge when we consider the HP operation over all cuts.

TABLE II
CAPACITY INDICATORS

Rcut	Rcut of HP at depth d						Capacity indicator
	1	2	3	4	5	6	
R2222	R2222	R2222	R2222	R2222	R2222	R2222	FULL
R1032	R1032	R1032	R1032	R1032	R1032	R1032	FULL
R2143	R2143	R3254	R4365	R1032	R2143	R3254	3/4
R2243	R2243	R3364	R0032	R2243	R3364	R0032	2/3
R2244	R2244	R0022	R2244	R0022	R2244	R0022	1/2
R3254	R3254	R1032	R3254	R1032	R3254	R1032	1/2
R2155	R2155	R5266	R2155	R5266	R2155	R5266	1/2
R3265	R3265	R2153	R5376	R3265	R2153	R5376	1/3
R4365	R4365	R3254	R2143	R1032	R4365	R3254	1/4
R2266	R2266	R2266	R2266	R2266	R2266	R2266	NULL

If we use *c/s* to denote the increase in *c*apacity per added *s*tages, the well known indicators are 1/1 (full), 1/2 (half), and 0/1 (null) noted in [21]. Table II gives examples of 4 new possibilities: 3/4, 1/4, 2/3, 1/3. Eight of the above cuts are used in cases 3 and 4 in the Applications Section VII.

The following examples show the relation between pipeline depth and shape capacity, using the above Table. Lcut L000 always generates a homogeneous pipeline with Lcut L000 whatever the pipeline depth:

1) R2222 has full capacity and HP(L000[1]R2222, 6) is equivalent to L000[6]R2222
2) R2244 has half capacity and HP(L000[1]R2244, 6) is equivalent to L000[3]R2244
3) R2266 has null capacity and HP(L000[1]R2266, 6) is equivalent to L000[1]R2266

IV. Modelling mixed pipelines

In the next three sections, we present new work on mixed pipelines. In this section, we lay the groundwork by presenting the extra Rcuts that arise in mixed pipelines, relating the Lcut wedge to each of the four Rcut wedges that arise, and summarising our experimental results over mixed pipelines.

A. Introduction

Our interest was sparked by our noting that locally timed (LT) circuits may produce more efficient implementations than untimed (UT) circuits. This suggested investigating the possibility of constructing pipelines as in figure 12 from an inner pipeline of selected efficient LT shapes which is then *bookended* by UT shapes selected so that the augmented pipeline interface is untimed. The inner LT protocols will retain any performance and power benefits, against which their additional timing constraints will need to be validated. But having an untimed interface is very attractive for modular system composition.

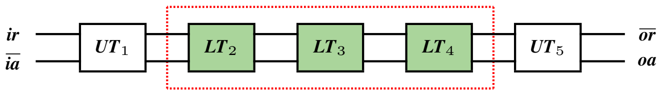


Fig. 12. Mixed pipe: a core of LT shapes bookended by UT shapes

Figure 13 shows the composition *S3* of two core shapes *S1* and *S2* with cut pairs (L1, R1) and (L2, R2) respectively. If live, *S1* will maintain its output 4cycle ($\overline{or}\uparrow$, $oa\uparrow$, $\overline{or}\downarrow$, $oa\downarrow$) and *S2* will maintain its input 4cycle ($ir\uparrow$, $\overline{ia}\uparrow$, $ir\downarrow$, $\overline{ia}\downarrow$). Since these signals share handshake wires appropriately, if *S1* is live and *S2* is live, then *S3* will be live. Note that we may mix live shapes freely to form live pipelines.

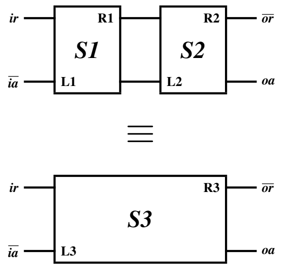


Fig. 13. *S3* = MP[*S1*,*S2*]

To specify a mixed pipeline, we have to spell out each and every constituent shape in order. This is easiest to express as a list: here *S3* = MP[*S1*, *S2*]. As indicated in Figure 13, *S3* has the cut pair (L3, R3).

As expected from previous work on 2phase controllers [2], the Lcuts are closed under MP, the Rcuts are not. For example MP[L000[1]R2222, L000[1]R4044] returns L000[2]R6266. In total, there are 15 such extra cuts ranging from R5155..R7377 which arise from mixed compositions. These extra Rcuts are only live at pipeline depths of 2 or more.

Note that max_1 may tolerate a loss of 4 states at the right hand end of each of its rows and remain live. The shape of this max_0 = L000[1]R4444 is shown in Figure 14.

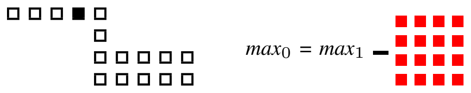


Fig. 14. Shape for max_0

Taking Rcuts R4444..R4488 from max_1 generates shapes that are equivalent to shapes produced by taking Rcuts R0000..R0044 from max_0. Thus including shape max_0 allows us to dispense with all 15 RR4 cuts. On replacing them by the 15 Rcuts R5155..R7377, we find the latter distribute perfectly over wedges RR1..RR3 in that they flesh out RR1, RR2, RR3 wedges to be isomorphic with the RR0 wedge, as indicated in Table III.

TABLE III
Mixed Rcut characteristics

Wedge	Extent	Corner points			Size
LL	T5..T1	L000	L400	L444	35
RR0	T5..T1	R0000	R0040	R4044	35
RR1	T5..T1	R1111	R1151	R5155	35
RR2	T5..T1	R2222	R2262	R6266	35
RR3	T5..T1	R3333	R3373	R7377	35

Subject to extent, each Rcut Rwxyz in wedge RRx has a simple relationship with Lcut Labc belonging to LL:

$$R_{wxyz} \approx L_{(y-x)(z-x)(w-x)}$$

As examples, R1032, R2143, R3254 and R4365 in wedges RR0..RR3 respectively all relate to L321.

