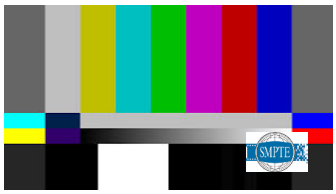
Efficient Carriage of Sub-Rasters With ST 2110-20

Paul Briscoe, Televisionary Consulting
Toronto Canada

On behalf of Evertz Microsystems

Sub-Rasters

- Let's define what a sub-raster is
  - "A raster which does not occupy the entire space of a full "Standard" video frame"
- A good example is a logo or "bug"
- Consists of a full-resolution background image with a superimposed smaller graphic added at the same resolution
- Unlike a video frame, these are arbitrarily-sized images.



How Sub-Rasters are Made and Consumed

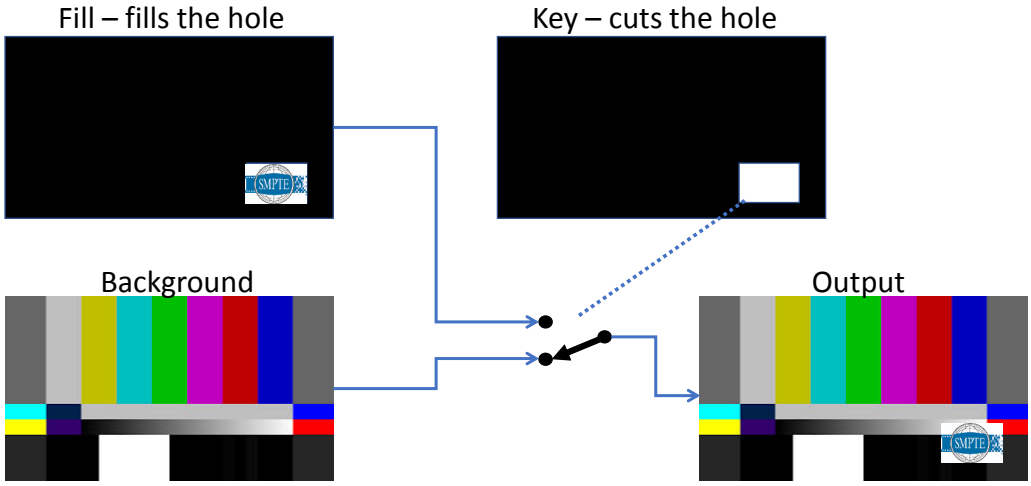
- Small pixel map, arbitrary shape and size created on a CGI workstation
  - Static or moving sequence, let's say a logo
- Rendered internally as two images (or image sequences):
  - The small graphic plus a white-on-black outline of the edges of the graphic (and holes)
- Images are positioned as required in actual keyed output raster
- In the keyer, the background feeds two multipliers whose outputs are added
  - Each multiplier is fed with one of the sub-rasters
  - One is multiplied by the value of the alpha channel
  - Other is multiplied by (1 *minus* the value of the alpha channel)
- The outputs are summed, yielding the mixed-in logo. This is a linear key.
- For purposes of this discussion, we will just hard switch background and fill

The System

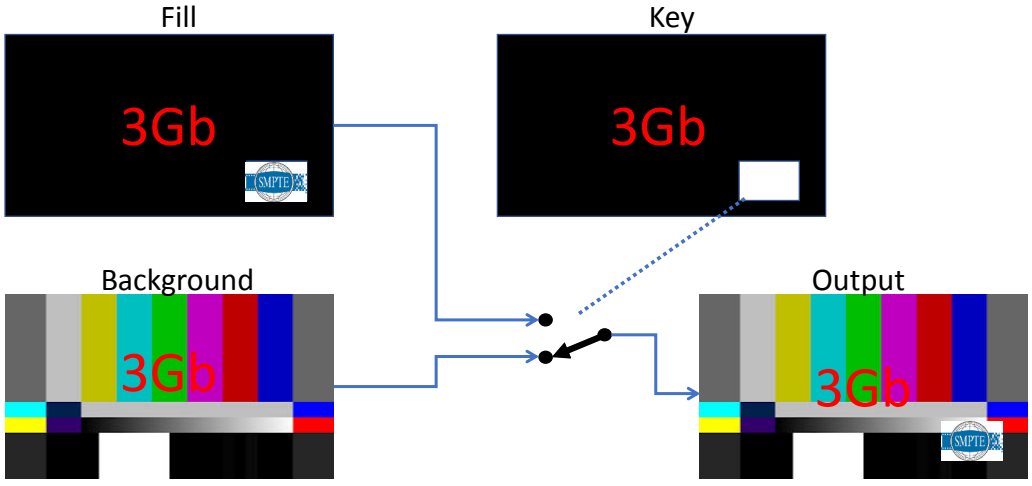
- Inside a purpose-build CGI / keying workstation, the sub-rasters exist as bitmaps
- They never leave the device