There is also a simple relationship between the RRx wedges. To obtain the corresponding Rcut one level below, just add 1 to each index.

B. Experimental mixed pipeline results over S3 = MP[S1,S2]

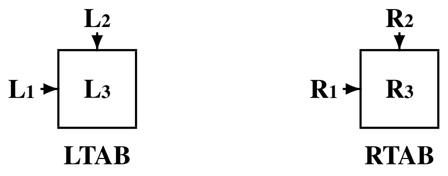
1) **Liveness**: S3 is live iff S1 is live and S2 is live.
2) **Closure**: L3 $\in \mathcal{L}$ and R3 $\in \mathcal{R}$. Thus, as with arbitrary length homogeneous pipelines, we will be able to specify minimised arbitrary mixed pipelines as cuts from some max_d.
3) **Independence**: The input cut R3 depends solely upon input cuts R1 and R2. The output cut L3 depends solely upon output cuts L1 and L2.
We have conducted exhaustive experiments to show the independence of Lcuts and Rcuts for homogeneous 4phase pipelines, and for mixed 4phase pipelines over all combinations of untimed and locally timed Lcuts and Rcuts, i.e. for established current design practice.
Rider: Independence holds over all cuts in the illustrative examples shown in Section VII (any many more besides). Work is underway to verify these properties over the full design space.

Independence means that given a mixed pipeline PIPEn MP[S1, S2, ..., Sn] of n shapes, we may calculate the Lcut of PIPEn from the Lcuts of the pipeline shapes only and calculate the Rcut of PIPEn from the Rcuts of the pipeline shapes only. We use the shorthand notation Lp = L1.L2....Ln and Rp = R1.R2...Rn for these calculations.

4) **Association**: Assuming the Rider regarding cut independence, Lcuts and Rcuts associate. For any Lcuts, La, Lb, Lc (La.Lb).Lc = La.(Lb.Lc) and (Ra.Rb).Rc = Ra.(Rb.Rc). Associativity facilitates composing mixed pipelines in suitable chunks.
5) **Identities**: $L_{321}.L_{abc} = L_{abc}$ and $L_{abc}.L_{321} = L_{abc}$ and $R_{1032}.R_{wxyz} = R_{wxyz}$ and $R_{wxyz}.R_{1032} = R_{wxyz}$
Note that L_{321} and R_{1032} are locally timed cuts.

C. LTAB and RTAB

Assuming Lcut and Rcut independence, we can generate tables LTAB and RTAB with dimensions 35×35 and 140×140 respectively instead of a single table of size 4900×4900.



One may then calculate mixed pipeline specifications of arbitrary length by iteration. As a simple example, consider a mixed pipe of length 4: PIPE4 = MP [SH1, SH2, SH3, SH4] where shape SHn = Ln[1]Rn for n = 1..4. Then by cut independence, the Lcut of PIPE4 may be calculated by LTAB[LTAB[LTAB[1,2],3],4], in shorthand ((L1.L2).L3).L4; or L1.(L2.(L3.L4)) by associativity ; or (L1.L2).(L3.L4) by chunking two smaller pipelines. Similarly for the Rcut of PIPE4.

V. The characterisation of LTAB:$\mathcal{L}\times\mathcal{L}$

In this section we examine the structure of the 35×35 LTAB [Labc,Lxyz] looking for patterns that build insight and develop them to yield a simple equivalent formula.

Table IV shows LTAB[L_{abc},L_{xyz}] in full. Each of its 35 rows are distinct. Four major patterns arise, namely VERTEX, EDGE, FACE, and CENTROID, as set out in Table V.

Table V groups LTAB rows L100, L200, L300 together as E01 and tells us that these three rows have only five distinct Lcuts L_{000}, L_{100}, L_{200}, L_{300}, L_{400}. However, each of these rows has a different pattern, say P1 for L100, P2 for L200, and P3 for L300. If we parameterise these patterns, then EDGE rows L100, L110, L111, ..., L441 have pattern P1; L200, L220, L222, ..., L442 have pattern P2; and L300, L330, L333, ..., L443 have pattern P3. The four FACE groups also boast just three distinct parameterised patterns. This structure of four vertices, six edges, four faces, and one centroid maps neatly onto a tetrahedron, hence the terminology.

1) Labc ∈ **VERTEXr**: each entry in a VERTEX row Labc is the corresponding Lcut L_{abc}.

TABLE V
Classifying Lcuts in LTAB

VERTEXr	=	Vr
EDGErs	=	Vr + Vs + Ers
FACErst	=	Vr + Vs + Vt + Ers + Ert + Est + Frst
CENTROID	=	all the Lcuts
where		
V0	=	{ L000 }
V1	=	{ L400 }
V2	=	{ L440 }
V3	=	{ L444 }
E01	=	{ L100, L200, L300 }
E02	=	{ L110, L220, L330 }
E03	=	{ L111, L222, L333 }
E12	=	{ L410, L420, L430 }
E13	=	{ L411, L422, L433 }
E23	=	{ L441, L442, L443 }
F012	=	{ L210, L310, L320 }
F013	=	{ L211, L311, L322 }
F023	=	{ L221, L331, L332 }
F123	=	{ L421, L431, L432 }
CENT	=	{ L321 }

2) Labc ∈ **EDGErs**: an entry in EDGE row Labc may be a vertex Vr or Vs or any of the three entries in Ers.
3) Labc ∈ **FACErst**: each entry in FACE row Labc may be from one of three vertices, three edges, or Frst.
4) Labc in **CENTROID**: every Lcut appears once echoing L_{xyz}.