- In larger production systems, graphics generation is often common to many facilities, and sub-rasters need to be routed to many destinations
  - This is done using SDI (today)
  - It is very bit-inefficient
  - It is equally router port and cable inefficient
  - On-screen placement is 'cooked in' at time of generation

Simplified “Hard Cut” Logo Keying Model



Bandwidth Usage



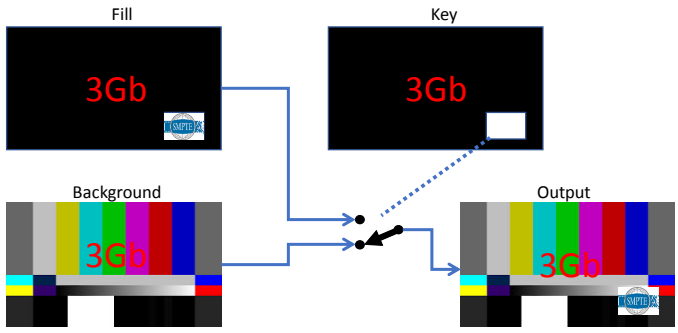
Why do we do this?

- Keying started life as a discrete function (= device)
- Full, timed rasters were needed in analog – simple switching
- Same in SDI (because it's really a digitized analog stream)
- "We've always done it this way"
- Specialty logo devices can hold key and fill subrasters
  - Keys them into full raster background internally
  - Internal processing positions the logo pixels in the desired location in raster
- We still often ship key and fill around from one production workflow to another via SDI

Bandwidth Usage

Using 1080p as example:
3 Gb Background +
3 Gb Fill +
3 Gb Key +
---
3 Gb Output
Not great efficiency
- 3 source cables in SDI

- What would we do in 2110?
- Same thing – 3 streams



Moving a Key Around on SDI

- A 4:2:2:4 stream on one cable:
  - Carries a full background-sized 4:2:2 raster of fill and a full Y-only raster of key - *not common* Two 4:2:2 streams on two cables:
- Two 4:2:2 streams on two cables
  - One carries a full background-sized 4:2:2 raster of fill
  - Other carries a full background-sized 4:2:2 raster of key
  - *Typical / legacy*

- *Very inefficient. If logo is 10% of picture area:*
  - *Fill bandwidth efficiency is 1%*
  - *Key bandwidth efficiency is 0.5% (empty chroma channels)*

  1.5% of 6 Gb is actually used.

Key / Fill in 2110

- In 2110 today we can do the same thing
  - Imagine gateway-ing the SDI model into IP streams
- In use now, works just fine
- Same bandwidth (in)efficiency as SDI
  - Lesser impact on switch port and link utilization

It would be bandwidth-beneficial if we could send the key and fill "atomically" rather than being embedded in two full rasters