Notice that the **VERTEXr** rows contain indices constructed from 0 or 4 only; the **EDGErs** rows contain indices constructed from 0, 4 and one digit from {1,2,3}; the **FACErst** rows contain indices constructed from 0, 4 and two different digits from {1,2,3}. Only the **CENTROID** contains indices constructed from 0, 4 and each of 1, 2, and 3. So one can tell much of the variety in a row of LTAB by inspection.

A. LTAB formula

When we examined LTAB row by row, we found just six interesting patterns (the vertex and centroid rows are trivial). Our initial approach was to try to characterise each row of LTAB separately. That is given LTAB[L_{abc}, L_{xyz}] returns L_{pqr}, fix abc, and seek a pattern for indices pqr in terms of the 35

TABLE VI
Lpqr = LTAB[L322,Lxyz]: per distinct Lpqr

Lpqr	⟸	[Lxyz] such that LTAB[L322, Lxyz] = Lpqr
L000	⟸	[L000]
L400	⟸	[L400]
L444	⟸	[L440, L441, L442, L443, L444]
L100	⟸	[L100]
L200	⟸	[L200]
L300	⟸	[L300]
L111	⟸	[L110, L111]
L222	⟸	[L220, L221, L222]
L333	⟸	[L330, L331, L332, L333]
L411	⟸	[L410, L411]
L422	⟸	[L420, L421, L422]
L433	⟸	[L430, L431, L432, L433]
L211	⟸	[L210, L211]
L311	⟸	[L310, L311]
L322	⟸	[L320, L321, L322]

TABLE IV
LTAB: Labc×Lxyz

L1\L2	L000	L100	L200	L300	L400	L110	L210	L310	L410	L111	L211	L311	L411	L220	L320	L420	L221	L321	L421	L222	L322	L422	L330	L331	L332	L333	L430	L431	L432	L433	L440	L441	L442	L443	L444
L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000	L000
L100	L000	L000	L000	L000	L000	L000	L000	L000	L000	L100	L100	L100	L100	L000	L000	L000	L100	L100	L100	L200	L200	L200	L000	L100	L200	L300	L000	L100	L200	L300	L000	L100	L200	L300	L400
L200	L000	L000	L000	L000	L000	L100	L100	L100	L100	L100	L100	L100	L100	L200	L200	L200	L200	L200	L200	L200	L200	L200	L300	L300	L300	L300	L300	L300	L300	L300	L400	L400	L400	L400	L400
L300	L000	L100	L200	L300	L400	L100	L200	L300	L400	L100	L200	L300	L400	L200	L300	L400	L200	L300	L400	L200	L300	L400	L300	L300	L300	L300	L400	L400	L400	L400	L400	L400	L400	L400	L400
L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400	L400
L110	L000	L000	L000	L000	L000	L000	L000	L000	L000	L110	L110	L110	L110	L000	L000	L000	L110	L110	L110	L220	L220	L220	L000	L110	L220	L330	L000	L110	L220	L330	L000	L110	L220	L330	L440
L210	L000	L000	L000	L000	L000	L100	L100	L100	L100	L110	L110	L110	L110	L200	L200	L200	L210	L210	L210	L220	L220	L220	L300	L310	L320	L330	L300	L310	L320	L330	L400	L410	L420	L430	L440
L310	L000	L100	L200	L300	L400	L100	L200	L300	L400	L110	L210	L310	L410	L200	L300	L400	L210	L310	L410	L220	L320	L420	L300	L310	L320	L330	L400	L410	L420	L430	L400	L410	L420	L430	L440
L410	L400	L400	L400	L400	L400	L400	L400	L400	L400	L410	L410	L410	L410	L400	L400	L400	L410	L410	L410	L420	L420	L420	L400	L410	L420	L430	L400	L410	L420	L430	L400	L410	L420	L430	L440
L111	L000	L000	L000	L000	L000	L000	L000	L000	L000	L111	L111	L111	L111	L000	L000	L000	L111	L111	L111	L222	L222	L222	L000	L111	L222	L333	L000	L111	L222	L333	L000	L111	L222	L333	L444
L211	L000	L000	L000	L000	L000	L100	L100	L100	L100	L111	L111	L111	L111	L200	L200	L200	L211	L211	L211	L222	L222	L222	L300	L311	L322	L333	L300	L311	L322	L333	L400	L411	L422	L433	L444
L311	L000	L100	L200	L300	L400	L100	L200	L300	L400	L111	L211	L311	L411	L200	L300	L400	L211	L311	L411	L222	L322	L422	L300	L311	L322	L333	L400	L411	L422	L433	L400	L411	L422	L433	L444
L411	L400	L400	L400	L400	L400	L400	L400	L400	L400	L411	L411	L411	L411	L400	L400	L400	L411	L411	L411	L422	L422	L422	L400	L411	L422	L433	L400	L411	L422	L433	L400	L411	L422	L433	L444
L220	L000	L000	L000	L000	L000	L110	L110	L110	L110	L110	L110	L110	L110	L220	L220	L220	L220	L220	L220	L220	L220	L220	L330	L330	L330	L330	L330	L330	L330	L330	L440	L440	L440	L440	L440
L320	L000	L100	L200	L300	L400	L110	L210	L310	L410	L110	L210	L310	L410	L220	L320	L420	L220	L320	L420	L220	L320	L420	L330	L330	L330	L330	L430	L430	L430	L430	L440	L440	L440	L440	L440
L420	L400	L400	L400	L400	L400	L410	L410	L410	L410	L410	L410	L410	L410	L420	L420	L420	L420	L420	L420	L420	L420	L420	L430	L430	L430	L430	L430	L430	L430	L430	L440	L440	L440	L440	L440
L221	L000	L000	L000	L000	L000	L110	L110	L110	L110	L111	L111	L111	L111	L220	L220	L220	L221	L221	L221	L222	L222	L222	L330	L331	L332	L333	L330	L331	L332	L333	L440	L441	L442	L443	L444
L321	L000	L100	L200	L300	L400	L110	L210	L310	L410	L111	L211	L311	L411	L220	L320	L420	L221	L321	L421	L222	L322	L422	L330	L331	L332	L333	L430	L431	L432	L433	L440	L441	L442	L443	L444
L421	L400	L400	L400	L400	L400	L410	L410	L410	L410	L411	L411	L411	L411	L420	L420	L420	L421	L421	L421	L422	L422	L422	L430	L431	L432	L433	L430	L431	L432	L433	L440	L441	L442	L443	L444
L222	L000	L000	L000	L000	L000	L111	L111	L111	L111	L111	L111	L111	L111	L222	L222	L222	L222	L222	L222	L222	L222	L222	L333	L333	L333	L333	L333	L333	L333	L333	L444	L444	L444	L444	L444
L322	L000	L100	L200	L300	L400	L111	L211	L311	L411	L111	L211	L311	L411	L222	L322	L422	L222	L322	L422	L222	L322	L422	L333	L333	L333	L333	L433	L433	L433	L433	L444	L444	L444	L444	L444
L422	L400	L400	L400	L400	L400	L411	L411	L411	L411	L411	L411	L411	L411	L422	L422	L422	L422	L422	L422	L422	L422	L422	L433	L433	L433	L433	L433	L433	L433	L433	L444	L444	L444	L444	L444
L330	L000	L110	L220	L330	L440	L110	L220	L330	L440	L110	L220	L330	L440	L220	L330	L440	L220	L330	L440	L220	L330	L440	L330	L330	L330	L330	L440	L440	L440	L440	L440	L440	L440	L440	L440
L331	L000	L110	L220	L330	L440	L110	L220	L330	L440	L111	L221	L331	L441	L220	L330	L440	L221	L331	L441	L222	L332	L442	L330	L331	L332	L333	L440	L441	L442	L443	L440	L441	L442	L443	L444
L332	L000	L110	L220	L330	L440	L111	L221	L331	L441	L111	L221	L331	L441	L222	L332	L442	L222	L332	L442	L222	L332	L442	L333	L333	L333	L333	L443	L443	L443	L443	L444	L444	L444	L444	L444
L333	L000	L111	L222	L333	L444	L111	L222	L333	L444	L111	L222	L333	L444	L222	L333	L444	L222	L333	L444	L222	L333	L444	L333	L333	L333	L333	L444	L444	L444	L444	L444	L444	L444	L444	L444
L430	L400	L410	L420	L430	L440	L410	L420	L430	L440	L410	L420	L430	L440	L420	L430	L440	L420	L430	L440	L420	L430	L440	L430	L430	L430	L430	L440	L440	L440	L440	L440	L440	L440	L440	L440
L431	L400	L410	L420	L430	L440	L410	L420	L430	L440	L411	L421	L431	L441	L420	L430	L440	L421	L431	L441	L422	L432	L442	L430	L431	L432	L433	L440	L441	L442	L443	L440	L441	L442	L443	L444
L432	L400	L410	L420	L430	L440	L411	L421	L431	L441	L411	L421	L431	L441	L422	L432	L442	L422	L432	L442	L422	L432	L442	L433	L433	L433	L433	L443	L443	L443	L443	L444	L444	L444	L444	L444
L433	L400	L411	L422	L433	L444	L411	L422	L433	L444	L411	L422	L433	L444	L422	L433	L444	L422	L433	L444	L422	L433	L444	L433	L433	L433	L433	L444	L444	L444	L444	L444	L444	L444	L444	L444
L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440	L440
L441	L440	L440	L440	L440	L440	L440	L440	L440	L440	L441	L441	L441	L441	L440	L440	L440	L441	L441	L441	L442	L442	L442	L440	L441	L442	L443	L440	L441	L442	L443	L440	L441	L442	L443	L444
L442	L440	L440	L440	L440	L440	L441	L441	L441	L441	L441	L441	L441	L441	L442	L442	L442	L442	L442	L442	L442	L442	L442	L443	L443	L443	L443	L443	L443	L443	L443	L444	L444	L444	L444	L444
L443	L440	L441	L442	L443	L444	L441	L442	L443	L444	L441	L442	L443	L444	L442	L443	L444	L442	L443	L444	L442	L443	L444	L443	L443	L443	L443	L444	L444	L444	L444	L444	L444	L444	L444	L444
L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444	L444