So let's only send the pixels of the key and fill to the network

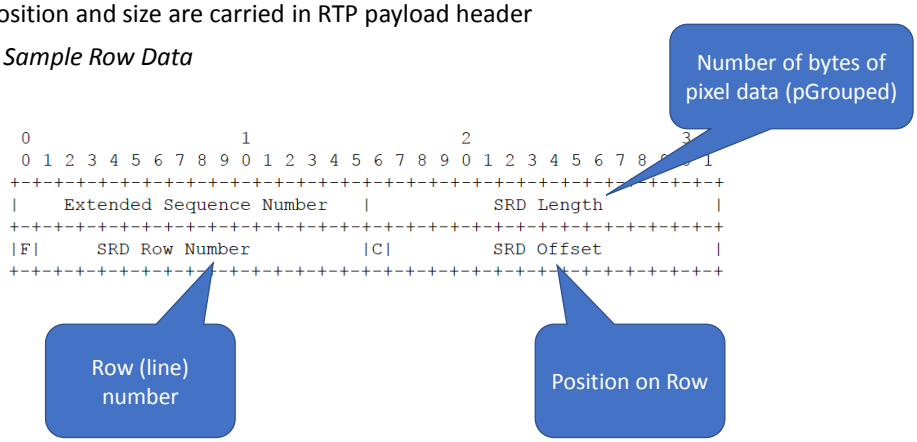
Sub-Rasters in ST 2110

- 2110 offers the ability to have a pixel map of up to 32K x 32K
  - Means we can have any arbitrary image size below that
    - Besides what's in 2110-20
- 2110 also carries the raster row and column position for each packet
- To carry a rectangular sub-raster:
  - Packetize the raster like a regular image.
  - Row / Column for upper-left corner is 0,0
  - Timestamp it with the same timestamp and TROffset as the associated BG
  - When received, it will be in the upper-left corner of the BG, so to speak
- *This is possibly useful, but requires ability to move it in the keyer*

Mechanism for Sub-Rasters

Image position and size are carried in RTP payload header

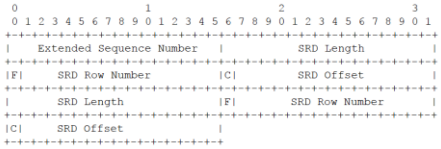
"SRD" is Sample Row Data



Example: a 192x108 logo in 4:2:2

- 2110 packs pixels into bytes using pGroups
  - Smallest number of pixels that they completely fill an integer number of bytes
    - For 4:2:2, pGroup size is 5 bytes containing 2 pixels
  - 192 pixels @ 2.5 bytes per pixel = 480 bytes.
    - This is a nice small IP packet, or a bit less than a third of 2110's MTU.
  - 108 rows of 480 bytes = 51,840 bytes.
  - A full 1920x1080 frame is 5,184000 bytes.
  - This example takes 10% of the HD frame's bandwidth in 108 small packets
- To build this transmission, we use 2110's rules
  - Start at upper left corner (like forever) and scan right until the end.
  - Finish at bottom right

Peculiarities



- The pGroup rules place some limitations:
  - pGroup size limits horizontal placement resolution
    - In 4:2:2, pGroups are 2 pixels, so placement on even 2-pixel boundaries
    - Other sample structures and bit-depths will vary as will placement limitations
- Packet sizes are arbitrary (up to the 2110 limit)
  - Can have from 1 to a full packet of pGroups per packet
  - No strict rule (except if BPM is used)
- The Continuation (C) bit can let two rows reside in a packet
- Up to 3 rows can be in one packet (like our example)

Other complexities

- Since 2110 isn't on fixed-bitrate cable, 4:2:2:4 is quite viable
  - 2110 doesn't have 4:2:2:4 (yet)
- 2110 has specification for KEY (alpha) signals
  - Must be compliant with RP157
  - RP 157 specifies a full-frame key signal, Y-only. No benefit to use this.
  - For a 2110 KEY at 10 bits depth, pGroup size is 4.
    - Means H placement resolution is halved. This means that the 4:2:2 fill has to abide.
- For transport today of the key element, 4:2:2 with no chroma would seem the best option.

Transmission Options

- Send the sub-raster aligned to the upper-left of the background frame
  - Simpler for keyers that can place the logo internally
  - Not useful for simple keyers
- Send the sub raster aligned to the desired target position in frame
  - Enables simple keyers
  - Enables placement at the origin, not keying location
    - Artistic / workflow choice
- Send the sub-raster aligned to no stream
  - No reason not to do this
    - Receivers would receive a stream of sub-rasters and do what they want

Timestamps

- The timestamp used to send the sub-raster is the same as the BG video timestamp
  - Makes best sense for processing when keying
  - If sub-raster is at upper-left origin, this is valid, device has to do final placement based on internal knowledge
  - If sub-raster is placed elsewhere, what breaks (if anything)?
- When sending an unassociated stream, arbitrary timestamps at the same framerate as the intended target can be used
  - Responsibility of receiver to re-align
    - Worst-case temporal placement error +/- 1 frame

  May simply be received and consumed into another process

Other Considerations

- The key and fill streams will require their own SDP objects
  - (Can't use the KEY construct of 2110)
- Something like:

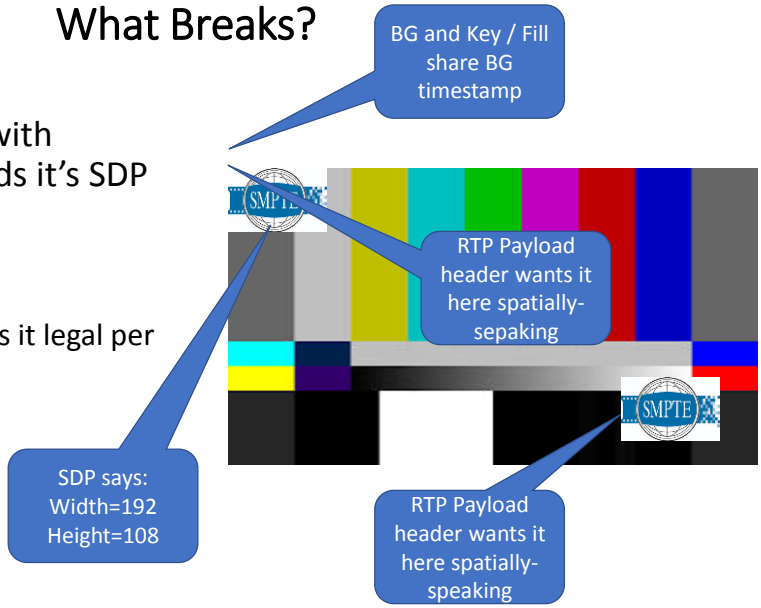
```
a=fmtp:112 sampling=YCbCr-4:2:2; width=192;
height=108; exactframerate=60000/1001; depth=10;
TCS=SDR; colorimetry=BT709; PM=2110BPM;
SSN=ST2110-20:2017;
```

As mentioned before, the ":2:2" part of the key signal is not used

Maybe empty data space for future keying use?

What Breaks?

- Sending a sub-raster with placement that exceeds it's SDP X by Y
  - Probably blows up implementations
  - Although it's bizarre, is it legal per the Standard?



Opportunity / Limitations

- Sending pGroup-aligned sub-rasters means H-placement granularity is limited by the pGroup size
  - It is expected that the keyer will work at the pixel level to overcome this issue
- If hard-cut keying is acceptable, it can be done in the switch
  - requires code in the switch
  - Horizontal pGroup alignments and packet size between the BG and key / fill packets are not fixed
    - Could cleverly use BPM and small packets, still very limited alignments
  - Packet rebuilding around the sub-rasters (and possibly every packet thereafter) may be required.
  - Only hard-cut / rectangular keying is supported
    - Using linear keying, the rectangle would simply outline the desired image

Other Applications

- Logo Keying is one example to illustrate the concept.

Could also be used for

- Character Generator insertion
- Lower third graphics
- Text Crawls, stock tickers
- Subtitle insertion
- Rating Codes
- On-screen advisories
- Picture-in-Picture displays

Is This Useful to Pursue?

- Could have significant utility
- Offers bandwidth conservation
- Hard-keying per the example is nasty except for direct display. The primary value-adds here are actually:
  - Atomic transport of the key and fill, saving bandwidth
  - Ability to send target coordinates with the stream
- There will be some work:
  - Is this an edit of 2110-20 or maybe an RP (I think latter)
    - Rules about what's allowable
      - e.g. key / fill dimensions can't exceed dimensions of BG frame
    - Association of key / fill to a stream

Application Thoughts

- By using arbitrary positioning via the RTP header, x-y motion can be achieved, moving the key around the screen. This could be in conjunction with dynamic dimensioning of the key.
- Multiple streams with / without position information could be used to achieve multiviewing displays
- Limited bandwidth means that multiple key / fill steams can be associated with a background (read:program) for selective use downstream.
  - e.g. rating code graphics – send them all, insert according to rating
  - personalized network logo bugs
  - etc.

Does it Have a Future?

- I will offer this presentation (in shorter form) to the 32NF80 DG-SVIP group (the “2110” drafting group) and see if there are others interested.
- If so, a Project Statement can be crafted
- Lower priority than other current work, unless I’m proven wrong
- Should it be an addition to 2110-20? (not my first choice)
- Maybe an RP – could still require tweaks to 2110-20

- Are you interested? Let me know – coordinates to follow

Thank-you! Questions?

Efficient Carriage of Sub-Rasters With ST 2110-20

Paul Briscoe, Televisionary Consulting

televisionary@teksavvy.com | +1 647 460 4466

On behalf of

Evertz Microsystems, Burlington, Ontario, Canada

Thank you to our Media Partners

IP SHOWCASE THEATRE AT IBC2019 : 13–17 SEPT 2019 25