xyz from the cuts appearing in row Lwyz. Table VI considers FACE row L322 of LTAB and displays its 15 possible distinct Lcuts in its left column headed Lpqr and on the right, those Lxyz cuts which map to it. Table VI lists the pattern found for each row in LTAB listed as *row* → *pattern*.

The L322 pattern is simple: pqr = xyy. We now apply this technique to all 35 rows of LTAB. Table VII supplies the KEY to unlock generality. By inspection, we now have a single formula that fits all rows Labc in LTAB.

TABLE VII
Row by row formulae for LTAB

KEY: [0 → 0, 1 → z, 2 → y, 3 → x, 4 → 4]				
L000 → L000	L100 → Lz00	L200 → Ly00	L300 → Lx00	L400 → L400
	L110 → Lzz0	L210 → Lyz0	L310 → Lxz0	L410 → L4z0
	L111 → Lzzz	L211 → Lyzz	L311 → Lxzz	L411 → L4zz
		L220 → Lyy0	L221 → Lyyz	L222 → Lyyy
		L320 → Lxy0	L321 → Lxyz	L322 → Lxyy
		L420 → L4y0	L421 → L4yz	L422 → L4yy
	L330 → Lxx0	L331 → Lxxz	L332 → Lxxy	L333 → Lxxx
	L430 → L4x0	L431 → L4xz	L432 → L4xy	L433 → L4xx
L440 → L440	L441 → L44z	L442 → L44y	L443 → L44x	L444 → L444

In LTAB[Labc,Lxyz] ⇒ Lpqr, [0, z, y, x, 4] provides the *range* of indices in Lpqr[1] and [a, b, c] provides the *selectors* to pick p, q, r from the range. Hence the formulae quoted below can be used instead of LTAB:

$$\text{LTAB}[\text{L}_{abc},\ \text{L}_{xyz}] = \text{PICK}\ [\ a,\ b,\ c\ ]\ [\ 0,\ z,\ y,\ x,\ 4\ ]$$

As examples,

[1]The sequence 0, z, y, x, 4 is monotonically increasing.

LTAB [L322, L300]	=	PICK	[3, 2, 2]	[0, 0, 0, 3, 4]	returns L300
LTAB [L322, L211]	=	PICK	[3, 2, 2]	[0, 1, 1, 2, 4]	returns L211
LTAB [L322, L431]	=	PICK	[3, 2, 2]	[0, 1, 3, 4, 4]	returns L433

VI. The characterisation of RTAB:$\mathcal{R}\times\mathcal{R}$

In this section, we examine the structure of the 140×140 RTAB [Rabcd, Rwxyz]. In order to build upon the insights generated in the previous section, we introduce a change of Rcut notation that relates closely to the Lcut notation (it uses exactly the same cut indices as the Lcuts) and made the initial exploratory work much easier. It is simple to swap between the two Rcut notations.

TABLE VIII
Mixed Rcut characteristics

Wedge	Extent	Corner points			Size
LL	T5..T1	L000	L400	L444	35
RR0	T5..T1	R0000	R0040	R4044	35
RR1	T5..T1	R1111	R1151	R5155	35
RR2	T5..T1	R2222	R2262	R6266	35
RR3	T5..T1	R3333	R3373	R7377	35
W	T5..T1	W000	W400	W444	35
X	T5..T1	X000	X400	X444	35
Y	T5..T1	Y000	Y400	Y444	35
Z	T5..T1	Z000	Z400	Z444	35

The four cuts R1032, R2143, R3254, R4365 lie on wedges RR0, RR1, RR2, RR3 respectively in identical positions (the centroid of their respective T4 triangles). Informally we move from one cut to the next by simply incrementing its indices wxyz by one each. All Rcuts in wedge RR0 have a zero x index: R_{w0yz}. Suppose we rotate the indices from R_{w0yz} to R_{0yzw}. We may then associate the 0 with R $(R0)_{yzw}$. The typography is easier to handle if we change the wedge names to single letters W, X, Y, Z resulting in W_{321}, X_{432}, Y_{543} and Z_{654}. If in addition, we decrement all X indices by 1, all Y indices by 2, and all Z indices by 3, the cuts become W_{321}, X_{321}, Y_{321}, Z_{321}. In this notation, the L, W, X, Y, Z lattices have identical structures and identical indices, as indicated in the third level of Table VIII. To move from one $\mathcal{R}$ lattice level to another we merely change its wedge indicator. We use this more mathematical notation when relating LTAB and RTAB, and the RR notation for pipeline experiments. It is easy to convert between the two notations.

RTAB has 19600 elements, too large to be tackled without structuring. We chose to partition RTAB in terms of wedges (W, X, Y, Z) rather than (RR0, RR1, RR2, RR3) as it made the initial visual search for patterns much easier. It is simple to switch between these notations if using the 4-index R_{wyxz} on practical problems. When partitioned as (W,X,Y,Z)×(W,X,Y,Z), RTAB splits naturally into 16 35×35 blocks we label as shown in Figure 15.

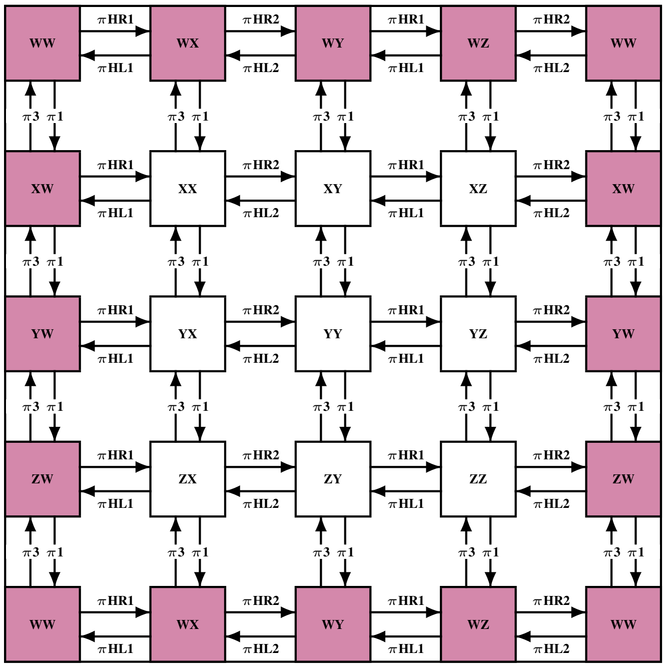


Fig. 15. Complete WXYZ RTAB transformations

Figure 15 is marked with labelled arrows between blocks. Each arrow represents a transformation (basically a row permutation) between adjacent blocks. For ease of diagramming, the blocks in column 5 repeat those of column 1, and the blocks in row 5 repeat those of row 1, so the figure is a flat representation of doughnut (toroidal) structure.

There is a simple isomorphism between LTAB and WW. Given the function *LtoW* which takes Lcut L_{abc} and returns W_{abc}, then WW is the transpose of LTAB with LtoW applied to every element: WW = APPLY LtoW (TRANSPOSE LTAB).

The four 35×140 row blocks, WW..WZ. XW..XZ, YW..YZ, ZW..ZZ are also simply related. Displacing each element in WW..WZ by incrementing its wedge indicator and leaving its indices untouched returns row blocks XW..XZ and so on.

Hence we are left with the simpler task of formulae for just WW, WX, WY, and WZ.

TABLE IX
LABELS FOR THE 140 DISTINCT ROWS OF RTAB

W000	W100	W200	W300	W400	W500	W110	...	W444
X000	X100	X200	X300	X400	X500	X110	...	X444
Y000	Y100	Y200	Y300	Y400	Y500	Y110	...	Y444
Z000	Z100	Z200	Z300	Z400	Z500	Z110	...	Z444

Consider HEX = WW ++ WX ++ ... ++ ZZ. Then HEX has the full 19600 members of RTAB now arranged as 560 rows each of length 35. In each of these 560 rows, the entries in each row belong to the same W, X, Y or Z wedge. There are 140 distinct such rows, each of which occurs exactly four times. As with LTAB, they have a straightforward labelling which corresponds to their entry in the column labelled $W_{321}/X_{321}/Y_{321}/Z_{321}$ (only one of these tags will arise). Each of the 16 hexes has 35 rows. As labelled in Table IX, each distinct index will appear once in each hex, but may be associated with wedge W or X or Y or Z.

A. *RTAB formulae*

It suffices to generate PICK based formulae for WW, WX, WY, and WZ. Here we detail the formulae for WW and WX.

1) Case WW: In the WW block, all Rcuts belong to the W wedge. We first examined WW row by row, looking for a formula, but this time in PICK notation. For Rcuts, *Wabc gives the range* and *Wxyz gives the cut choices*: to be expected since WW is the transpose of LL.

We require that PICK extracts pqr and the appropriate wedge W, X, Y, Z for the result. Here are 4 examples taken from rows W110, W210, W311, and W432 of WW; followed by their generalisation. We remind you that PICK selects from columns encoded 0..4. In the 4 examples, we use WPICK as a reminder of the wedge expected.

In this and the next two tables, we save space by using Wabc.Rxyz to abbreviate RTAB[Wabc,Rxyz].

WW block: RTAB[Wabc,Wxyz]

W110.Wxyz	=	WPICK	[x, y, z]	[0, 0, 1, 1, 4]
W210.Wxyz	=	WPICK	[x, y, z]	[0, 0, 1, 2, 4]
W311.Wxyz	=	WPICK	[x, y, z]	[0, 1, 1, 3, 4]
W432.Wxyz	=	WPICK	[x, y, z]	[0, 2, 3, 4, 4]
Wabc.Wxyz	=	PICK	[x, y, z]	[0, c , b, a , R]
		where R = 4		

Above, R denotes the appropriate wedge for the resulting Rcut R_{pqr}. This detail generalises to all formulae if we encode R with (W, 4 or 0), (X, 3), (Y, 2), (Z, 1).

```
                         WX block: RTAB[Wabc,Xxyz]

W310.Xxyz  =  WPICK  [  x,  y,  z  ] [   0,    1,    3,    4,   4  ]
                                     [   c,    b,    a,    W ,  4  ]
W311.Xxyz  =  XPICK  [  x,  y,  z  ] [  0 ,   0 ,   2,    3 ,  4  ]
                                     [  c-1,  b-1,  a-1,  X ,  4  ]
W322.Xxyz  =  YPICK  [  x,  y,  z  ] [   0,    0,    1,    2,   4  ]
                                     [  c-2,  b-2,  a-2,  Y ,  4  ]
W333.Xxyz  =  ZPICK  [  x,  y,  z  ] [   0,    0,    0,    1,   4  ]
                                     [  c-3,  b-3,  a-3,  Z ,  4  ]

Wabc.Xxyz  =    PICK  [  x,  y,  z  ] [   0,   b-c,  a-c,   R,   4  ]
                          where R = 4-c
```

2) Case WX: Note the second Rcut is always some Xxyz. For this example, we consider Wabc rows W310, W311, W322, and W333. Their respective abc indices occur in range columns 2, 1, 0 respectively but appear as c-c, b-c, a-c, so that the first range index is always zero. We have annotated each range with its interpretation on the line below. Notice that the wedge level is defined by the c index of Wabc.

3) Formulae for WW, WX, WY, and WZ: The remaining block formulae for WY and WZ follow the pattern laid down for WY and WX in that it is possible for PICK to extract the appropriate wedge as well as its indices for every entry these subtables.

With the definition: **m⊗n = if m<n then 4+(m-n) else m-n**, the formulae for each block in the top row of RTAB are:

```
Wabc.Wxyz  =  PICK [ x, y, z ] [  0,    c,     b,     a,    R  ]  with R = 4
Wabc.Xxyz  =  PICK [ x, y, z ] [  0,  b⊗c,  a⊗c,    R,    4  ]  with R = 4-c
Wabc.Yxyz  =  PICK [ x, y, z ] [  0,  a⊗b,    R,   c⊗b,  4  ]  with R = 4-b
Wabc.Zxyz  =  PICK [ x, y, z ] [  0,    R,   c⊗a,  b⊗a,  4  ]  with R = 4-a
```

We now have formulae for every entry in the top row block WW..WZ. It is a simple matter to use these 4 formulae to calculate the entries in the remaining row blocks given the simple (vertical) relationship between them.

A simple way of working with 4 index Rcuts, is to convert them to 3 index notation, use the above formulae, and then convert the back to 4 index notation.

VII. Application: calculating mixed pipelines

We now present case studies. The following four are each over mixed pipelines of depth 8.

MP [S1, S2, S3, S4, S5, S6, S7, S8]

Each case is tabulated over five lines. The results are given in the tables below, depth by depth, as the pipelines grow. Per column n, lines one and two give Ln and Rn, the Lcut and Rcut of shape Sn = Ln[1]Rn added to the pipe at depth n. Lines 3..5 give the Lcut, state size, and Rcut of the mixed pipeline at this depth.

The first two cases use the same Lcut sequence and demonstrate the claim that Lcuts work independently as we use the same Lcut sequence in both cases.

The Lcut sequences contain no untimed cuts, 3 locally timed cuts, and 5 architecturally timed cuts and serve to support the claim for Lcut independence. The Rcuts are equally varied and each is of full capacity.

CASE 1:

SH.Lcuts	[	L320,	L331,	L322,	L210,	L310,	L311,	L211,	L221]
SH.Rcuts	[	R1132,	R2032,	R1142,	R0032,	R2132,	R1031,	R2133,	R1033]
LP.Lcuts	[	L320,	L330,	L330,	L220,	L110,	L110,	L110,	L110]
LP.size	[	20	34	47	65	83	99	112	128]
LP.Rcuts	[	R1132,	R2132,	R2252,	R2252,	R2252,	R2252,	R2255,	R2255]

CASE 2:

SH.Lcuts	[	L320,	L331,	L322,	L210,	L310,	L311,	L211,	L221]
SH.Rcuts	[	R0021,	R3143,	R0031,	R1143,	R1021,	R2142,	R1022,	R2144]
LP.Lcuts	[	L320,	L330,	L330,	L220,	L110,	L110,	L110,	L110]
LP.size	[	24	34	52	68	90	106	126	134]
LP.Rcuts	[	R0021,	R2042,	R0042,	R0044,	R0040,	R0040,	R0000,	R0044]

Since the state size of max_8 is 144, it is straightforward to check that the final pipelines are L110[8]R2255 and L110[8]R0044 respectively.

CASE 3:

SH.Lcuts	[	L000,	L000,	L000,	L000,	L000,	L000,	L000,	L000]
SH.Rcuts	[	R2222,	R2143,	R2243,	R3254,	R2155,	R3265,	R4365,	R4444]
LP.Lcuts	[	L000,	L000,	L000,	L000,	L000,	L000,	L000,	L000]
LP.size	[	24	36	48	56	64	68	72	72]
LP.Rcuts	[	R2222,	R2262,	R2266,	R2222,	R2266,	R2262,	R2222,	R2222]

CASE 4:

SH.Lcuts	[	L000,	L000,	L000,	L000,	L000,	L000,	L000,	L000]
SH.Rcuts	[	R4444,	R4365,	R3265,	R2155,	R3254,	R2243,	R2143,	R2222]
LP.Lcuts	[	L000,	L000,	L000,	L000,	L000,	L000,	L000,	L000]
LP.size	[	16	20	24	36	44	56	68	80]
LP.Rcuts	[	R4444,	R4044,	R0044,	R4044,	R0040,	R0044,	R4044,	R0000]

In cases 3 and 4, we have chosen Rcuts of full and half capacity, and then three pairs from (3/4, 1/4), (1/2, 1/2) and (2/3, 1/3) but in jumbled order. In both cases, the depth of the resulting mixed pipelines is four. To aid in seeing how these pipelines grow, we have used Lcut L000 only.

Since the state size of max_4 is 80, it is straightforward to check that the final pipelines are L000[4]R2222 and L000[4]R0000 respectively.

A. Bookending

Since L000 is a vertex cut, its corresponding row in LTAB (see Figure IV) contains only L000 entries and the column corresponding to R0000 in RTAB contains only R0000 entries. A simple way of bookending (Figure 12 would be to prefix and postfix a timed pipeline by max_0. As an example, we apply this to CASE1, but show only the Lcut and Rcut sequences:

CASE1B:

SH.Lcuts	[	L000,	L320,	L331,	L322,	L210,	L310,	L311,	L211,	L221	L000]
SH.Rcuts	[	R0000,	R1132,	R2032,	R1142,	R0032,	R2132,	R1031,	R2133,	R1033	RR0000]
LP.Lcuts	[	L000	L000,	L000,	L000,	L000,	L000,	L000,	L000,	L000,	L000]
LP.Rcuts	[	R0000,	R0000,	R0000,	R0040,	R0040,	R0040,	R0040,	R0044,	R0044	R0000]

Note that although the above pipeline contains 10 stages, it minimises to max_8. Other methods of converting CASE1 to an untimed interface would be to change the first Lcut from L320 to L000 and the last Rcut from R1033 to R0000.

We have conducted our experiments in CCS [14]. Shapes are expressible directly in CCS and easy to map into pipelines. CCS has a reliable public domain tool support, the CWB (Concurrency Workbench [15]), with built-in minimisation to the smallest equivalent state machine and the powerful modal μ-calculus property checking notation [20]. Both these attributes were necessary. The cases above were all mapped into CCS and run on the CWB. In each case, the LTAB/RTAB formulae predictions were confirmed.

B. Implementation

The final case study implements two 5-deep homogeneous pipelines and a 5-deep mixed pipeline following Figure 12. The LT and UT designs implement the L300[1]R3044 burst-mode [5] and L220[1]R0044 delay-insensitive protocols respectively. Both controllers are built using five static gates. The designs were synthesized using Petrify and Design Compiler, implemented to 65nm layout using Cadence EDI, and compared for power and performance using ModelSim and PrimeTime. The results of the design are shown below:

Design	**Cycle Time**	**Forward Latency**	**Ave. Energy Per Stage**
LT	345ps	111ps	893fJ
UT	458ps	114ps	549fJ
Mixed	410ps	114ps	710fJ

The delay-insensitive protocol is 33% slower than the burst-mode controller, but also expends 39% less energy. The mixed pipeline degrades performance over the pure LT pipeline, but is only 19% slower, and expends 20% less energy. More significantly, the burst-mode timing that results in increased performance is hidden inside the mixed pipeline. The UT controllers at the ends of the mixed pipeline present a robust delay-insensitive interface to the rest of the system with no relative timing assumptions, increasing modularity.

VIII. Summary

Here we present new experiment-based results on the structure of the complete 4phase family and the patterns of its design space culminating in succinct methods for calculating the behaviour of mixed linear pipelines of arbitrary depth.

The patterns arising have suggested algorithmic rules for predicting the behaviours of mixed and homogeneous pipelines. New 4phase patterns herein described include: complete lattices for $\mathcal{L}$ and $\mathcal{R}$ cuts and hence the complete design space as $\mathcal{L}\times\mathcal{R}$; a uniform way of expressing and relating the cut lattices; the seven item full..null range of shape capacities; a practical exploration of 4phase mixed pipelines; clarification of the mathematical structures behind mixed pipelines; functions for mapping lattices between $\mathcal{L}$ and $\mathcal{R}$ cuts; and the development of a concise and transparent method of calculating the behaviour of mixed pipelines in terms of protocol concurrency and storage capacity. Demonstrations of the latter's practical use were given by tested examples.

IX. Acknowledgments

Thanks are due to the asynchronous community who have made great efforts to document and explain their circuits so they are clear to the community at large. We also wish to thank the anonymous referees for their constructive criticisms and colleagues Robert Najvirt and Mike Stannett for their help, insights and guidance.

References

[1] Graham Birtwistle and Kenneth Stevens. The Family of 4-phase Latch Controllers. In Andrei Voronkov and Margarita Korovina, editors, *ASYNC 2008, 14th International Symposium on Asynchronous Circuits and Systems*, pages 34–65. EasyChair, 2014.

[2] Graham Birtwistle and Kenneth S. Stevens. A Design Space and its Patterns: Modelling 2phase Asynchronous Pipelines. In Andrei Voronkov and Margarita Korovina, editors, *HOWARD-60. A Festschrift on the Occasion of Howard Barringer's 60th Birthday*, pages 34–65. EasyChair, 2014.

[3] I. Blunno, J. Cortadella, A. Kondratyev, L.Lavagno, K. Lwin, and C. Sotiriou. Handshake protocols for de-synchronisation. In *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits*, pages 149–158. IEEE/ACM, April 2004.

[4] W. S. Coates, A. L. Davis, and K. S. Stevens. Automatic Synthesis of Fast Compact Self-Timed Control Circuits. In *IFIP Working Conference on Design Methodologies*, pages 193–208, April 1993.

[5] J. Cortadella, M. Kishinevsky, S. M. Burns, A. Kondratyev, L. Lavagno, K. S. Stevens, A. Taubin, and A. Yakovlev. Lazy transition systems and asynchronous circuit synthesis with relative timing assumptions. *IEEE Transactions on Computer-Aided Design*, 21(2):109–130, Feb 2002.

[6] A. Davis. Burst Mode Controllers: Synthesis and Experience. In A. Davis and G. Birtwistle, editor, *Proceedings VII Banff Workshop: Asynchronous Digital Circuit Design*, pages 104–150. Springer Verlag, Workshops in Computing Series, 1995.

[7] J. C. Ebergen. A Formal Approach to Designing Delay-Insensitive Circuits. *Distributed Computing*, 5(3):107–119, 1991.

[8] Aristides Efthymiou and Jim D. Garside. Adaptive pipeline structures for speculation control. In *Ninth International Symposium on Asynchronous Circuits and Systems*, pages 46–55. IEEE, May 2003.

[9] Stephen B. Furber. A small compendium of 4-phase macropipeline latch control circuits. Technical Report v0.3, 17/01/99, University of Manchester, Dept. of Computer Science, 1999.

[10] Andrew M. Lines. Pipelined asynchronous circuits. Master's thesis, California Institute of Technology, Pasadena, CA, 1998.

[11] JianWei Liu. Arithmetic and Control Components for an Asynchronous System. PhD Thesis, Department of Computer Science, University of Manchester, 1997.

[12] Alain J. Martin. The Design of a Delay-Insensitive Microprocessor: An Example of Circuit Synthesis by Program Transformation. In M. Leeser and G. Brown, editors, *Hardware Specification, Verification, and Synthesis: Mathematical Aspects, Proceedings of the Mathematical Sciences Institute Workshop, Cornell University, Ithaca, N.Y., July, 1989*, pages 244–259, New York, 1989. Springer-Verlag.

[13] Peggy B. McGee and Steven M. Nowick. A Lattice-Based Framework for the Classification and Design of Asynchronous Pipelines. In *Proceedings of the Digital Automation Conference (DAC05)*, pages 491–496. IEEE/ACM, June 2005.

[14] R. Milner. *Communication and Concurrency*. Prentice-Hall, London, 1989.

[15] Faron G. Moller and Perdita Stevens. *The Edinburgh Concurrency Workbench (Version 7)*. University of Edinburgh, October 1992.

[16] Robert Navjirt, Syed Rameez Naqvi, and Andreas Steininger. Classifying Virtual Channel Access Control Schemes for Asynchronous NoCs. In *International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 115–123, Santa Monica, CA, 19-22 May 2013.

[17] Steven M. Nowick and Montek Singh. High Performance Asynchronous Pipelines: An Overview. In *IEEE Design & Test of Computers*, pages 8–22. IEEE, 2011.

[18] Kenneth S. Stevens, Ran Ginosar, and Shai Rotem. Relative Timing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1(11):129–140, February 2003.

[19] Kenneth S. Stevens, Shai Rotem, Ran Ginosar, Peter Beerel, Chris J. Myers, Kenneth Y. Yun, Rakefet Kol, Charles Dike, and Marly Roncken. An Asynchronous Instruction Length Decoder. *IEEE Journal of Solid State Circuits*, 36(2):217–228, February 2001.

[20] C. Stirling. *Modal and Temporal Properties of Processes*. Springer Verlag, New York, Berlin, Heidelberg, 2001.

[21] Santosh Varanasi, Kenneth Stevens, and Graham Birtwistle. Concurrency Reduction of Untimed Latch Protocols – Theory and Practice. In *International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 26–37, Grenoble, France, May 2010.

[22] Yang Xu and Kenneth S. Stevens. Automatic Synthesis of Computation Interference Constraints for Relative Timing. In *26th International Conference on Computer Design*, pages 16–22. IEEE, Oct. 2009.